Real-time power system dynamic simulation

Bousnane, Kafiha

Real-Time Power System Dynamic Simulation

A thesis presented for the degree of

Doctor of Philosophy

by

Kafiha Bousnane (BSc)

University of Durham

School of Engineering

and Applied Science

October 1990

ABSTRACT

The present day digital computing resources are overburdened by the amount of calculation necessary for power system dynamic simulation. Although the hardware has improved significantly, the expansion of the interconnected systems, and the requirement for more detailed models with frequent solutions have increased the need for simulating these systems in real time. To achieve this, more effort has been devoted to developing and improving the application of numerical methods and computational techniques such as sparsity-directed approaches and network decomposition to power system dynamic studies.

This project is a modest contribution towards solving this problem. It consists of applying a very efficient sparsity technique to the power system dynamic simulator under a wide range of events.

The method used was first developed by Zollenkopf [117]. Following the structure of the linear equations related to power system dynamic simulator models, the original algorithm which was conceived for scalar calculation has been modified to use sets of $2 * 2$ sub-matrices for both the dynamic and algebraic equations.

The realisation of real-time simulators also requires the simplification of the power system models and the adoption of a few assumptions such as neglecting short time constants. Most of the network components are simulated. The generating units include synchronous generators and their local controllers, and the simulated network is composed of transmission lines and transformers with tap-changing and phase-shifting, non-linear static loads, shunt compensators and simplified protection. The simulator is capable of handling some of the severe events which occur in power systems such as islanding, island re-synchronisation and generator start-up and shut-down.

To avoid the stiffness problem and ensure the numerical stability of the system at long time steps at a reasonable accuracy, the implicit trapezoidal rule is used for discretising the dynamic equations. The algebraisation of differential equations requires an iterative process. Also the non-linear network models are generally better solved by the Newton-Raphson iterative method which has an efficient quadratic rate of convergence. This has favoured the adoption of the simultaneous technique over the classical partitioned method. In this case the algebraised differential equations and the non-linear static equations are solved as one set of algebraic equations.

Another way of speeding-up centralised simulators is the adoption of distributed techniques. In this case the simulated networks are subdivided into areas which are computed by a multi-task machine (Perkin Elmer PE 3230). A coordinating subprogram is necessary to synchronise and control the computation of the different areas, and perform the overall solution of the system.

In addition to this decomposed algorithm the developed technique is also implemented in the parallel simulator running on the Array Processor FPS 5205 attached to a Perkin Elmer PE 3230 minicomputer, and a centralised version run on the host computer. Testing these simulators on three networks under a range of events would allow for the assessment of the algorithm and the selection of the best candidate hardware structure to be used as a dedicated machine to support the dynamic simulator.

The results obtained from this dynamic simulator are very impressive. Great speed-up is realised, stable solutions under very severe events are obtained showing the robustness of the system, and accurate long-term results are obtained. Therefore, the present simulator provides a realistic test bed to the Energy Management System. It can also be used for other purposes such as operator training.

ACKNOWLEDGEMENTS

I would firstly like to thank Professor M.J.H. Sterling for his supervision, guidance and support throughout the project. I would like to express my gratitude to Dr. M.R. Irving for his precious help and continuous assistance. Thanks are also due to Mr. J.O. Gann and Dr. J.R. Bumby for their valuable comments. I also would like to express my thanks to the Algerian government for the sponsorship, and to all the member of staff of the School of Engineering and Applied Sciences at the University of Durham, especially OCEPS group. Finally, I would like to thank my family and friends for their encouragement.

CONTENTS

STATEMENT OF COPYRIGHT

DECLARATION

The work contained in this thesis has not been submitted elsewhere for any other degree or qualification and that unless otherwise referenced it is the author's own work.

CHAPTER 1

INTRODUCTION

1.1 POWER SYSTEM DEVELOPMENT

The discovery of electromagnetic induction in the early 1800s gave birth to the electricity industry which has grown to occupy a key place in the social and economical life of developed nations.

The first station to generate public dc electricity of 30 KW for lighting started functioning in 1882. Right from its beginning, the electricity industry faced the problem of load/generation imbalance. By 1890 Tesla invented the polyphase ac motors and conceived a polyphase power system. The alternating system was then adopted as a standard because of its advantages over the dc system. Electrical power started being used for other purposes with the invention of the motor. More current was needed to meet this demand which posed the problem of unacceptable voltage drops. The solution to this problem was either to use short transmission lines or transport the electrical energy at a higher voltage. This gave rise to transformers and the further expansion of electrical systems. The necessity for standardising the frequency became apparent. Europe standardised its frequency on 50 Hz and United States on 60 Hz.

The increasing demand over the years obliged the utilities to transmit the power at higher voltages. With the variety of voltage levels the utilities standardised on certain voltage norms for each stage of the power systems hierarchy. These norms may vary from country to another. The voltage at the generation level varies from 480 V to 25 KV but only a few voltages within this range are normal values. To minimise the voltage drops electrical

energy is transported at voltages ranging from 115 KV up to 765 KV with line capacities ranging from 100 MVA to more than 4000 MVA. Subtransmission lines, which are shorter than the transmission lines, serve as an intermediary between transmission and distribution. Some bulk loads such as large industrial factories are often supplied at this voltage level. The network voltages continue stepping down to the consumer's voltage level (240 V in UK).

The high cost of generation facilities encourages inter-area and intercontinental connections. As power and interconnection levels increase, the role of stability as an operating factor increases. It frequently becomes necessary to solve larger systems, with increased detail of modelling, over longer response times. The occurrence of drastic disturbances such as the USA 1965 black-out also encouraged the utilities to modernise their energy control centres by using powerful computers to gather more power system information, sophisticated algorithms to achieve real-time calculations and to undertake new research projects.

1.2 HARDWARE AND SOFTWARE FOR SIMULATION

During the past two decades advances on power system studies have been made on three fronts: software improvements, new analytical tools, and applications of new computer hardware [2]. Development of one of these areas only may not be very adequate because these three problems are inter-linked. Sophisticated software may not perform well on poor hardware and vice-versa.

In recent years development of the hardware technology has significantly reduced computer prices. This has favoured the development of special-purpose machines which are either fully dedicated to one single problem or a class of problems. The first category includes the multi-microprocessors which are used for one particular application. The second includes Vector or Array Processors which are very efficient in solving matrix and long vector operations. Therefore, these machines are not fully dedicated to one specific area such as power systems, but can be applied in any domain which exhibits the characteristics required for the efficient use of Array Processors. The availability of such powerful

computation favoured the development of fast numerical methods which can exploit the advantages offered by both the hardware and the problem to be solved.

At the same time the complexity of interconnected power systems has increased with the expansion of power systems and the utilities expectations of efficient real-time control systems. The aim of realising a real-time comprehensive integrated power controller, including all the necessary details related to all aspects of a large power system, is far beyond the capabilities of today's computers and algorithms. At present many assumptions are accepted for the sake of getting approximate results, which are realistic enough to ensure reliable and secure power supply.

This has motivated research to break down the control problem into individual topics such as economic dispatch, security assessment, and so on. Specialising on one of these areas would surely help in developing special techniques which would suit the characteristics of the different problems. The new algorithms are first investigated and developed off-line and then need to be tested on-line. Testing such algorithms on real power systems is infeasible. This has produced a necessity for the simulation of power systems over a long period.

1.2.1 Definition of simulation

Mathematically, two things are considered as similar if they can be mapped together in such away to preserve only certain relations (not necessarily all).

Man's progress in science and technology is actually related to his success in the use of analogy and simulation by observing the natural world around him. For instance, a plane is a simulation of a bird, a submarine is a mechanical copy of a fish, a computer is a simulation of human intellect.

The simulation of a system, a phenomena, or an object is achieved through an approximate model. This model can be either physical i.e. a

miniature of the object of simulation [100], or mathematical. The mathematical model represents the object of simulation and its functions analytically using sets of equations and data which can be manipulated by classical methods using calculators, or computers [96]. Today, with the wide spread of computers, digital techniques are becoming the main tool of simulation. Using mathematical models, highly complicated systems coupled together can be condensed and simulated realistically over a wide range of conditions.

1.2.2 Development of the hardware

A survey on the different generations of computers has been presented by Hockney and Jesshope in reference 58, and Hwang in reference 61.

The first commercially produced computer UNIVAC1 appeared in 1951. Since then the technological development of the hardware components together with the exploitation of parallelism at all levels of the computer architecture resulted in powerful computers. Comparing the number of instructions per second of the EDSAC1 (1950) and Cray 1 (1975) a total performance improvement of 10^6, of which only about a factor of 160 can be attributed to improvements in technology, has been achieved over the time of about 25 years [58].

Babbage has pioneered many of the basic ideas of computing in his work on the difference and analytical engines over a 100 years ago [58]. The idea of improving the performance of a machine by using parallelism had occurred to him when he invented the analytical engine.

Early computers (EDSAC, UNIVAC1) performed an arithmetic operation on two 32 bit numbers bit by bit (bit serial). To reduce the time taken to perform such operations from 32 machine cycles to one cycle this calculation was performed in parallel (bit parallel). This necessitated the possibility of reading all the bits of a word in parallel. Such computers are described as word-serial bit-parallel processors. The first commercial computer with parallel arithmetic was the IBM 701. The IBM 704 was the first machine to use hardware floating-point arithmetic units (1955).

The IO was a problem for the machines of that time. Data reading and writing had to pass through a register in the arithmetic unit thus preventing the machine from performing useful calculations at the same time. This was alleviated by using a separate computer to transfer data to or from the slow peripherals (card readers, magnetic tapes, line printers) and the main memory of the computer. This can be considered as an early multiprocessing.

ATLAS (1961) was the first machine to use a complex multiprogramming operating system. This system has also made early use of dividing the memory into independent banks which can be accessed simultaneously in one cycle (parallelism in memory). This has made possible the use of pipelining by overlapping the following steps: instruction fetching, addressing, operand fetching and arithmetic operation.

In order to efficiently exploit pipelining and parallel features such as multiple arithmetic units, registers and memories it was necessary to look ahead in the instruction stream and determine the instructions which can be performed in parallel without affecting the logic of the program. This enabled the potential parallel instructions to be carried out in an optimal way.

The slow memory access has initiated the implementation of cache memory as an intermediary storage between the main memory and the arithmetic registers. The machine which has used this facility together with the use of two separate pipelined arithmetic units (a floating-point unit and an integer addressing unit) and a pipelining instruction processing was the IBM 360/95 which appeared on the market in 1971.

The first commercially successful special-purpose pipelined vector processor was Cray 1. This computer has achieved measured rates of 130 Mflops on suitable problems such as matrix multiplication.

A relatively inexpensive range of special-purpose computers which involved parallelism arose in 1973. These machines use multiple processing units which process arrays of data and are referred to as Array Processors. This name

does not imply that these computers include arrays of processors but it is derived from their function. Among these machines are those of Floating-Point Systems (FPS) which include multiple processing units interconnected by a number of data buses. These machines are attached to host computers which can be minicomputers or main-frames and perform 38 bit floating-point arithmetic calculation in separate pipelined multiplication and addition units, and 16 bit counting and address calculation in an independent integer arithmetic unit. Three memories for data, tables, and program, and two pads of registers are provided with multiple paths between each memory and each arithmetic unit. These machines are highly cost effective and may enable calculations which are prohibitively expensive to become feasible at reasonable prices.

Prior to the early 1960's, nearly all power system analysis was performed on network analysers, which were special-purpose analog computers designed to perform static network solutions of bulk power systems [2]. These analysers became inefficient when power systems started expanding rapidly. The invention of digital computers and their availability provided an ideal solution to the power system analysis problem.

The computers of the late 1950's and early 1960's are probably best characterised by the IBM 704. These machines were widely used by the larger utilities for engineering computation, despite the slow computation time for large problems such as transient stability. Still these digital methods gained wide acceptance and ushered in a new age in engineering computation [2].

Since then both the computer capability and the complexity of power systems have spectacularly expanded. During the mid 1970's the power systems in the USA expanded by a factor of four whereas the computers capability expanded by a factor of five [2]. Researchers expected these computers to solve this size of networks without difficulty. However, all the signs were to the contrary. This clearly shows that the relationship between the network size and the computer power is not linear. More emphasis has subsequently been placed on the development of both hardware and software (including algorithms) as complementary tools.

Software costs are rising while hardware continues to go down in price. This favours the development of special fast hardware to solve a particular problem. The algorithms and the hardware structure should be carefully matched otherwise no great benefit from special-purpose computers can be gained [2]. At the same time one should consider the rapid change of both numerical methods and the electronic devices. Therefore, before starting any of these projects care should be taken in investigating the possibility of expanding the machines to be developed and replacing the algorithms, if necessary, without sacrificing the performance of the hardware.

The Array Processors are relatively inexpensive, thus efficient algorithms, operating on cheap hardware, can produce a cost/performance that is very attractive. Therefore, these machines can be considered as the first competitors with the special-purpose multi-microprocessors; especially, if the flexibility of the Array Processors and their potential use in many areas are taken into consideration.

1.2.3 Development of the software

As a counterpart to the development of special-purpose hardware, the development of special-purpose software languages has received the attention of researchers. To take advantage of the rapid changes in computing resources and their architecture, special-purpose simulation languages such as the one developed at the University of Durham by Sterling et al.[105] are widely used. This is capable of running on a wide range of machines and can support a variety of solution algorithms necessary for diverse power system problems. At present the language supports IBM PC's and MOTOROLA 68020 transputers and can generate code in languages such as FORTRAN and C [105]. This approach uses special techniques to evaluate various mathematical functions such as derivatives, integrations, matrix manipulation etc. Keyhani [75] has also used a similar simulation language. This has enabled the author to develop a very flexible simulator which can serve many purposes such as teaching, and support various models and computational techniques.

Most power system simulations require the inversion of very large sparse matrices. Using general-purpose sparsity techniques at each iteration can be very expensive. In order to solve this problem efficiently, without sacrificing the accuracy and the stability of the system, intensive research has concentrated on investigating algorithms which can provide fast and reliable solutions [1,23,37,66,82,111,113,117]. Commercial packages have been developed and put at the disposition of the user.

1.3 UTILISATION OF SIMULATORS

Theoretically any physical object or phenomena can be simulated. However it is not always possible to provide the necessary data. Analytical simulators can be applied in many areas such as medicine, defence, etc. Each of these areas is in itself very wide.

Power system simulators came into existence with the expansion of power systems and the desire for developing adequate Energy Management Systems (EMS) to be used in control centres to monitor system operation under normal and emergency conditions. To satisfy such requirements simulators must be able to analyse the stability of a system and its ability to sustain a disturbance such as the loss of a line or a generator and yet provide the required energy at a low cost.

Power system modelling reflects the influence of the different components of the power system on stability at different stages of the response. The ultimate aim of any research in this field is the representation of all these components in great detail leading to an ideal simulator. However, because of the complexity of power systems and the phenomena which occur in them it became convenient to recognise three distinct areas or modes of disturbance called short-, mid-, and long-term, covering post disturbance times of up to 10 seconds, 5 minutes, and 20 minutes, respectively. The typical time steps of interest are 0.1 to 0.2 seconds for short-term and 1 second or longer for

mid- and long-term studies. Another classification which denotes the short-term problem "transient stability", while anything longer is termed "dynamic stability", is now also being used.

The simulation of electrical power systems can also be used for other purposes such as planning and design.

1.3.1 Transient stability studies

Transient stability studies emphasise the rapidly responding components of electrical power systems.

Real-time instability detection for a general power system is rather an unsolved problem at present. And yet what is really needed is faster than real-time determination of instability, which is even more difficult [87]. This would allow for the detection of the problem in advance so that preventive measures can be taken to prevent the propagation of the disturbance.

Transient stability is characterised by angular swings between synchronous machines resulting from temporary imbalances such as short circuits. Therefore, it can be said that the generator models are the nucleus of the short-term simulation

1.3.2 Dynamic simulation

For many years transient stability simulations over a period of approximately one second were the norm. Then many system analysts realised the necessity to examine longer term phenomena [2] to analyse the cascading effect and develop a better understanding of the factors which contribute to this problem, although the unpredictability of these consequential events is a serious problem. Cascading can be defined as a sequence of failures and outages of important power system elements such as generating units and customer loads caused by a severe initial disturbance which has not been damped. This analysis necessitates the consideration of slower components which have a more pronounced effect over longer periods.

The dynamic simulation models are more concerned with representing the slow oscillatory system power balance (mid- and long-term components), assuming that the rapid electrical transients have damped out. Thus dynamic simulations are widely used to analyse system behaviour caused by active power imbalance and the subsequent frequency excursions. The simulation of voltage regulators and controllers such as AVRs, transformer tap-changing, and compensators allows also the use of dynamic simulations in VAr problems and the resulting voltage transients such as voltage collapse.

When systems are assumed to include slow transients and steady state only, this problem can be solved at the expense of some accuracy, and more than real-time speed-up can be achieved provided efficient hardware and skilful software are available. This inaccuracy has two sources: the simplified system models and the associated data, and the solution methods. Nevertheless, it should be noted that the results obtained from such studies should be acceptable and realistic. With the development of fast algorithms and better computational resources it is hoped that more details (fast models) can be included in these simulators. This perspective has been reported in references 108, and 44.

1.3.3 Planning and design

Simulators can be applied efficiently in planning and design. The use of approximate simulators for planning is accepted by most utilities, because developing detailed simulators, which require small time steps and long run periods, necessitates enormous computer cost. Therefore, a long-term simulator would easily serve this purpose. Whereas plant design usually requires a thorough modelling of the equipment under consideration. Therefore, great detail and very accurate data are needed. This may necessitate the simulation of the element under consideration only.

Planning includes the extension of existing systems to meet the increasing load, improvement of existing plants and so on. This can be either physically-oriented e.g. introducing new plants and considering emerging concerns such as emission control, or related to the development of power systems control. In

this case new disciplines may be required to improve the performance of control centres. The use of simulators for EMS is the subject of the next subsection.

If the predicting tasks such as unit commitment and load prediction can be considered as a planning aspect of power systems operation, this area can be subdivided into three categories:

(1) The long-term class which includes the development of the actual power system either by modifying the existing plants or by constructing new ones. This has a time scale of 6 months to 10 years [102] or perhaps longer.
(2) The mid-term planning includes system maintenance which can last from one week to a few months.
(3) The short-term category includes plant ordering, load prediction etc. This period varies from a few minutes to a few hours.

The role of the simulator in this case is to provide a test bed for these developments and predictions so that a choice can be made from the available alternatives.

Design is mainly concerned with the development of new equipment to ensure a better performance of the power system.

1.3.4 Energy Management Systems

The occurrence of dramatic blackouts is rare. However, because of their drastic economical and social consequences, the utilities and energy companies are concentrating a lot of effort and investment to prevent such events. Owing to the increased demand for more sophisticated real-time automatic control in normal, emergency and restorative states, and to the necessity of improving the handling of traditional concerns such as economic dispatch, and security assessment, a further growth in complexity of power system analysis and control has occurred. Better algorithms and more powerful computers are needed to meet these requirements. The direct on-line application of the EMS packages is

not advisable. The simulator provides a realistic means for testing the efficiency of such new algorithms before applying them to real power systems.

1.3.5 Operator training and teaching

With the improvement in the reliability of power systems it is becoming more and more difficult to gain operating skills and expertise. This is because of the rarity of severe incidents. The severity of the blackouts made the utilities realise the need for improved operator training. The most efficient way of training operators is the creation of artificial operational environments by using simulators. This provides the trainees with the necessary training opportunities ranging from small disturbances to major events such as islanding, loss of generation and load, blackouts etc. This alternative is realistic since the use of actual power systems to create such operating problems is not desirable[31]. It is also more economical than the creation of actual physical breakdowns on miniature systems, because the occurrence of these incidents often results in material damage. The main advantages of training simulators over other means of instruction are:

(1) The capability of accelerating and expanding the experience of the operator in a short span of time.
(2) Presenting the operator with different responses to various contingencies by using scenarios of events.
(3) The ability to analyse the system response with the help of the instructor. Therefore, the results of the simulator should be realistic otherwise the trainee may learn incorrect procedures.
(4) Provides trainees with the opportunity of training on their own systems otherwise they may not be able to operate their real systems efficiently.

For these reasons most of the utilities have developed training simulators to help the trainee develop his operational decision making expertise.

The training simulator consists of three fundamental parts:

(1) A power system simulator representing the physical network and the other operational personnel who interacts with the operator.
(2) The control system which includes security analysis, load frequency control etc.
(3) The interaction between these two schemes by simulating the gathering of the data, its processing and transfer from the network to the control centre and the man-machine interface.

The use of simulators for teaching and training purposes necessitates advanced interactive graphical displays. For training purposes the man-machine interface should be an exact replica of that in the control centre [31] since this is the operator's real environment.

The simulator should be a faithful representation of real life systems to provide the trainee with a realistic environment. Some assumptions can be included in the power system modelling as long as enough information would be provided. Therefore, since training simulators are meant to teach operation skills of system restoration after disturbances in a minimum time, preventive steps if an abnormal state is detected, and emergency operation, most training simulators are not as detailed as those developed for transient and dynamic stability studies. Longer sampling intervals varying from 1 to 10 seconds (typical values of 3 or 5 seconds are usually used) are used for training simulators. Two distinct time steps can be used [69,109]. A large time step for the discretisation of the slow dynamics and a shorter time step for the faster models (approximate transient stability) can be applied. However, if the computer resources available allow for more detail, shorter time steps and less simplification of the power system can be considered.

The authors of reference 97 were able to produce a real-time training simulator using a single time step of 0.1 second. The machine used for this study was a Vector Processor. The model used included simplified dynamics of turbine, governor, exciter and generator electrical and mechanical equations. The transmission network models included transmission lines, transformers, and so on. Automatic controls and switching equipment, were also modelled as well

as different types of protection. With this time step more realistic training can be achieved since more accurate results can be obtained.

The simulator developed by Lo et al. [79] for teaching purposes offers the student the opportunity to control and alter the simulated system and assess the consequences of these alterations. Among the interventions allowed are, change of the load and generation, voltage level control, optimal generation scheduling, and contingency and planning studies.

1.4 REVIEW ON SIMULATION

For many years researchers concerned with simulation have concentrated their efforts on transient stability studies. Mid-term and long-term simulations are relatively recent developments. In 1965 Concordia presented a paper on the effect of prime mover speed control on the relatively long-term power swing [55]. EPRI is one of the pioneers in long-term stability studies using digital and hybrid computers [19,27,55,85] and started working on this field since the early 70's.

With the growth of digital computers and the availability of analog machines, hybrid simulators have emerged. Excellent results have been obtained by Ott et al. [85] by using hybrid simulation of the long-term dynamics. However, the use of analog computers is not flexible. With the enhancement of digital computers these systems have been abandoned because of their complexity.

The traditional simulation problem was the transient stability, on which the vast majority of effort and literature have concentrated. One of the best of this rich literature is the review presented by Stott [106]. Many aspects of power system modelling such as the discontinuity constraints are presented and suggestions are given to overcome these problems. Also a good survey on a few numerical methods which are used for solving the differential and the network equations is presented together with their merits and implementations.

Johnson et al. [72] have used a multi-numerical method approach for power system dynamic studies. A tuneable factor is used to pass from one method to another according to the time constants and time steps. This approach adequately overcomes the stiffness problem and greatly reduces the errors involved. Therefore, this method is efficient for transient stability studies because of its accuracy which is crucial for the simulation of this power system operating mode. The drawback of this technique is the instability which can be caused by the explicit Euler method.

Taoka et al. [110] developed an efficient transient stability simulator. This simulator was applied to an Array Processor AP-120B hosted by a DEC VAX-11/780 computer. The authors of this paper reported that they had faced the problem of memory limitations. The limitations of the memory space for large networks may be attributed to the fact that the algorithm used require full matrix operations. In an attempt to overcome this deficiency, to a certain extent, the authors of this paper used a partitioned method in which a special technique is used to transform the nodal admittance matrix of the network. A performance of about 5 times faster than the single processor simulator running on the host only was obtained. Real-time simulation was achieved for small networks with a time step of 10 milliseconds for solving the differential equations by the modified Euler method. The 103 bus and 60 generator network execution time was 3 times slower than real-time.

Long-term simulation started gaining popularity when the importance of longer term phenomena had became apparent because primary disturbances are sometimes not damped out during the transient period. These can result in devastating problems such as islanding, loss of generation or loads, and the blackout of an area or even of the whole system.

Reference 27 proposed long-term stability analysis to determine some corrective steps which can be used at the system design stage to avoid the cascading problem in the case of severe power system disturbances. Among these suggestions:

(1) Inhibiting the load voltage regulators at the distribution substations when the frequency and the voltage are low.
(2) Modifying the characteristics of boiler controllers to widen the range of frequency to which they will respond.
(3) Increase the range of frequency over which power plant auxiliaries will operate at full capacity.

The first measure showed that less load shedding is required although the load voltage is no longer regulated to attain constant load voltage. The two last suggestions had a detrimental effect, since they had rather accelerated the load shedding instead of establishing the load/generation balance at minimum cost.

In 1975 Davidson et al. [19] presented a survey of major disturbances and the factors which lead to the propagation of the problem over parts of the network or over the whole power system. The authors identified that most of the problems occur during week days between 7 am and 4 pm. The major initial causes of these events were natural conditions such as strong wind, storms etc. 80 % of these disturbances occurred initially in the transmission system because this was the most widespread part of the system and was thus more exposed to potential problems. However, disturbances caused by human factors and equipment failure were often the most devastating. Although most of the initiating events were moderately severe, because of the existence of worsening factors such as a malfunctioning equipment, most of these events resulted in both islanding and load shedding and the recovery of the system lasted hours. Therefore, this reference can be used for establishing realistic scenarios to test the simulators which are developed.

The work of Hemmaplardh et al. [55] describes a revision of the previous EPRI long-term simulator. Slow variables such as boiler models are presented and the numerical methods used are mentioned. The partitioned method is applied to solve the differential equations and the power flow models separately. The differential equations are algebraised by using the implicit trapezoidal rule. The dynamic state variable mismatch is evaluated by the Newton-Raphson

algorithm. The load-flow is solved using the decoupled method with accelerating factors.

Only a few physical simulators are available in the world. These are not widely adopted and most of them are not in use at present [100]. This is due in part to the space required (only a few dozen busbars can be built) and the difficulty in designing accurate scale models. Also these simulators are not as flexible as the numerical simulators. To allow for the simulation of larger systems, the physical simulators are assisted by computers. In this case some of the components are represented by mathematical models and others are represented physically. The Energy Systems Research Centre at the University of Texas [100] built a three phase scale model which includes Energy Management System control in order to allow the operator to perceive the physical concepts of what is represented by only a mathematical model in the numerical simulators. This simulator is capable of performing stability studies, operator training, teaching etc. as well as component research. As far as teaching and operator training is concerned, this simulator may be considered as a means of giving the trainee the opportunity for real life experience with blackouts and other disturbances. However, the proposed system is too small (4 nodes). In a discussion on this reference it has been pointed out that such a system may not be very efficient for control system research such as state estimation, security analysis etc. It has also been described as an expensive and inflexible system.

This physical simulator includes simulated nuclear, fossil and hydro plants. These plants together with their turbines are represented by dc motors controlled by a microprocessor. Actual loads such as induction motors and passive loads are included. The system also includes tap-changing transformers which are initiated by the SCADA.

Keyhani [75] developed a very flexible multipurpose simulator using the Speakeasy simulation language, to be employed as a research tool for simulation studies. This simulator is also a valuable tool for teaching. It includes modules dealing with many problems such as decoupled load-flow, transient stability

with different models and so on. The flexibility of this system allows the user to interact with the simulator and modify or add new power system models, according to the requirements of the problem to be solved, in the form of commands. These commands are executed immediately and do not require any programming effort.

An impressive simulator combining long-term stability analysis with transient stability has been developed by Stubbe et al. [108]. This is achieved by dynamically varying the time step following the required accuracy. The method used for this purpose is the implicit multi-step multi-order backward differentiation Gear technique [46]. A range of 400 time steps varying from 0.0001 seconds to 40 seconds is used. The authors have reported that this simulator is capable of restoring the system model from a blackout. In a discussion on reference 109 it has been stated that BPA (Bonneville Power Administration) has also completed an extensive blackout restoration simulation program.

Another unified transient/mid-term/long-term simulator has been developed for EPRI by Frowd et al. [44]. These authors modified the swing equation by using a flexible artificial damping factor. According to the conditions of the system the damping can be increased to suppress inter-machine oscillations. This factor is decreased if the algorithm has to switch over from long-term to transient stability to allow for more accuracy if fast swings are detected. The time step is also adjusted to speed-up the solution.

The results obtained from this study are very promising. Compared to using the transient stability program as a stand alone package the execution time was speeded up by a factor of 75% and the difference in the accuracy of the unified system was small. Although this simulator is faster than the transient stability program, the timings obtained in tests on a 68 bus network were slower than real-time.

Susumago et al. [109] have succeeded in developing a well equipped comprehensive operator training simulator. Details of the power system, the control system, and the control centre environment are presented. This simulator

uses a transient stability system for scenario building from breaker openings following a disturbance. Zhang et al. [115] also used the same approach for scenario generation. Time steps of 0.25 and 3 seconds are used for solving dynamic and load-flow models respectively.

In general the solution speed obtained in simulation is very dependent on the numerical algorithms which have been applied. Many of the mathematical methods developed for transient stability systems are also convenient for dynamic simulators. For instance, the presentation of the benefits of the implicit trapezoidal rule and its stability under a wide range of time steps (solution of stiff system without problems) [24,106] in comparison with the explicit methods such as Runge-Kutta technique has led to the wide adoption of the former approach for both transient and dynamic stability [44,93]. The bottleneck of the solution of large systems is matrix inversion. The following paragraphs present some of the widely used methods for solving this problem.

The LU decomposition is a well-known and widely used technique for solving sparse matrices. Many studies have been computed on Array Processors (mainly AP-120B) [2,82,84,113]. A large amount of work has been reported on the use of the sparsity-oriented LU decomposition technique for power flow, transient stability, and dynamic simulation.

Woo [113] used the LU Gauss elimination to solve a set of sparse linear equations for reservoir simulation. Reordering is carried out only once for each network topology and is performed on the host computer. The factorisation simulation needs to be run once for each solution. This scheme does not involve too much data transfer since the ordered matrix is evaluated and transferred only when the network topology changes.

Dodson [23] used the same method as Woo to solve the set of linear equations for load-flow studies (LU triangular decomposition) with a further improvement. He exploited the $2 * 2$ structure of the non zero elements of the Jacobian matrix. The pivoting which is generally required for numerical stability has been ignored since it is not necessary. The analysis step (ordering)

interchanges rows and columns of the matrix to reduce the number of generated fill-ins.

Rainbolt [94] used the upper triangular sparsity-directed method to simulate the transient stability of systems ranging from 49 busbars to 1454 busbars. An Array Processor (AP-120B) was used in parallel with a VAX 11/780 which is used as a host computer. The simulator running on the host alone was faster than the parallel simulator for small networks. For the 1454 system the parallel simulator was only twice as fast as the single processor simulator although the performance of the Array Processor alone was high. It can be concluded that the limiting factor was mainly the data communication between the two machines.

Alvarado et al. [1] and Enns et al. [37] have recently proposed a method which decomposes the inverse of the upper and lower triangular matrices into n factor matrices. These are partitioned into a number of sparse matrices. The number of partitioned inverse of the upper and lower triangular factor matrices is chosen in such a way that the number of fill-ins is optimised, by using a new ordering scheme. Another strategy is to minimise the number of partitioned inverse matrices by allowing a certain percentage of fill-ins. According to these authors this algorithm is superior to the Tinney schemes [111] for applications where the main burden is repeated direct solutions with sparse vectors. This approach is more oriented towards sparse vector systems, however it can also be used for full vectors. It has been developed for parallel processing where very large network direct solutions can be obtained by partitioning the inverse of $[L]$ and $[U]$ factor matrices. This method is performing poorly for serial applications. The Tinney scheme 2 [111] is better for solutions with full vectors and where the major computational burden is factorisation.

Some researchers have turned towards hardware development in order to provide cheaper simulations. In the last decade a lot of work has been done on the use of distributed microprocessors. They offer great potential savings in cost because the main hardware is very inexpensive. However, the most favoured hardware seems to be Array Processors, which can be hosted

by ordinary computers. This is mainly due to their efficiency when solving numerical problems, especially if these involve large vectors. The superiority of Array Processors over other distributed approaches can also be attributed to the use of the host computer which can provide further processing power and storage. A drawback of decomposed methods is the data transfer and the extra computation necessary to coordinate the different processors. There are reports of experiments in which the power system stability calculation has been speeded up by an impressive twenty times compared to serial approaches [2].

The efficient use of Array Processors requires the consideration of the following suggestions:

(1) The amount of data to be transferred between the host and the AP should be minimised, as well as the frequency of this data communication.
(2) The idle time of these two machines should be exploited to the maximum by performing independent calculations in parallel.
(3) Integer calculations should be minimised.
(4) Operations such as FORTRAN IF and GOTO statements should be avoided.
(5) Minimisation of data memory access is necessary. Operations between data memory require 3 machine cycles, thus reducing the performance of the AP. The calculation should be organised in such a way that a large amount of data can be extracted from the memory at each memory access.

Among the multi-microprocessors presently available is the *cm** designed and implemented at Carnegie-Mellon University [45,74]. This multiprocessor is used as an investigative tool to further develop this issue and solve various problems such as memory mapping, and alternate structures encountered in this field. The ultimate aim from developing such hardware is the possibility of replacing large conventional uni-processors or reaching a performance as high as these machines, although, this aim is apparently questionable [45] because of the restrictions on the number of processors which can be used in parallel. Also the uni-processor development is not likely to stand still. Single processor

computers are already exploiting parallel hardware designs and using pipeline techniques. Their prices have also decreased significantly over the years. Therefore, machines such as Array and Vector Processors supported by optimising compilers are remarkably cost effective. The gap between multi-microprocessors and conventional computers is thus expected to be maintained.

Durham et al. [28] suggest the use of a flexible set of machines instead of using dedicated ones for each sub-problem or group of sub-problems. This would allow for the variation of the configuration of the hardware and could handle the system efficiently in the case of component failure or maintenance. Software would have to be provided to control the allocation of the work among the available processors. Theoretically, there is no limit on the number of machines to be used in parallel; however, comparing the expected improvements and the expense involved, an optimum number of parallel processors can be determined.

Parallelism can also include conventional networking of ordinary minicomputers, distributed microprocessing, special-purpose computers and various forms of loosely and tightly coupled multiprocessing. Hatcher et al. [54] presented decomposed simulation software for transient stability studies. This algorithm has been implemented on minicomputers. The algorithm used by Hatcher et al. [54] is based on the same principle as the OCEPS decomposed simulator package which has also been applied using a minicomputer [92].

Research on distributed methods is not new and their application to power system problems by using parallel computation facilities came as a result of the potential decomposition of most of these problems, such as economic dispatch, state estimation, load frequency control etc. The theoretical aspect of this area has been developed as early as the late 1950's by Kron [51]. Since then many books have been published on this field [10,51] and its practicality has been proved in many papers [12,14,16,26,28,35,41,45,53,54,80,91,92,107].

There are several ways of approaching the network tearing, including sparse diakoptics. Most of these methods concentrate on tearing the network or partitioning matrices into smaller parts that can be solved in parallel on

eparate processors. As a consequence of the network tearing methods a new liscipline concerned with the automatic optimal sectioning of the system has emerged [67]. Most of the available literature in this field agrees that the solution of large sparse networks takes more operations with tearing than without. If the absolute computing time can be decreased significantly and the cost of extra processing is negligible, this method can be very promising.

1.5 OBJECTIVES OF THE RESEARCH

Developing a robust comprehensive real-time simulator capable of simulating a large power system requires the use of small time steps. This is currently beyond the power of even the largest computers. Fortunately, the time scales of interest can usually be separated with a corresponding reduction in computational burden. This was the major motivation for separating power system stability studies into transient and dynamic studies. The present simulator is of the second type.

Adopting this separation strategy can help in speeding-up the simulator, but this does not necessarily mean achieving faster than real-time simulation. To realise this aim, hardware and software also need careful investigation. Powerful computers and sophisticated mathematical algorithms are necessary. These two aspects have to be considered together, because if one of them is inadequate this would result in a less sophisticated simulator.

For this purpose an Array Processor (FPS 5205) which is used in parallel with a host computer (Perkin Elmer PE 3230) has been dedicated to the simulator. The centralised simulator which was previously running on a minicomputer (PE 3230), was modified to suit the architecture of the new system. The linear sparse matrices were solved by using the *Harwell* library which is based on the LU decomposed technique. Real time simulation of the IEEE 30 node test network was realised. However, the characteristics of the Array Processor and power system models were not fully exploited. This machine is capable of efficiently handling larger networks in real-time.

It was clear from the results obtained from this simulator [90] that most of the execution time was spent in performing the repetitive solution of the set of linear equations. Therefore, additional research was needed on the methods of solution applied to these equations.

In order to preserve the accuracy of the results and speed-up the simulator, more attention was directed towards enhancing the use of sparsity techniques. One of the best alternatives to replace the *Harwell* routine which was previously used in the conventional simulator is the *bifactorisation* technique [90]. Since this method seems to overcome most of the shortcomings of the former approach and has shown good results when applied to power flow studies [66] it has been adopted here.

The original prime motivation behind the development of the simulator was to create a tool for more efficient investigation of power system research projects. Also this simulator is readily applicable to operator training and teaching, since these usually require long-term simulation, a powerful man-machine interface, and a control package, all of which are available. The latest version of the simulator has also been used for planning purposes in a collaborative project with the CEGB and has also revealed its efficiency in this domain [68]. Analysis of the long-term response characteristics also allows the use of this simulator to assess power system characteristics at the design stage.

The simulator has also shown the potentiality of being used for voltage collapse studies since it includes devices which have the task of controlling the voltage, such as AVRs, tap-changing and phase-shifting transformers, and shunt compensators. A project is under development on this important issue.

The refinements in the simulation models and implementation of more sophisticated sparsity techniques which take advantage of power system properties is expected to improve the simulator stability, robustness and execution time. With the fulfilment of these aims larger networks can be simulated in real-time and the behaviour of these systems under severe disturbances ranging from a

few seconds to a few hours can be successfully represented. Thus providing a more realistic and efficient test bed for the development of control algorithms.

In addition to research into improved solution methodologies that exploit the sparsity of matrices, the increasing complication of simulation models and the need for faster solutions have motivated research into techniques to divide problems into smaller subsystems, which can be solved separately in parallel.

1.6 THESIS LAY OUT

This project is mainly concerned with the algorithmic aspects needed to speed-up the simulator so that larger networks can be simulated in real-time. The work is presented in seven chapters together with an introductory and a concluding chapter.

The simulator environment and its implementation as an integrated element of a coordinated control system is presented in chapter 2. The role of the different elements constituting the control system and analysis such as load frequency control, state estimation, and so on are also represented together with the function of the simulator supporting packages such as the loader, exact topology determination etc.

Algorithms associated with power system simulation require the repeated solution of large sets of linear equations of the form $Ax = b$. These algorithms are described in chapter 3 together with methods of building the non linear equations into this linear form. This chapter presents a review of appropriate numerical methods. A few well known iterative and numerical methods are presented. Then the justification for choosing the implicit trapezoidal rule for the discretisation of the differential equations and their simultaneous solution with the network equations using the Newton-Raphson method is given. Also an overview of matrix inversion methods is given with the emphasis on the Zollenkopf *bifactorisation* technique and its merits.

Chapter 4 includes the mathematical models of all electrical equipment and functions which are to be represented. These elements are:

(1) A simplified representation of the synchronous generator models together with their prime movers, boilers, and Automatic Voltage Regulators (AVR).
(2) Voltage and frequency dependent non-linear static loads.
(3) Π configuration of the transmission lines, and the phase-shifting and tap-changing transformers.
(4) Static compensators including capacitors as well as reactors.
(5) Simplified under-frequency, over-speed, under-speed and overload protection.
(6) Measurement data corruption and telemetry.

This chapter also presents the application of the general formulas of the trapezoidal method developed in chapter 3 to certain equations to illustrate the construction of the Jacobian matrix.

Other activities to be simulated, such as the re-synchronisation of a split system, and their relation with the topology variation are also presented in this chapter.

Chapter 5 presents the algorithms and the programmed models of the conventional centralised simulator running on the Perkin Elmer minicomputer PE 3230 and the parallel simulator applied to the Array Processor. The *Harwell* library using LU decomposition for solving the linear equations is described, and the process of building these factor matrices is included. Data and message communication between the Array Processor and the host computer is presented.

Chapter 6 is a further development of the concepts presented in the previous chapter. An original simulator algorithm which is based on a new technique for the numerical solution is presented, and some additional power system models are included. In this case the linear equations are solved using the *bifactorisation* technique. The Jacobian matrix is built in a new fashion

using a new $2 * 2$ sub-matrix algorithm. The routines used to order, factorise and repeatedly solve this problem are described. Also a comparison is given between the new algorithm employed to solve the sparse matrices and the one previously used.

As far as the present author is aware the work currently developed is the first which has implemented the *bifactorisation* technique for dynamic simulation. This method has been applied to load-flow calculation by El-marsatawy et al. [35] and Irving et al. [66]. It has been reported, in a discussion of reference 110, that Brameller and Rudnick also applied this method to a transient stability simulation using an Array Processor.

Chapter 7 describes the decomposed simulator. The *Harwell* library is replaced by the sparsity-directed *bifactorisation* technique. This simulator has been developed to run on a Perkin Elmer PE 3230 minicomputer and is tailored to exploit the shared memory and multi-task characteristics of this machine.

Various tests to highlight the performance of the new algorithm in comparison with the old one, both from the point of view of speed and robustness, are presented in chapter 8. This is supported by sets of graphs representing the most important variables for each of these tests.

Concluding remarks about the contribution of the present work and suggestions for future prospects and trends are presented in chapter 9.

An appendix, which gives details of the differential equations algebraisation and the construction of the Jacobian matrix is also included.

CHAPTER 2

POWER SYSTEM SIMULATION AND OPERATIONAL CONTROL

2.1 INTRODUCTION

Due to technological development the demand for electrical energy has grown over the years. To meet this increasing demand power networks had also to expand. The consumer has become more demanding, for instance, in certain domains of research and industry, a cut in electrical power cannot be tolerated even for a few seconds. This has led to the necessity of developing more efficient control schemes, to ensure a continuous production of electrical energy at a low cost, to improve the security of supply, and to maintain a safe and stable operation of the power system under a wide range of normal and severe operating conditions.

Recent years have also seen the rapid development of computers and their application in various fields. Computer based power system control is a powerful means for assisting the operator in handling power systems operation. Its importance lies in producing fast results so that the behaviour of the system can be detected, and helping the operator in decision-making, especially in emergency conditions where quick intervention is vital. This necessitates efficient and sophisticated software.

System frequency is maintained by the continuous reduction of the mismatch between power generation and load consumption taking into consideration economical constraints and operating limitations. This imbalance is detected by observing the system frequency behaviour. Therefore, eliminating the frequency deviation implies restoration of the balance between the load and power supply.

The continuously varying demand requires continuous action to control the frequency. This can be carried out by a well coordinated system which includes governors assisted by load frequency control (LFC) as a short-term response, economic dispatch as a mid-term action, and unit commitment in the long-term prediction. The generating unit local controllers require varying power set points to meet the new consumer demand and bring back the frequency to the normal value. Load shedding is considered as a last resort alternative to the generation rescheduling if insufficient generation is available.

The short-term action of the local controllers of the generators and load frequency control is effective in the case of small variations of the load. The governor is designed for load control whereas the load frequency control is used to control the system frequency i.e. varying the power set points so that the frequency would be fixed at 50 Hz. The sharp increase and decrease of the load as is the case for the morning and evening peaks, need to be anticipated in the longer term by the economic dispatch and unit commitment so that prior control signals can be sent to the generating unit. This overcomes the limitations on the rate of change of the boiler and generator outputs at a minimum cost.

Generally the governor's action on the inlet turbine valve leads to variation in the boiler pressure. Therefore it is necessary to act upon the boiler quantities such as the fuel, water etc. to bring the steam pressure back to its normal value. However, for fast load changes the steam pressure may be considered as constant [102]. Hence the governor is considered as the primary element in the frequency control loop.

The governor's action is fully explained in chapter 4, and the rest of the secondary control subsystems are included in the present chapter. Figure (2.1) shows typical time scales at which the different power system control functional elements are operating. The different interactions between the control system, and network and generator local controllers are illustrated in figure (2.2).

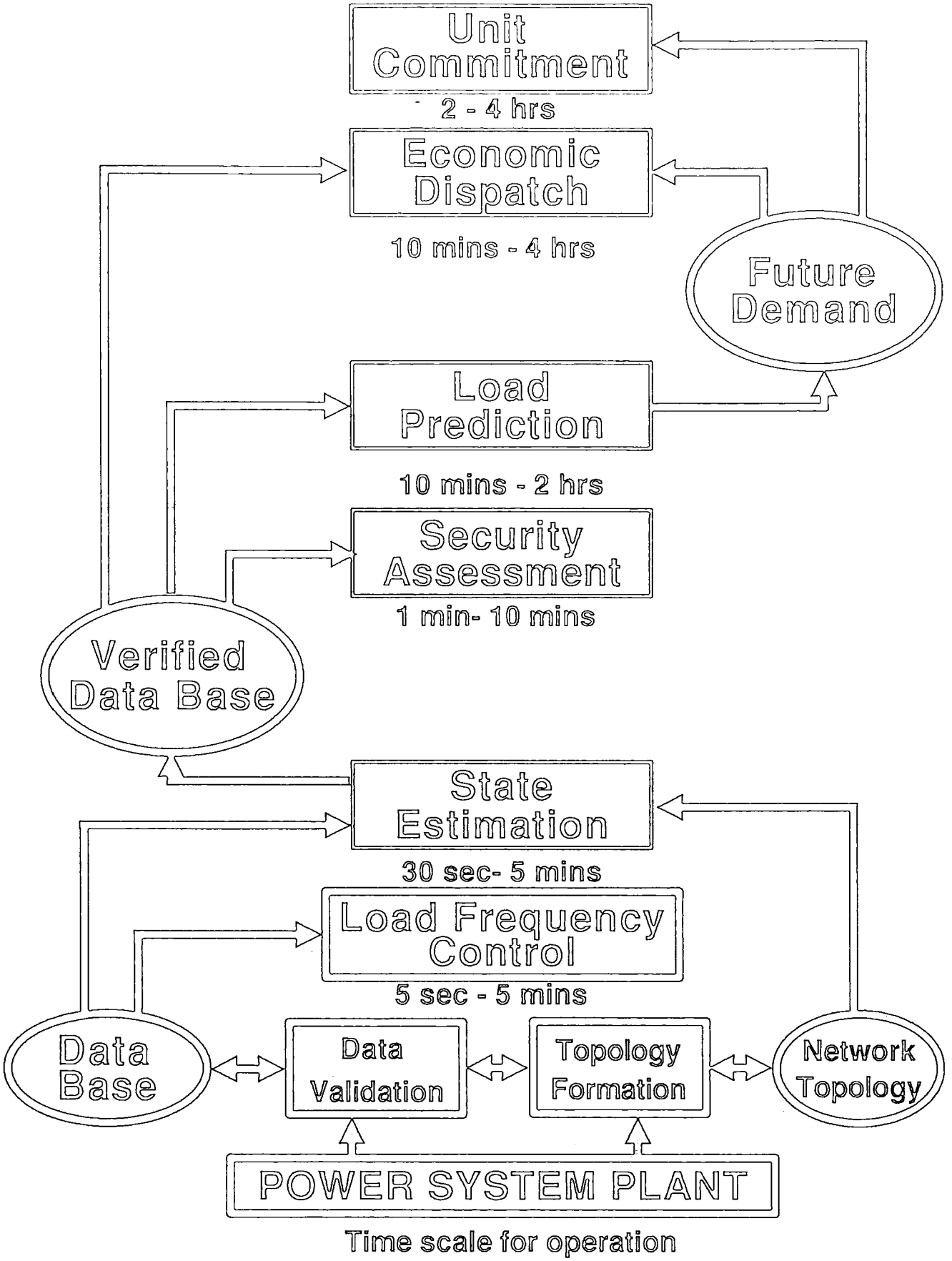


Fig. 2.1 Time scale of power system control

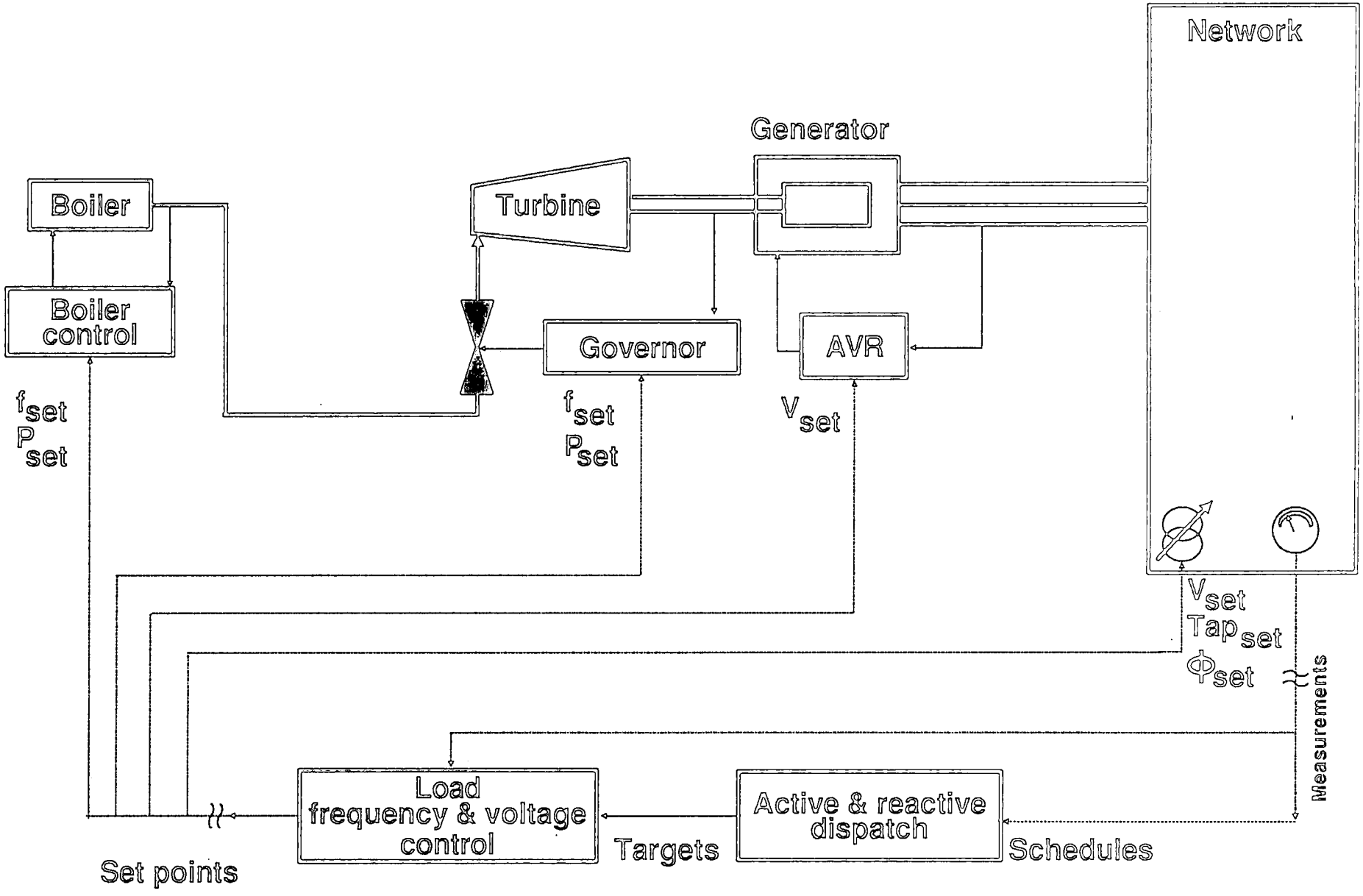


Fig. 2.2 Interaction between a power system and its controllers

Besides power system modelling and the selection of the most suitable numerical techniques, on-line dynamic simulation requires the consideration of such control schemes. This Energy Management Software (EMS) enables the simulator to give a realistic image of power systems behaviour. To alleviate the complexity of power system control and analysis, this has been subdivided into the following functional elements: state estimation, load prediction and monitoring, security analysis, and load and generation control.

The simulator corrupts selected calculated data and sends it to the state estimator where it is filtered. The validated data is then communicated to the database which can be accessed by load monitoring and load prediction, and security analysis. These provide the necessary information for generation and load rescheduling, unit commitment and the economic dispatch. The feedback to the simulator is provided through load frequency control.

The major aim in developing the OCEPS (Operational Control of Electrical Power Systems) software at the University of Durham is its implementation in actual power systems. Therefore, this software should be stable and robust in order to withstand severe events such as islanding, plant outages etc. It also should be fast enough to be able to predict the subsequent problems following such events so that precautions can be taken to avoid the dramatic consequences of these disturbances. To achieve this, adequate computational techniques and hardware are required. The present OCEPS functional elements are quite efficient and are running in real-time i.e. the computational time is equal to or less than the clock time.

The OCEPS control and analysis system has a structure similar to the actual conventional automatic control systems [104]. The available hardware is allocated to the simulator and the control subsystems according to their requirements, to enable the whole system to run in real-time without creating any idle time. The standard version of the simulator is running on an Array Processor FPS 5205 hosted by a Perkin Elmer minicomputer PE 3230. A VAX 8600 minicomputer is assigned to the control and analysis package. The SCADA (Supervisory Control and Data Acquisition) runs on a GEC 4160 computer and

handles the communication between the control package and the simulator. A direct serial link between VAX 8600 and the Perkin Elmer is also available so that the communication can be carried out without having recourse to the SCADA machine. The OCEPS computer network is illustrated in figure (2.3). This figure shows also the recently added hardware comprising Sigmex Args 7000 displays and a VAX 6440 multiprocessor which can be used for parallel processing.

The language used for programming these algorithms is FORTRAN 77. This has been selected for the following reasons:

(1) During the early development of OCEPS there was no language available or sufficiently well developed to compete with FORTRAN.
(2) The wide use of FORTRAN all over the world and the availability of FORTRAN compilers for most hardware. This is very useful for transporting OCEPS easily to different machines.
(3) FORTRAN is designed mainly for numerical calculations. This is the major task of the OCEPS project.

Although more sophisticated and user friendly languages, which are also portable and even more efficient in handling numerical calculations than FORTRAN, have now been developed; translating all the OCEPS package into these modern languages may be too expensive and time consuming.

In addition to the above mentioned requirements, both the numerical dynamic simulator and the control package require a test network together with its physical data, the representation or simulation of the load variation, application of disturbances and outages and the updating of the topology of the network as shown in figure (2.4).

2.2 SIMULATION

The fundamental part of a controlled power system is the physical network. This is represented by a comprehensive dynamic model which includes

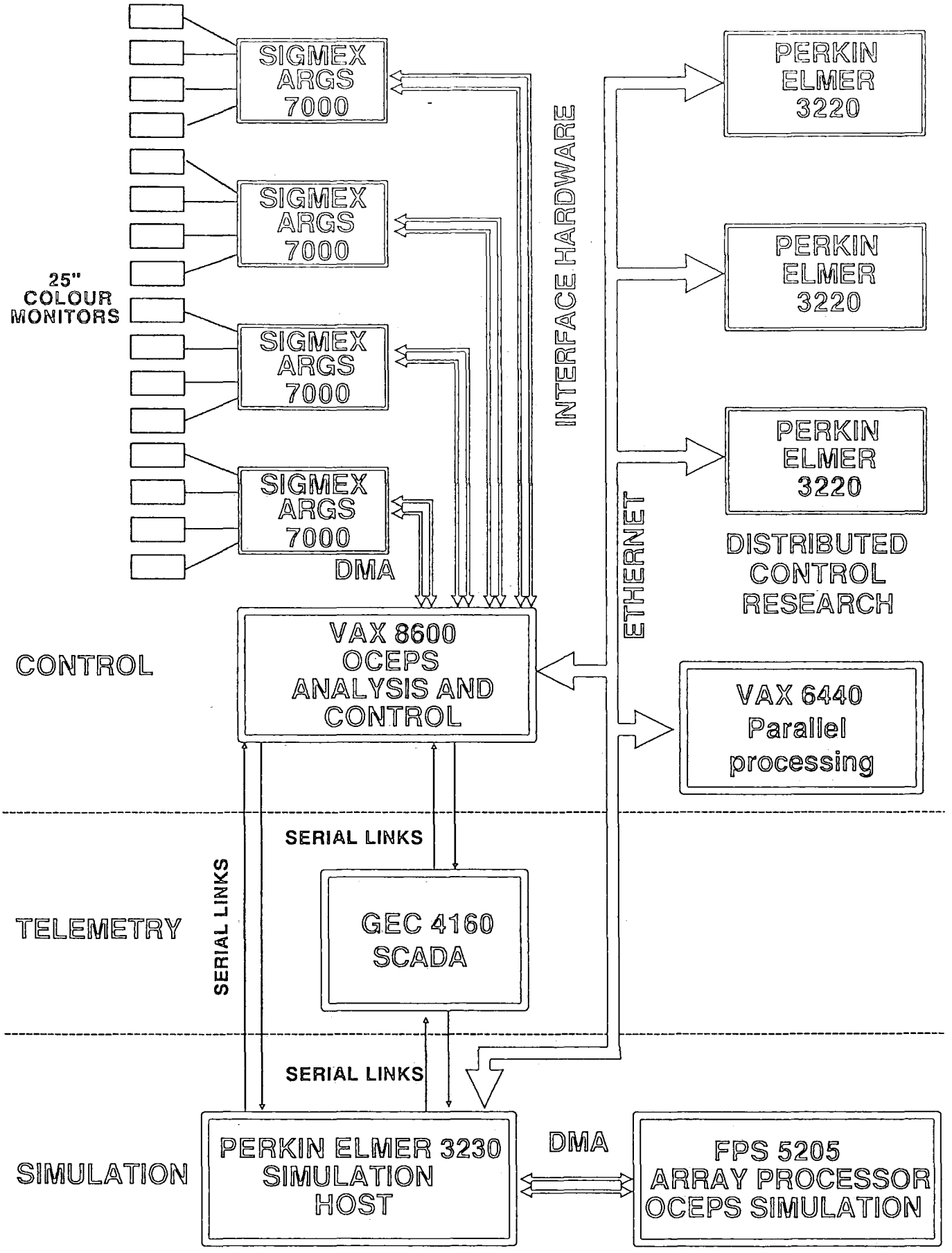


Fig. 2.3 OCEPS Computer network

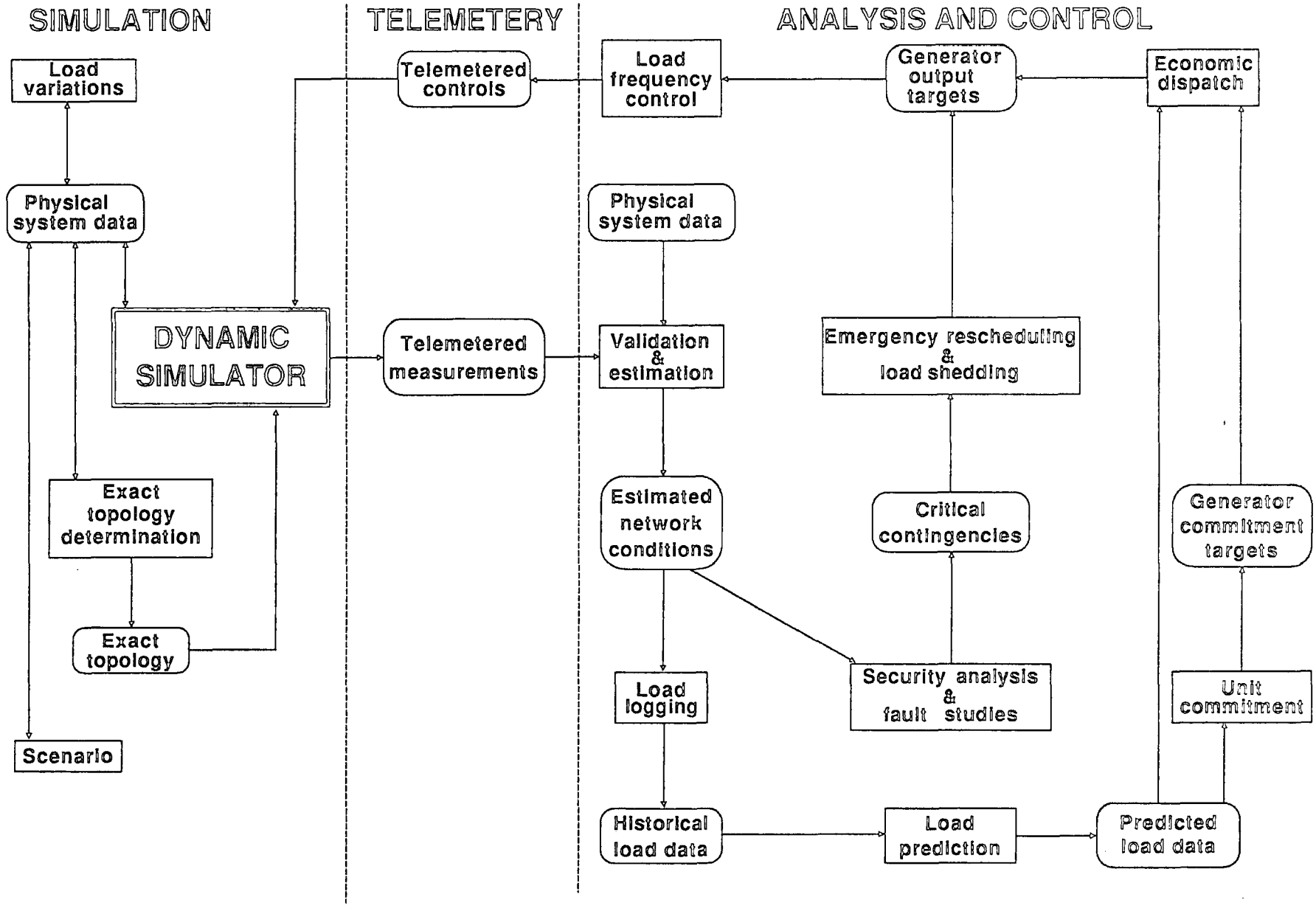


Fig. 2.4 Interaction simulator/(control & analysis) package

most of the actual elements of a power system. As the network is the main part of an electrical power system, similarly, the simulator is also essential for testing the developed Energy Management Software (EMS) without having recourse to a real system.

Power systems integrity, depends primarily on the network control of the voltage, frequency, transmission switching etc. and consequently the simulator must be able to replicate these features.

Since the simulator is the first subsystem to be run, it must be designed in such a way that it can run independently of the control and analysis subsystems as a stand-alone package. The initial set points can be provided by local tasks. The frequency set point in this case can be set to 50 Hz, and the transformer's tap-changing and phase-shifting initial step position can be determined in the control initialisation task. The power set points and the plant merit order can be evaluated by the load flow task. Some variables such as the generators and transformers voltage set points can be evaluated within the simulator.

2.2.1 Simulator function

The simulator modelling is fully described in chapter 4. Figure (2.5) shows the relationship between the simulator, its supporting tasks and local common blocks.

2.2.2 Simulated network

Both simulation and control and analysis systems require realistic test networks which can be either existing systems or hypothetical built to suit research purposes.

The standard OCEPS test network is based on the standard IEEE 30 node network [40]. This is shown in figure (2.6).

This system consists of groups of substations linked via transmission lines to make an island and groups of busbars connected by links within a substation

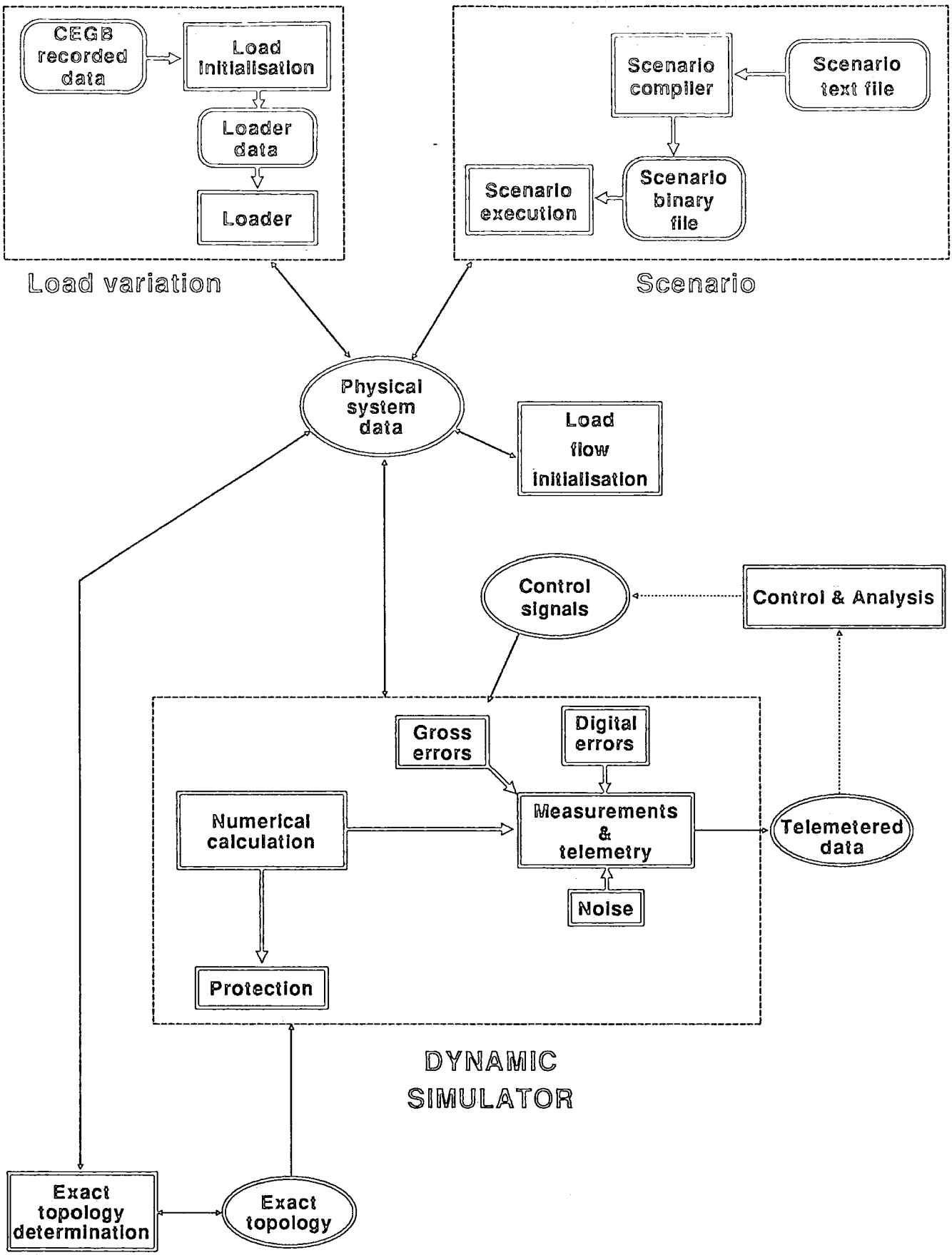


Fig. 2.5 The simulator and its supporting packages

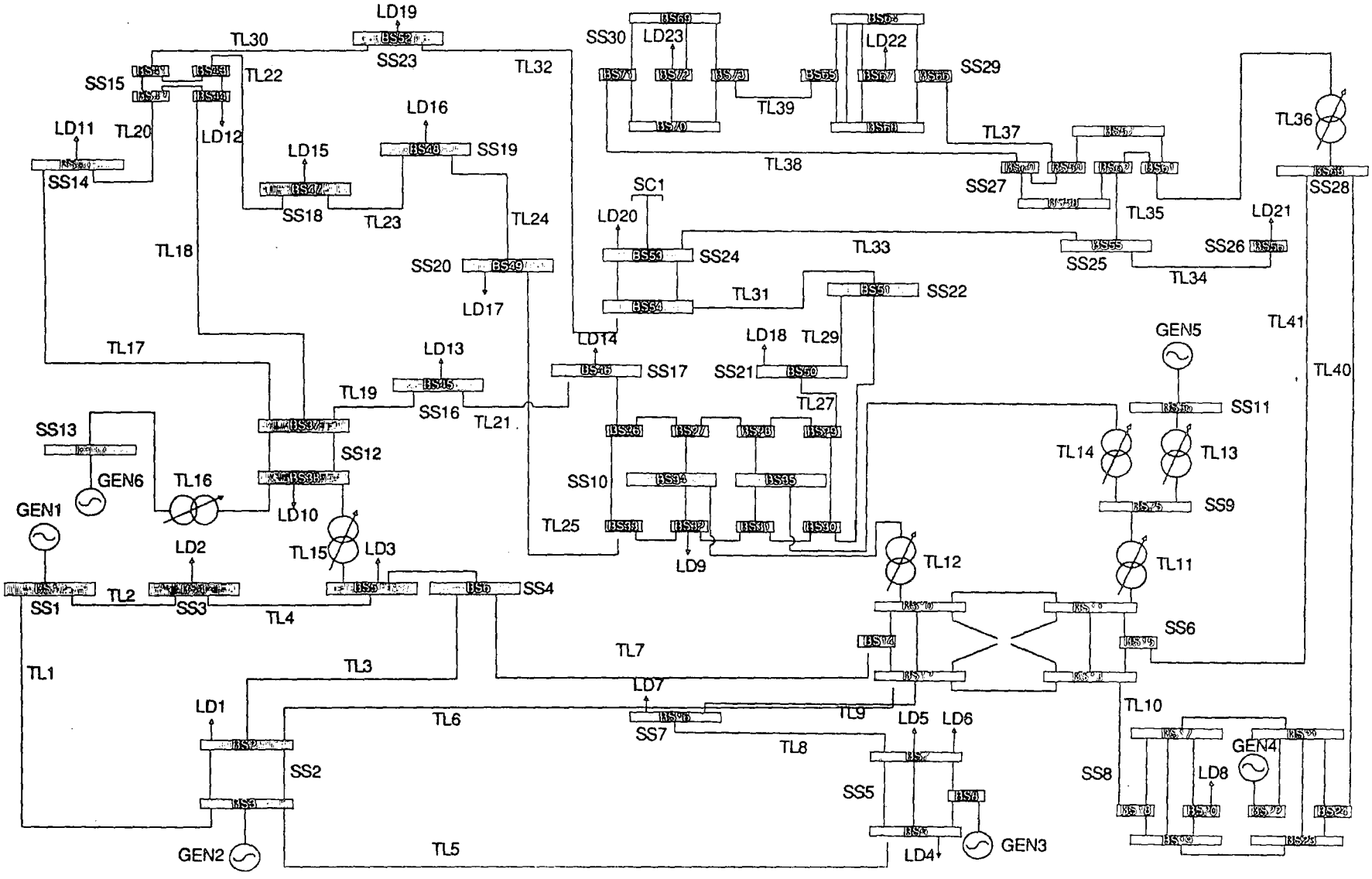


Fig. 2.6 30 node test network

to make a node. The configuration of the substations have been modified to allow for more complex tests to be carried out [7]. The number of generators in this test network is 6, and these deliver a total maximum power of 570 MW. The largest generator has a maximum capacity of 200 MW whereas the smallest ones have a capacity of 50 MW. 41 transmission lines, 4 transformers, 23 loads, and 2 static compensators are included. The physical network data such as switch states, plant ratings and parameters etc. are included in a common block which can be accessed by the simulator and all its supporting subsystems

Some of the physical data are dynamic. They are altered by the consumer load variation, topology changes etc. Others remain static throughout the simulation period.

2.2.3 Exact topology determination

In order to account for changes in the network resulting from deliberate actions and protection tripping, and to update its connectivity, a topology determination program is necessary. To preserve the integrity of the simulator/control system and the validity of the analysed system both subsystems have to consider the same altered network. A flag can be used to inform the topology task about the state of the network. If a change in any of the switches occurs, the topology program should be allowed to access the physical data. According to the new switch and circuit breaker states, the energised elements should be considered and the disconnected ones should be dismissed. To avoid the analysis of an invalid network structure, this function should be carried out as soon as any change is detected.

The topology determination consists of scanning through a list of busbars, and allocating them to their corresponding nodes and islands. A node is defined as a group of one or more busbars connected through energised links. An island is defined as a group of one or more active nodes linked by energised lines or transformers. If no generators are available within an island it is considered as a dead or passive island. An active node is a node which is not isolated from the rest of the network.

The control package also requires a topology validation. This is because of the possible corruption of switch states during the telemetry process. The validation of the network connectivity is therefore necessary before starting any task, otherwise the whole study will be erroneous.

2.2.4 Consumer load variation

The simulation of the consumer load variation requires recorded past data. The system described in this thesis uses the southwest region of the CEGB grid recorded load data. This data has been scaled to suit the test network generation capabilities. The processed data is spread over a period of about six weeks starting from midnight of the 8th of February 1985. This task has been designed in such a way that the initial time can be chosen according to the requirements of the test run. For instance, a heavy load can be selected to test the behaviour of the system under such conditions. The seasonal daily loads show a regular pattern, with a morning and evening peaks as shown in figure (2.7).

The global load provided is subdivided by the loader algorithm among the different loads throughout the simulated system. A fixed percentage is assigned to each of them. This is acceptable since each load is simulated as an aggregated one, not as individual loads such as industrial load, domestic load etc.

2.2.5 Load-flow calculation

For simulation purposes the load-flow algorithm should run only once for initialising certain variables. Load-flow is also used for security analysis. In this case Solution efficiency is important since power flow is fundamental for contingency evaluation and is performed frequently.

The load-flow provides the present simulator with quite accurate initial voltages and phase angles necessary for the Newton-Raphson solution. It also calculates the active and reactive power flow. A simple merit order subroutine is included in this supporting package. It evaluates the initial power set points

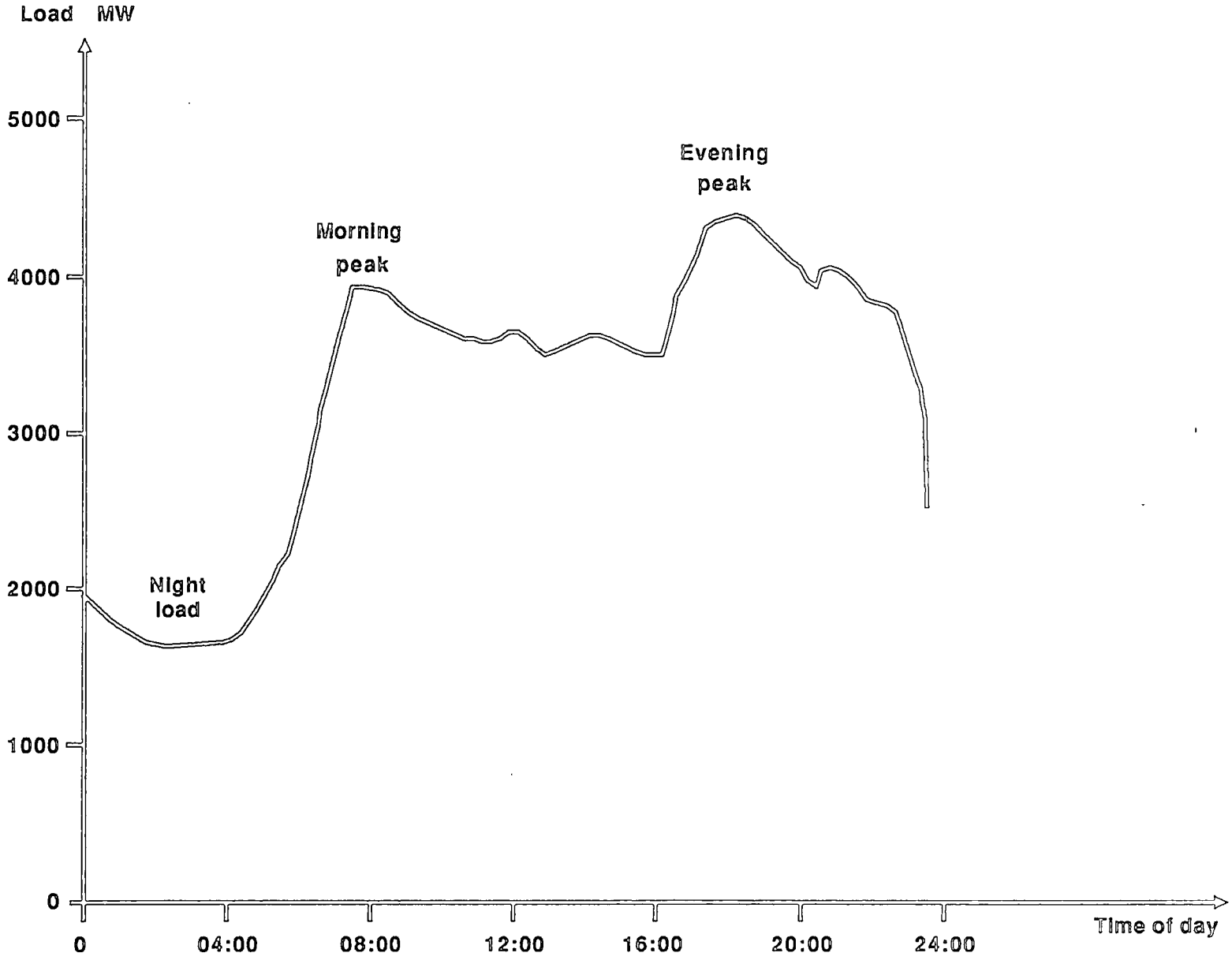


Fig. 2.7 Variation of a typical daily load pattern

for each generator so that a generation level could be attributed to every unit according to its capability.

The used algorithm is based on the Newton-Raphson iterative method. Many techniques have been tried to speed-up this package processing time. Among which are the development of efficient decomposed methods and the implementation of the *bifactorisation* technique [66,91].

2.2.6 Control signals

Control inputs to the simulator include voltage targets, phase-shifting and tap-changing, and frequency and power set points. These are provided by load voltage and frequency control respectively. Active and reactive power targets are set by active and reactive dispatch respectively. The former is fully developed and included in the present version of OCEPS, whereas the latter is actually under development.

Set points are updated periodically by the load frequency and voltage control at a typical time step of 5 seconds. During this period the simulator considers them as constants. Whenever a serious event such as a sharp increase or decrease of the load, load shedding or generator start-up or shut-down takes place, frequency fluctuations occur. These are initially supported by the inertia of the rotating mass of the generators. Then the governor and the rest of the generator local controllers react as well as the load frequency and voltage control which provides the frequency and power set points.

2.2.7 Scenario generation

A scenario can represent a set of actual events recorded by a utility for validating the behaviour of the simulator [68], or a set of hypothetical events for testing the effect of severe disturbances on the stability of the simulated system [68,89,93,103,104], or generated automatically by a program included in the package [109,115]. In this last case a transient stability program is run prior to the simulator. If the transient stability program is stable, the resulting events such as protection tripping, and the corresponding time are stored for subsequent use by the simulator.

The first two strategies are adopted within the present project. A scenario representing the 1981 incident in the south of England area [68] has been reconstructed and used to validate the present simulator. Generally, more severe events than those occurring in real life disturbances are needed to check the robustness of the developed algorithms.

Two ways of implementing scenarios in the system are available: manual interaction by feeding in specific commands, or gathering these instructions in a file with the desired times of operation.

The available scenarios can include opening and closing circuit breakers and switches, re-synchronisation of the split network for re-connection, and the introduction of gross errors and equipment failure. Scenarios are especially useful for training where the instructor can set a sequence of events and the trainee has the task of clearing up the disturbance and restoring the normal operating conditions of the simulated system. They are also very useful for testing the robustness and efficacy of the developed energy management software.

Following these disturbances a series of events are created. The important ones are stored by the event logging subsystem. The resulting topology changes and calculated quantities are displayed on graphical terminals. Audible alarms are also used to attract the operator's attention for intervention.

2.3 CONTROL AND ANALYSIS FUNCTIONS

This package should also be stable under severe conditions so that restoring normal conditions will be fast and not aggravated by control system inadequacy. For instance, in the case of a split network, the various software modules included in power system control should be able to treat the different islands as independent sub-networks. In addition, the selected algorithms and numerical methods should be robust and stable so that physical instability would be valid and the software results would be realistic.

2.3.1 The SCADA system

Prior to the late 1960s the only real-time data available in the control centres were the circuit breakers status, the system frequency and real power measurements [30]. Drastic power system problems such as blackouts emphasised the need to develop these centres in order to control and monitor efficiently the power system and ensure the security of supply. This has led to the creation of the SCADA systems. These consist of gathering periodically as much data as possible from the whole system. The acquired data can then be used for supervising the operating conditions of the system and the detection of abnormal situations. In this case alarms are initiated and the events are handled by loggers and displayed on graphical screens. This data is usually corrupt because of many factors such as the transducers inefficiency. Furthermore, some of the necessary data can be unavailable. This has led to the adoption of state estimators (described in the following subsection) to filter the raw data acquired by the SCADA system, and estimate the unmeasured values.

The SCADA system operated by OCEPS group ensures a similar function. It acquires the measured data telemetred by the simulator and sends it to the control package through the state estimator and data validation. This system is subdivided into various functional subsystems such as data acquisition, database management, supervisory control, man/machine interface, alarms, event logs and result display etc.

A wide spectrum of power system events and phenomena together with the corresponding data are processed and displayed on graphic terminals. The different alarms generated during any disturbance are also displayed on these screens to show the network state. These interactive graphical programs are designed so that the operator can assess these alarms and try to restore the normal operating conditions. Event logs are used to select only the most important and relevant alarms and events. This man/machine interface algorithm is a powerful tool especially for training and teaching purposes.

Reference 48 used the SCADA system to remotely control the capacitors following the loading level. This flexible process decides every 15 minutes

whether or not a capacitor is needed. Compared to the manual operation of capacitors this scheme is more economical and improves the voltage profile and the power factor on the distribution level. It also reduces the transmitted reactive flow.

2.3.2 State estimation, observability and data validation

Schweppe [30] was the first to recognise the necessity for a state estimator to overcome the inconsistency of the database gathered by the SCADA.

Therefore, it can be said that the success of automatic computer assisted power system control relies mainly on the validity and availability of such necessary information about the state of the network at any desired time. The data accumulated by transducers is often subject to corruption because of the noise effect during its transfer, and the accuracy and reliability of the transducers. These instruments can also give very big errors in the case of their failure. Filtering these measured quantities and providing good estimates of the quantities which have not been measured is the main task of the state estimator. It plays the double role of building a consistent database and being an on-line load-flow which can be used by the security analysis package.

The set of measurements and their location necessary for state estimation is dynamic; it varies with the topology, the state of transducers, and telemetry. If the available measurements are not sufficient in number and location the system is said to be unobservable. The observability function is to detect this problem and provide the additional pseudo-measurements to the measurement set [5,30].

Validation of the topology is also required since erroneous switch states may be recorded in certain cases.

The validated data and topology are then stored in a database and common blocks which can be accessed by the different subsystems of the control and analysis scheme.

A further benefit of state estimation is the minimisation of the metering points which can lead to a reduction in the capital invested in constructing the measurement system and its maintenance [102].

The state estimator algorithm should be robust and efficient so that the variables in the database can be updated quickly. This would facilitate data acquisition and increase the rate of accessing the database so that full potentiality of the control algorithms can be exploited.

The adoption of state estimation and its use as an on-line load-flow system can be considered as the impetus to the design of modern power system control, augmenting the SCADA with EMS facilities.

2.3.3 Load monitoring and prediction

The load monitoring function is a simulation of data recording of the system load. The state estimator provides the individual validated measured and estimated loads through the database. Based on this data the load monitoring then evaluates the global load of the system. This recorded global data is then used as a historical load by the load prediction task to estimate future consumer demand.

Although predicting the exact consumer load is impossible, a quite regular seasonal daily pattern, as shown in figure (2.7), enables an approximate estimation of the future load. The present standard OCEPS load prediction initially uses the past data provided by the CEGB. This data is then appended by the load monitoring output as the look ahead time is advanced. The lead time of load prediction varies between 10 minutes to 2 hours as shown in figure (2.1).

The sudden peaks are detected in advance by load prediction, and the necessary signals are sent to the generating units to provide the amount of power necessary to meet these requirements smoothly. This prevents system imbalance and the subsequent undesirable events such as large excursion of the frequency and sharp variation of the generator power outputs that can

cause over-speed and overload protection actions. Furthermore, since the rate of change of generation output is limited by the delays and slow time constants of the generating units, a large unexpected variation of the load may lead to load shedding.

The output of load prediction is fed to unit commitment and economic dispatch to predetermine the necessary generation and the target time at which this amount of power output should be available to meet the detected variation of the consumer demand at the lowest expense. Therefore, the incorporation of load prediction does not only ensure safe operating conditions, but also an economic generator schedule and sufficient spinning reserve.

2.3.4 Generation and load control

The objective of this functional element of power system control is mainly the determination of the most economical way of operating the generating units taking into account the future load and the different operational constraints and limitations.

The importance of assuring a continuous security of supply and the stability of the power system over its economical management has led to the adoption of a subdivision of the generation and load control into network control and operational economics. The network control includes the plant and network local controllers, load frequency control, generation rescheduling, and load shedding. These schemes are the primary controllers since considering the economical aspect in operating an unstable and unreliable system is meaningless. However, once the stability and reliability of the system are established huge benefits can be realised by economically operating the network. The economic operation necessitates the consideration of unit commitment and economic dispatch.

These tasks are closely linked. The data flow between these functions is shown in figure (2.8). The load frequency controller is dependent on the economic dispatch output and the latter depends on the unit commitment output while these two tasks depend on load prediction. These schemes run

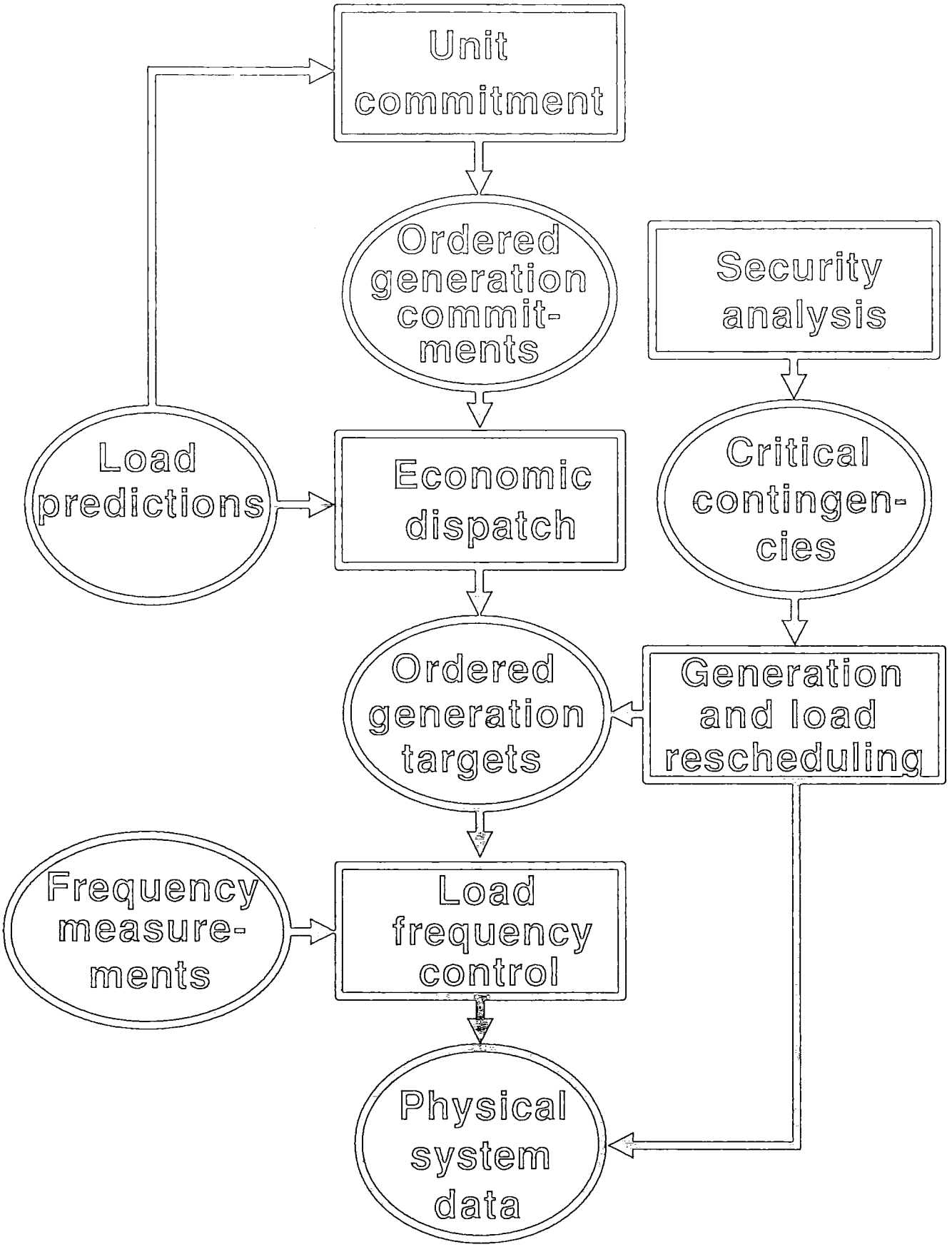


Fig. 2.8 Generation and load control

independently in an asynchronous way assuming that their input is constant until the keys and flags are reset to warn them of any relevant changes and data updating. Their initial data is provided through data files or assessed locally. Most of these control functions enter into a waiting mode until the new information is provided.

2.3.4.1 Unit commitment

Unit commitment can be automatic or run manually by the operator at selected intervals. It consists of ordering in an optimum economical sequence the start-up and shut-down of the available generators. It also indicates the most economical spinning reserve to enable the system to respond quickly to unexpected loss of generation and load increase, and maintain secure operation. The spare capacity is optimally divided among all the running generators so that in the case of a generator loss the remaining spare capacity will be able to provide the lost power and stabilise the system generation/load.

The selection of the plant order and spinning reserve depends mainly on economic factors, future load, and operational constraints and limitations.

Usually the strategy adopted in plant ordering is based on loading the cheapest plants to their maximum output and reserving the more expensive ones for peak hours.

2.3.4.2 Economic dispatch

The function of the economic dispatch is the determination of a power target for all the operating generators designated by unit commitment. It also sets a target time at which the preset power target should be reached. The allocation of power targets to the different generators is based on providing the required consumer demand at the minimum operating cost. However, in emergency conditions the security of supply has a higher priority than economical considerations. In these cases rapid generation rescheduling and/or load shedding are unavoidable.

At steady state economic dispatch operates at a period of about 10 minutes. However, the calculation can be interrupted and a new time interval can be restarted in the case of topology changes, or unexpected disturbances. The target times and power outputs are fed to the load frequency control to receive further processing.

2.3.4.3 Load frequency control

The load frequency control task has the function of ramping the power output towards the target points set by the economic dispatch within the target time. This provides the simulator with power and frequency set points, and ensures a smooth variation of generator outputs in the case of system frequency change or tie line power interchange variation caused mainly by sudden imbalance between the load and generation.

New set points are evaluated periodically (typically every 5 seconds [7]) and communicated as control signals to the simulator by a communication program or through the SCADA machine via serial links. These values should be stored in databases. Access to these variables must be synchronised by local flags set independently by each program. Thus allowing only the currently privileged task to read from or write to the database.

In the case of sudden failure of generation resources to meet the system load, a rescheduling function is necessary for rapid control of the situation. This provides a temporary power and time target for the load frequency action rather than waiting for the relatively slow economic dispatch response.

2.3.4.4 Generation rescheduling and load shedding

Generation and load rescheduling is both an automatic and manual function. It is activated automatically whenever an under-frequency or a large energy deficit occurs in the system, and can also be manually initiated by the operator.

As mentioned above this function provides a quick response to heavily loaded networks caused by unexpected generation losses or load increase. In this

case the economical constraints become secondary and a trade-off between load shedding and generation rescheduling is necessary. Although the load shedding is usually used as a last resort.

Optimum load shedding requires a careful selection of the location, the amount and the sequence of the loads to be shed [59]. If a preset under-frequency or a critical load generation imbalance is reached a selected amount of a given load has to be disconnected to minimise disruption and avoid the propagation of the disturbance over the system. The selection of the load to shed is therefore automatic whereas the execution of the actual load shedding requires the confirmation of the operator.

2.3.5 Security analysis

The necessity for generation and load rescheduling is detected by the security analysis. This is achieved by performing a fast load-flow calculation to predict any disturbance subsequent to any line or generation disconnection.

In order to detect these disturbances early enough this algorithm should be very fast to allow for rapid control actions before the predicted problem occurs.

In this chapter the environment of the simulator and its interaction with the EMS has been briefly described. The following chapter will be concerned with the numerical methods used in power system dynamic simulation and the selection of the most suitable techniques and numerical conditions to realise a fast simulator.

CHAPTER 3

NUMERICAL ANALYSIS FOR POWER SYSTEM SIMULATION

3.1 INTRODUCTION

The present day digital computing resources are overburdened by the amount of calculation necessary for power system dynamic simulation. In a large network thousands of dynamic-algebraic equations require an expensive step-by-step solution. The expansion of the interconnected systems, and the requirement for more detailed models with frequent solutions have added to the complexity of the problem and the difficulty of realising real-time simulators, although the hardware has improved significantly. To achieve this, more effort has been devoted to developing and improving the application of the numerical methods and computational techniques such as sparsity-directed approaches and network decomposition to power system dynamic studies. Thus the software is as important as the hardware.

Solution techniques introduce some errors which may affect the accuracy of the response of the system and its stability. These errors consist of truncation errors reflecting the inexactness of the chosen integration formula, and also round-off errors which are introduced by the non exact convergence of the iterative solutions, and the computational arithmetic inaccuracy. The latter are related to the length of words for floating-point operations and the size of the problem. 60- and 48-bit machines are reliable, whereas in the case of smaller words the main solution variables must be stored with high precision. Other errors are due to the approximations and simplifications in modelling the power system elements. It is very important that these errors are not propagated

from one time interval to the other during the overall solution of the network. The severity of this problem is directly related to the chosen methods.

The stability of the system depends on these accumulated errors (global error) and the stiffness of the simulated system. A problem is considered as stiff if the ratio of its largest to its lowest time constant is large. This can cause instability in the case of important disturbances if variations in the stiff variables are affecting the main variables. The stiffness of a system can be alleviated by neglecting the short time constants and simplifying the models of the dynamic components. However, this is strictly dependent on the nature of the study carried out. The truncation error can be reduced by choosing a high order integration formula. However, for certain methods this is restricted [46]. The accuracy available by using iterative methods is increased by using higher precision and smaller tolerance. The choice of small integration time steps can keep the errors within an acceptable level for all solution methods. However, these time steps should not be less than a critical value which would increase round-off errors [46,80].

Apart from the first suggestion of neglecting the short time constants, all these proposals are computationally very expensive and conflict with the requirement of a fast solution (real-time simulation). Consequently, the use of stable integration methods, which can tolerate large errors and time steps with a relatively accurate response, are the most adequate and economical solution to numerical stability problems.

Also the way the algebraic and dynamic equations are interfaced may be of great importance. These two sets of equations can either be lumped together and solved simultaneously as one set of equations, or treated sequentially with a transfer of the necessary common variables between the two subprograms dealing with the dynamic and the network equations.

For large problems where the selection of fast integration and iteration methods as well as sparsity programming are not sufficient for real-time simulation, additional sophisticated means such as network decomposition are

necessary. This technique can also be advantageous in terms of storage requirements. The decomposed technique consists of subdividing the large systems into subsystems which can be solved individually in parallel. Each one of these subsystems can be run in a separate machine called a slave. The overall solution is obtained by a master machine which receives the interfacing common variables and messages from the slaves.

3.2 PARTITIONED METHOD

In general the partitioned method can be defined as a technique where the problem to be solved is subdivided into separate sub-problems which can be solved by different methods. These sub-problems can be presented in the form of subroutines solved by a single computer in the case of centralised simulators or divided between different processors if a decomposed technique is adopted.

One of the advantages of the partitioned method is its ability to solve the differential and algebraic equations by different methods. In addition, this traditional approach can be of a high performance because the network equations may be solved once for each time step, or even once after a few time steps in the case of mid- and long-term stability studies [106].

However, this approximation is the source of an important error called interfacing error. This error can be reduced but not eliminated and if the chosen integration method is not very stable, it may cause the failure of the solution of the system [106].

Generally, this approach uses explicit integration methods for solving the dynamic set of equations.

The partitioned solution approach can also be applied to differential equations [106]. These can be split into sets representing different dynamic elements or groups of elements. The dynamic component partitioning can be based on the range of the time constants. This technique may be suitable for stiff problems, where different time steps can be used. Long time steps can be

chosen for the network and non stiff elements, and small time steps can be used for stiff components. Furthermore, these blocks of differential equations may be solved by different integration methods. Stable methods can be implemented for stiff problems and simple methods can be used for non stiff components. This strategy has been adopted by Gross et al. [49]. The availability of very stable methods, which are stable even for stiff systems, provides a better alternative to this approach even if it is showing a less accurate response, since partitioning of differential equations weakens the stability of the system as a whole and shows falsely stable solutions under certain conditions [106].

Based on the idea of using different methods and time steps for simulation, Johnson et al. [72] have used a tuneable first order approach. The values of the tuning parameter are obtained analytically. They depend on the time constants and time steps. For certain specific values of this parameter this approach takes the form of some of the well-known integration methods: the explicit Euler method, the implicit backward Euler, or the implicit trapezoidal rule. The advantage of this approach is the reduction of the global error. This significantly helps the accuracy of the response. The drawback of this technique is the possible instability which can be caused by the explicit Euler method [72]. Gear has suggested an automatic control of the step size and order to reduce the error [46].

3.3 SIMULTANEOUS METHOD

This method started becoming popular during the last two decades. Most of the research efforts before then were concentrated on the short-term stability studies and little was done on mid- and long-term simulation. Transient stability problems are generally solved by the partitioned approach [106]. The simultaneous method consists of algebraising differential equations and solving them simultaneously with the network models and with the same frequency, thus eliminating the interfacing error. Generally the integration methods used for the algebraisation of the dynamic equations are implicit [106].

3.4 INTEGRATION METHODS

Many of the differential equations describing dynamic physical systems cannot be determined analytically [101]. Approximate solutions are obtainable by using numerical techniques. A lot of these integration methods are available in the literature, but few of them are suitable for practical problems such as power system simulation. They are categorised as explicit and implicit integration methods. These methods can be either multi-step or single step. The single step methods have the advantage of being self-starting. The choice of a particular approach depends mainly on the requirements of the solution and its convenience. In certain cases explicit and implicit methods can be combined to solve a problem. This can be applied to the partitioned differential equations, and the predictor-corrector methods where the predictor is solved explicitly, and the corrector is solved implicitly [4].

3.4.1 Explicit methods

As their name indicates, these methods evaluate explicitly and individually the different differential equations. In order to evaluate a present step variable most of the explicit methods need one or more past steps values as well as the derivatives of that variable, depending whether they are single-step or multi-step approaches. This requires the storage of the required quantities over a few time intervals, which is not desirable especially when the computer memory is limited.

These approaches are not stable and their time steps are restricted by the smallest time constant of the system to be solved [24]. Furthermore, explicit methods have a greater requirement for error control than implicit methods, which can tolerate and recover from temporary periods of high-error generation during the course of a solution [106]. This can be reasonably well controlled by using small time steps and high order integration formula. A compromise between the accuracy and the computing efficiency must be considered.

3.4.2 Implicit methods

Compared to the explicit methods, implicit techniques have a complex

program structure. They involve a lot of effort for programming the algebraised differential equations which have to be solved simultaneously. Since the corresponding program is used routinely this disadvantage is not significant as long as the algorithm is flexible and easy to maintain. It is well known that implicit methods are several times faster than their equivalent explicit techniques [24,106]. They are solved iteratively, either together with the static set of equations as in the case of the simultaneous methods, or separately as in the case of the partitioned approach.

3.5 SELECTION OF THE NUMERICAL METHODS

The main factors to be considered when choosing a simulation solution technique are speed, accuracy, stability and flexibility. These depend more or less on the purpose and requirements of the developed study. For instance, a moderate accuracy is tolerated by mid- and long-term simulation, to alleviate the constraints on the speed especially for large systems. The storage required is not a problem as long as this does not affect the efficiency and reliability of the real-time simulation.

Numerical stability is another constraint in solving power system problems. There are many ways of approaching this serious problem as mentioned above. The most efficient way of overcoming this difficulty is to choose a stable, and relatively accurate method which would remain stable under different operating conditions and would cope with a wide range of time constants and constraints without sacrificing the computational time. A wide variety of methods are available, and many investigations have been carried out over the years to find out the most suitable techniques for solving stiff ordinary differential equations. The most convenient approach is the implicit trapezoidal method [2,24,55,73,106]. This is the case particularly for long-term dynamic studies of large power systems where a large step size is needed to achieve a real-time simulation.

Non-linear algebraic equations are usually better solved by the Newton-Raphson iterative method [4,101]. The solution of the differential equations require algebraisation by the implicit methods (trapezoidal rule) and an iterative

solution. This implies that the use of the simultaneous technique is suitable for this problem since both the dynamic and static equations require iterative solutions. The most important advantage of the combination of differential and algebraic equations is the automatic elimination of the interface error [73] which offers a better accuracy.

3.5.1 The implicit trapezoidal rule

The trapezoidal rule adopted here is a second order self-starting method. It is stable under a wide range of step sizes. Therefore, the choice of a 1 second time step is better for slow transients and steady state as this helps in speeding-up the simulator. The step by step integration methods are based on dividing the simulation interval into n discrete time segments. The resulting algebraised equation of the set of differential equations $[\dot{Y}]$ is given by equation (3.1).

$$[\dot{Y}]_{(t+\Delta t)} = [Y]_t + \frac{\Delta t}{2} * ([\dot{Y}]_{(t+\Delta t)} + [\dot{Y}]_t) \tag{3.1}$$

This can be rewritten as

$$[h([Y]^t, [X]^t)] = 0 \tag{3.2}$$

Where

$[h([Y]^t, [X]^t)]$ is the set of algebraised equations,
[X] is the network state variable vector,
[Y] is the dynamic state variable vector, and
t denotes the transpose of a matrix or a vector.

3.5.2 Newton-Raphson method

The Newton-Raphson method (N-R) is a widely used technique in the field of transient stability studies and load-flow solutions [2,4,24,66,101,106]. This is mainly due to the following characteristic: independently of the network size convergence can be obtained after a few iterations, usually from 2 to 5 without any acceleration coefficients. This is due to its quadratic rate of convergence (i.e. approaching convergence the error at each iteration is the square of the error at the previous iteration [9]).

It is mentioned in reference 4 that compared to the Gauss-Seidel method, the solution of a 500 node network using N-R method is 15 times faster. This

is because of the slow convergence of the Gauss-Seidel method. Therefore, this method is not recommended [11] because in certain cases it can require an excessive number of iterations. However, it is still used in some cases for transient stability studies [73]. The N-R method is therefore very attractive for large systems.

The N-R method is based on the Taylor series. Although only the first order terms are included as shown in equation (3.4), the N-R method is considered as an accurate method because of its rate of convergence.

The principle of the N-R method to solve the subset of algebraised differential equations $[h([Y]^t,[X]^t)]$ and the non-linear algebraic equations $[g([Y]^t,[X]^t)]$ for n variables is shown below.

$$\begin{aligned} [h([Y]^t,[X]^t)] &= 0 \qquad (a) \\ [g([Y]^t,[X]^t)] &= 0 \qquad (b) \end{aligned} \tag{3.3}$$

$$\begin{aligned} [h([Y]^t,[X]^t)]_{k+1} &= [h([Y]^t,[X]^t)]_k + [\tfrac{d[h([Y]^t,[X]^t)]}{d[Y]}]_k * [\Delta Y]_k + \\ &\quad [\tfrac{d[h([Y]^t,[X]^t)]}{d[X]}]_k * [\Delta X]_k + \epsilon_1 \qquad (a) \\ [g([Y]^t,[X]^t)]_{k+1} &= [g([Y]^t,[X]^t)]_k + [\tfrac{d[g([Y]^t,[X]^t)]}{d[Y]}]_k * [\Delta Y]_k + \\ &\quad [\tfrac{d[g([Y]^t,[X]^t)]}{d[X]}]_k * [\Delta X]_k + \epsilon_2 \qquad (b) \end{aligned} \tag{3.4}$$

Where ϵ_i is the error.

Neglecting the error, and supposing that equation (3.4) is satisfied at iteration $k+1$, this expression can be rearranged in a matrix form as follows

$$-[J]_k \times \begin{bmatrix} [\Delta Y] \\ [\Delta X] \end{bmatrix}_k = \begin{bmatrix} [h([Y]^t,[X]^t)] \\ [g([Y]^t,[X]^t)] \end{bmatrix}_k \tag{3.5}$$

Where

the square matrix $[J]$ is the $n * n$ Jacobian matrix,
$[h([Y]^t,[X]^t)]$ is the algebraised dynamic subset of equations,
$[g([Y]^t,[X]^t)]$ is the subset of the algebraic equations, and
k is the iteration number.

The Jacobian matrix is derived from equation (3.4) and given by

$$[J] = \begin{bmatrix} \left[\frac{d[h([Y]^t,[X]^t)]}{d[Y]}\right] & \left[\frac{d[h([Y]^t,[X]^t)]}{d[X]}\right] \\ \left[\frac{d[g([Y]^t,[X]^t)]}{d[Y]}\right] & \left[\frac{d[g([Y]^t,[X]^t)]}{d[X]}\right] \end{bmatrix} \tag{3.6}$$

Equation (3.5) may be rewritten as follows

$$-[J]_k^{-1} \times \begin{bmatrix} [h([Y]^t,[X]^t)] \\ [g([Y]^t,[X]^t)] \end{bmatrix}_k = \begin{bmatrix} [\Delta Y] \\ [\Delta X] \end{bmatrix}_k \tag{3.7}$$

Where $[\Delta Y]$ and $[\Delta X]$ are given by

$$\begin{aligned} [\Delta Y]_k &= [Y]_{k+1} - [Y]_k \qquad (a) \\ [\Delta X]_k &= [X]_{k+1} - [X]_k \qquad (b) \end{aligned} \tag{3.8}$$

From this equation and equation (3.7) $[Y]$ and $[X]$ at iteration $k+1$ can be evaluated.

$$\begin{bmatrix} [Y] \\ [X] \end{bmatrix}_{k+1} = \begin{bmatrix} [Y] \\ [X] \end{bmatrix}_k - [J]_k^{-1} \times \begin{bmatrix} [h([Y]^t,[X]^t)] \\ [g([Y]^t,[X]^t)] \end{bmatrix}_k \tag{3.9}$$

Each iteration of the Newton-Raphson (N-R) requires the construction of the Jacobian matrix and its solution by sparse techniques for the correction vectors $\Delta[Y]$ and $\Delta[X]$, if equation (3.5) is not satisfied. This procedure is repeated until the convergence is obtained. In other words the solution of equations $[h([Y]^t,[X]^t)]$ and $[g([Y]^t,[X]^t)]$ is obtained by continuously approximating these expressions by linear functions until convergence is obtained.

The N-R algorithm can be summarised in the flow chart represented in figure (3.1).

Some proposals have been suggested to speed-up the N-R solution. These consist of using the same Jacobian matrix over a time step or not updating it for each iteration. This is advantageous in that the factorisation of the sparse Jacobian matrix is re-calculated less often. However, this results in a less accurate solution. Extrapolation of the voltage can also reduce the number of iterations required for convergence [106].

3.5.3 Extrapolation

In order to improve the approximated state variables at each time

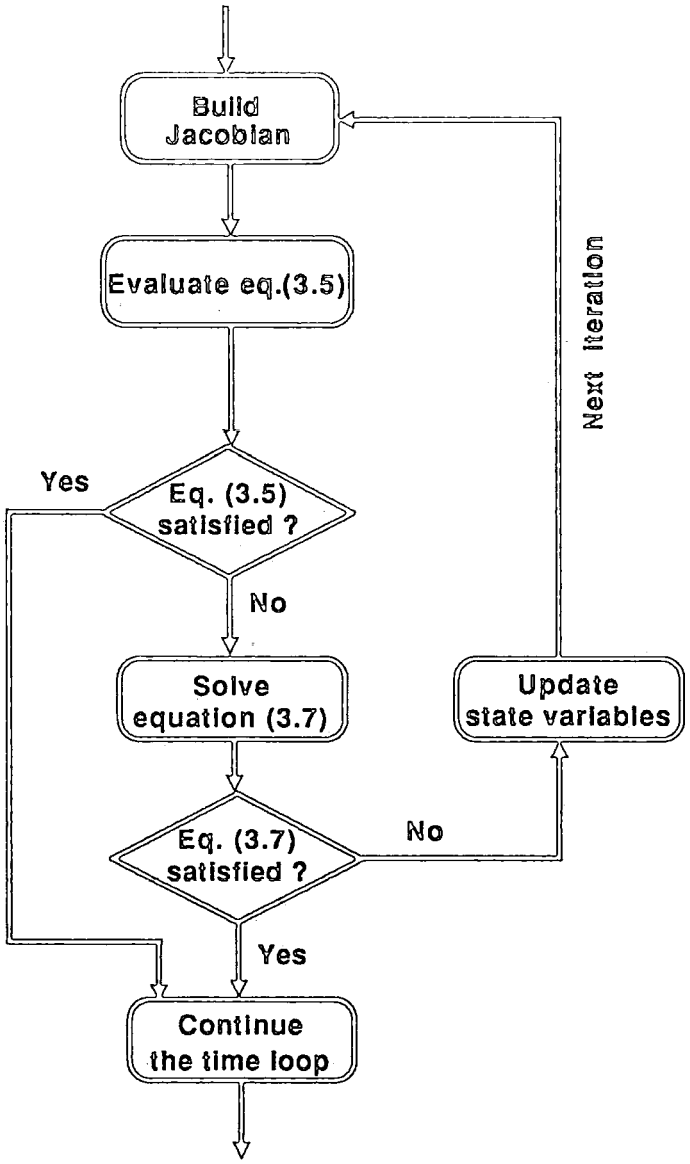


Fig. 3.1 The Newton-Raphson algorithm

step and improve the required starting values for the N-R method some of these quantities may be extrapolated. The adopted approach here is a simple linear extrapolation using the latest network solution. Stott [106] has adopted the extrapolation of the bus voltage only, whereas Irving [68] extrapolated the generator rotor angle as well as the bus voltage. According to Stott the voltage is better extrapolated in polar coordinates, because the voltage magnitude and angle tend to vary independently. Other dependent and independent variables can also be extrapolated if it is necessary.

The formula used for the variation of the rotor angle and the phase angle is as follows

$$\delta_{(t+\Delta t)} = \delta_t + 2\pi(f_{bus\,t} - f_o)\Delta t \tag{3.10}$$

$$\theta_{(t+\Delta t)} = \theta_t + 2\pi(f_{bus\,t} - f_o)\Delta t \tag{3.11}$$

$$\begin{aligned} V_{ri(t+\Delta t)} &= |V|_{it} \cos(\theta)_{(t+\Delta t)} \qquad (a) \\ V_{imi(t+\Delta t)} &= |V|_{it} \sin(\theta)_{(t+\Delta t)} \qquad (b) \end{aligned} \tag{3.12}$$

Where f_o, and $f_{bus\,t}$ are the nominal frequency and the busbar frequency at time step t respectively.

3.6 SOLUTION OF SIMULTANEOUS LINEAR EQUATIONS

In large power systems network analysis most of the computational capability is spent on solving repeatedly the above mentioned set of linearised equations (equation 3.7). To realise an efficient simulation of such systems in real- time, high speed computers are not sufficient. Sophisticated and fast mathematical techniques, and skilful programming are also necessary [2].

The Jacobian matrix is very sparse i.e. it contains only a few non zero elements. In certain cases such as triangular and band matrices where the non zero elements are concentrated on the upper or lower half of the matrix, or concentrated around the diagonal part of the matrix, even if the percentage of non zero elements is relatively high, the system is also defined as sparse [25]. The sparsity of the Jacobian matrix is mainly due to the structure of the electrical network (only a few lines are connected to the same node), and

the limited number of interfacing variables between the dynamic and the static subsystems.

Exploiting sparsity can result in enormous savings in computational time and storage space. Hence a set of simultaneous linear equations of the form of equation (3.7) with hundreds of variables can be solved quickly and efficiently, thus allowing for adequate solutions of larger systems and the consideration of more detailed models.

Many methods are available in the literature for solving linear sets of equations of the form $x = [A]^{-1}b$, but only the most practical ones are mentioned here. Full matrix methods are recommended for small systems, because the solution of such systems does not justify the effort of developing sophisticated algorithms. Whereas sparsity-oriented techniques are essential for large matrices. In this case the program should be skilfully organised so that the number of arithmetical operations is minimised i.e. preserves sparsity as much as possible and avoids all unnecessary operations such as repetition and evaluation of terms which involve zeros and ones.

Compared to their counterpart, sparsity techniques can improve the solution speed and size by a factor of ten or more [111].

3.6.1 Direct methods

Direct methods are simple techniques for solving sets of linear equations. They are very accurate especially if the pivotal elements of the coefficient matrix are predominant. However, they require large computer storage space and computation time if sparsity techniques are not used. Therefore, these methods are not recommended for large systems unless sparsity techniques are applied.

3.6.1.1 Gauss elimination

The simplest direct method is the Gauss elimination, where the coefficient matrix is transformed into a triangular matrix. The unknown variables are evaluated by back substitution.

Gauss elimination is considered as the basis of almost all modern methods dealing with the solution of linear equations. For instance, the LU method is obtained by decomposing the coefficient matrix into a lower and a unit upper triangular matrices. A further decomposition of the factor matrix [L] into a new unit lower triangular and a diagonal factor matrices has resulted in another approach called the LDU factorisation. The unknown variables in this case are calculated by forward and back substitutions.

These two methods implementing sparsity-directed schemes are widely used in power system analysis [94,113].

Gauss elimination requires about $\frac{1}{3}n^3$ operations to reduce a full matrix and $\frac{1}{2}n^2$ operations for the back substitution [11], for a matrix of order n. Methods which have such requirements are not recommended for solving large systems if sparsity is not exploited [4]. As stated in this reference, with sparsity programming the number of operations varies as a factor of n.

The standard OCEPS simulator uses the *Harwell* subroutine library for solving the set of simultaneous linear equations. This library is based on the sparsity-directed LU algorithm which can be described as follows

$$[A] = [L] * [U] \tag{3.13}$$

$$[A]x = [L][U]x = b \tag{3.14}$$

If $[U]x$ is substituted by the vector y this equation becomes

$$[L]y = b \tag{3.15}$$

$$y = [L]^{-1}b \tag{3.16}$$

The vector y is obtained by forward substitution and the unknown vector x is determined by back substitution since it is calculated by multiplying the vector y by the inverse of the upper factor matrix $[U]$.

$$x = [U]^{-1}y \tag{3.17}$$

3.6.2 Matrix inversion

In this case the unknown variables of expression (3.5) are determined by evaluating the inverse of the Jacobian matrix and multiplying it by the known vector as shown in equation (3.7) where the transpose of the state variables $[\Delta[Y], \Delta[X]]^t$ is equivalent to the unknown vector x and the transpose of the linearised functions $[[h([Y]^t, [X]^t)], [g([Y]^t, [X]^t)]]^t$ is equivalent to the known vector b.

3.6.2.1 Direct inversion methods

These methods evaluate explicitly the inverse matrix $[A]^{-1}$. They involve the direct manipulation of the matrices without exploiting any of their properties. The drawback of these methods is that, even if the original matrix is very sparse, its inverse is completely full. Therefore, the great advantage of sparsity in saving computational time and storage space is lost.

In addition, n^3 operations are needed for the solution, and n^2 words are required for the storage of the matrix.

3.6.2.2 Product form of the inverse matrix

In this case the inverse of a matrix can be obtained implicitly by subdividing it into n successive factor matrices. The factor matrices differ from the unit matrix in column i only. This method has an important application in the solution of linear programming problems [11], but, it shows inferiority in preserving sparsity [25].

3.6.2.3 The bifactorisation method

Zollenkopf [117] has combined the above mentioned approach and the LDU triangularisation method to develop a new technique which evaluates implicitly the inverse matrix and fully exploits the sparsity structure of the coefficient matrix (Jacobian matrix in our case). This technique used the second ordering scheme of Tinney [111]. Its name is derived from the evaluation of the inverse matrix which is given by the multiplication of 2n factor matrices. This method has so far been used for load-flow studies [35,66], and transient stability as

reported in a discussion on reference 110. It is fully explained in references 11, and 117.

Compared to the triangular factorisation, the asymmetric matrix storage requirements are similar for both the *bifactorisation* and triangular decomposition. However, the computational effectiveness is higher for the former because it is evaluating the inverse matrix instead of decomposing the coefficient matrix.

In order to ensure stability and accuracy of the *bifactorisation* technique, the sparse matrix should satisfy the following 3 conditions:

(1) the sparsity structure of the matrix must be symmetrical,
(2) the matrix should be non singular i.e. its inverse exists, and
(3) its diagonal elements should be large in comparison with the off-diagonal terms included in the corresponding rows or columns.

If the first condition is not satisfied some of the zero elements of the sparse matrix have to be stored and processed.

The third condition reduces the effect of the computational round-off errors [4]. In the case of ill-conditioned problems i.e. some of the diagonal elements are small compared with off-diagonal terms, the largest elements should be used as pivots and high floating-point precision is required to reduce round-off errors [11]. The first option is the classical way of selecting the pivots in order to maintain numerical stability. This alternative is not very efficient in preserving sparsity. Therefore, a compromise between accuracy and preserving the sparsity of the matrix should be considered. In our case double precision number representation can be used (if the computers word length is 32 bits) together with setting a low limit that the diagonal elements should not violate. Generally, the dynamic sub-matrices resulting from the discretisation of the differential equations have predominant diagonal elements [25], and the network sub-matrix can also be diagonally dominant if series capacitances which can cancel the series reactances are not included.

The *bifactorisation* method defines implicitly the inverse matrix as the multiplication of n left hand side and n right hand side factor matrices, satisfying the following condition:

$$[L]^n * [L]^{n-1} \dots [L]^1 * [A] * [R]^1 * [R]^2 \dots [R]^{n-1} * [R]^n = [I] \qquad (3.18)$$

Where

[A] is the $n * n$ sparse coefficient matrix,

$[L]^p$ and $[R]^p$ are the left hand side and the right hand side factor matrices, and

[I] is a unit matrix. It should be noted that $[R]^n = [I]$.

From this equation the inverse matrix can be expressed implicitly in terms of these factor matrices as follows

$$[A]^{-1} = [R]^1 * [R]^2 \dots [R]^{n-1} * [L]^n * [L]^{n-1} \dots [L]^2 * [L]^1 \qquad (3.19)$$

$[L]^p$ and $[R]^p$ differ from the unit matrix in the elements of the p^{th} column and row respectively. Their diagonal and off-diagonal elements are given by

$$\begin{aligned} {l_{pp}}^p &= \frac{1}{{a_{pp}}^{p-1}} \quad (a) & {l_{ip}}^p &= \frac{-{a_{ip}}^{p-1}}{{a_{pp}}^{p-1}} \quad (b) \\ {r_{pp}}^p &= 1 \quad (c) & {r_{pk}}^p &= \frac{-{a_{pk}}^{p-1}}{{a_{pp}}^{p-1}} \quad (d) \end{aligned} \qquad (3.20)$$

The reduced matrix elements are given by

$${a_{pp}}^p = 1; \qquad {a_{ip}}^p = 0; \qquad {a_{pk}}^p = 0;$$

$${a_{ik}}^p = {a_{ik}}^{p-1} - \frac{{a_{ip}}^{p-1}{a_{pk}}^{p-1}}{{a_{pp}}^{p-1}}$$

for $i, k = p+1, \dots n$; where p is the pivotal index.

In the case of an asymmetrical coefficient matrix, a further decomposition of the factor matrix $[L]^p$ into a diagonal factor matrix $[D]^p$ and an off-diagonal factor matrix $[C]^p$ is advantageous.

$$[L]^p = [C]^p * [D]^p$$

$[D]^p$ is given by

$$[D]^p = \begin{bmatrix} 1 & & & & \\ & \ddots & & 0 & \\ & & d_{pp}{}^p & & \\ & 0 & & \ddots & \\ & & & & 1 \end{bmatrix} \tag{3.21}$$

Where $d_{pp}{}^p = l_{pp}{}^p$

$[C]^p$ is given by

$$[C]^p = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & & & \\ \vdots & & \ddots & & \\ \vdots & & & 1 & \\ 0 & & & c_{p+1,p}{}^p & \\ \vdots & & & \vdots & \ddots \\ 0 & & & c_{n,p}{}^p & 1 \end{bmatrix} \tag{3.22}$$

Where

$$c_{ip}{}^p = \frac{l_{ip}{}^p}{l_{pp}{}^p} = -a_{ip}{}^{p-1} \qquad \text{for } i = p+1, \ldots n$$

The general form of the right hand side factor matrix is as follows

$$[R]^p = \begin{bmatrix} 1 & & & & & \\ & \ddots & & 0 & & \\ 0 & \cdots & 1 & r_{p,p+1}{}^p & \cdots & r_{p,n}{}^p \\ & & & 1 & & \\ & & 0 & & \ddots & \\ & & & & & 1 \end{bmatrix} \tag{3.23}$$

The non zero elements of the factor matrix $[R]^p$ are expressed by equation (3.20 d).

3.6.3 Principles of the bifactorisation technique

To fully exploit the advantages offered by sparsity-directed programming, such as preserving the sparse structure of the matrix, and reducing the computational time and storage space, the algorithm which has been developed is subdivided into the following independent subprograms:

(1) build the matrix into a compact form and store all the necessary indices;

(2) ordering of the compact matrix and simulation of the factorisation;
(3) reduction or factorisation of the ordered matrix; and
(4) direct solution.

These steps are illustrated by the flow chart given in figure (3.2).

3.6.3.1 Compacting and storing the sparse matrix

Conventional methods require n^2 words for storing $n*n$ matrices. Sparse matrix programming stores only the non zero elements in one or more vectors, together with tables of their indices and addresses to locate and identify them easily. The Zollenkopf algorithm requires only about $4 * (2 * b) + 4 * n$ memory space, where b indicates the number of branches and n is the dimension of the coefficient matrix. These non zero elements can be stored in any order.

The diagonal elements are all non zeros, and their positions are consecutive. Therefore, their location can be known implicitly without requiring any indexing. Hence a reduction in storage space can be achieved if they are stored separately in their natural order in a vector DE.

The off-diagonal elements are stored column-wise and row-wise in two different tables CE and RE. This gives a duplicated copy of these non zero elements. This may seem a waste of memory space, however, it avoids the use of a search subroutine which may slow down the process. Furthermore, the duplicated version is deleted at a point in the program when it is not needed any more. The resulting vacant memory space can be used for storing the new generated elements during the reduction phase described below. These free locations are stored in table NEWZ. The next vacant location is indicated by LF.

The number of non zero elements per column NOZE is also stored. This information is used in the process of reordering the coefficient matrix described below. During the simulation step the number of non zero elements is updated whenever an element is to be generated or deleted. Using this information the natural order of the different columns is altered according to the adopted

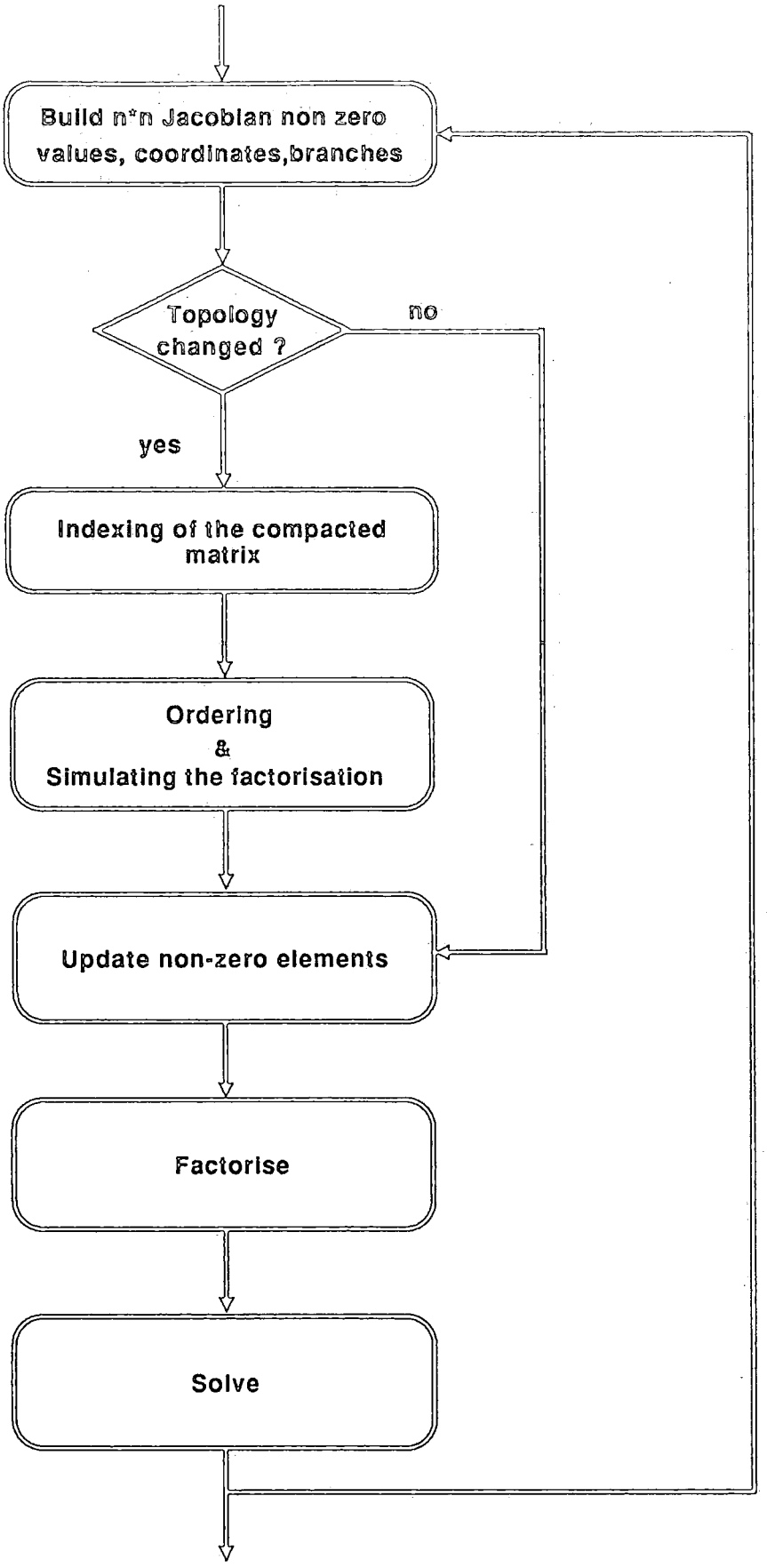


Fig. 3.2 Modified Zollenkopf algorithm

reordering scheme and stored in another table NSEQ. This implies that the columns are not really interchanged, but the pivotal order is kept in this table.

The location and the necessary indices of the different off-diagonal non zero elements of the $n * n$ coefficient matrix are required so that the elements in the compacted list can be identified easily. This is indicated by three tables.

(1) ITAG indicates the rows of the off-diagonal elements stored column-wise and the columns of the off-diagonal elements stored row-wise.
(2) LCOL contains pointers to the location of the first non zero element for each column.
(3) LNXT stores pointers to the next element in the list of the off-diagonal terms. It is set to zero for the last non zero term in the column.

This information is sufficient to reconstruct the coefficient matrix and perform the reduction process in any desired order. These variables are modified whenever an element is deleted or generated.

Since the position of the different elements remains the same if the structure of the matrix is not changed, the indexing of the off-diagonal elements is not determined repeatedly, but it is updated whenever the topology changes.

In order to illustrate the storage and indexing of the coefficient matrix, a small network consisting of 3 generators, 7 nodes, and 9 lines is used as an example. The topology of this network is shown in figure (3.3). Figure (3.4) illustrates the network-type representation of the dynamic and static elements of this system. Figure (3.5) represents the structure of the corresponding modified Jacobian matrix. Further details of the formation of the Jacobian matrix are given in chapter 4 and appendix A and the modification of this matrix is presented in chapter 6. This example is also used in chapter 6 to explain the programming issues of the different schemes involved in the process of solving the set of linear equations using the *bifactorisation* technique.

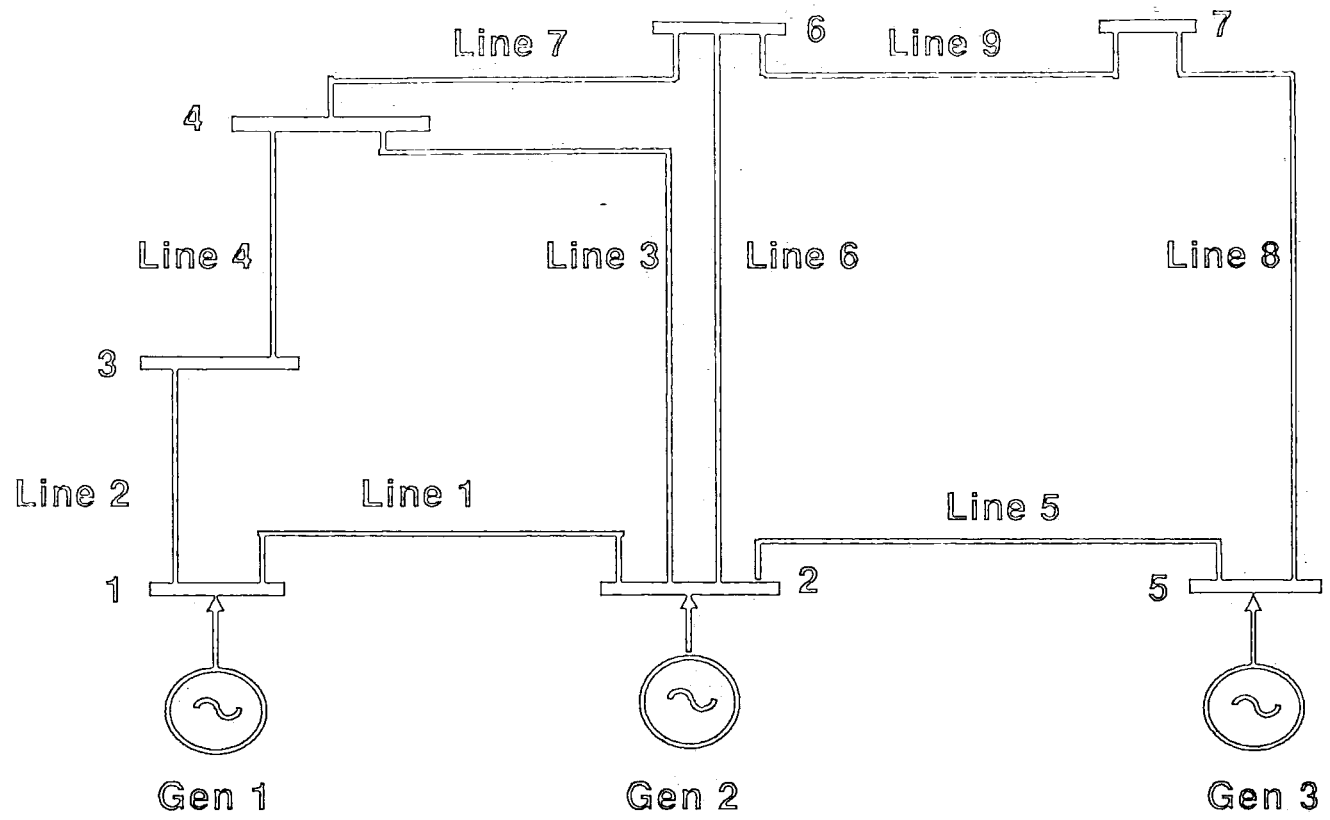


Fig. 3.3 Example system

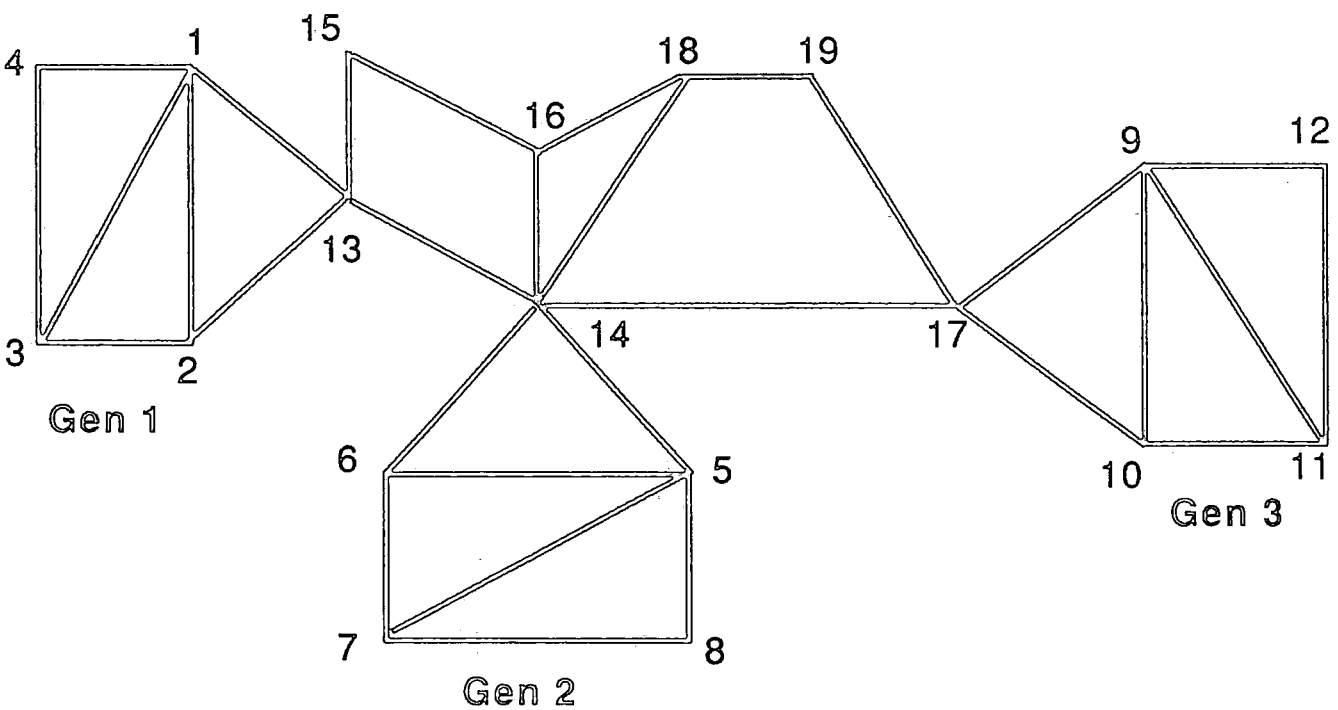


Fig. 3.4 Network-type representation of the example system

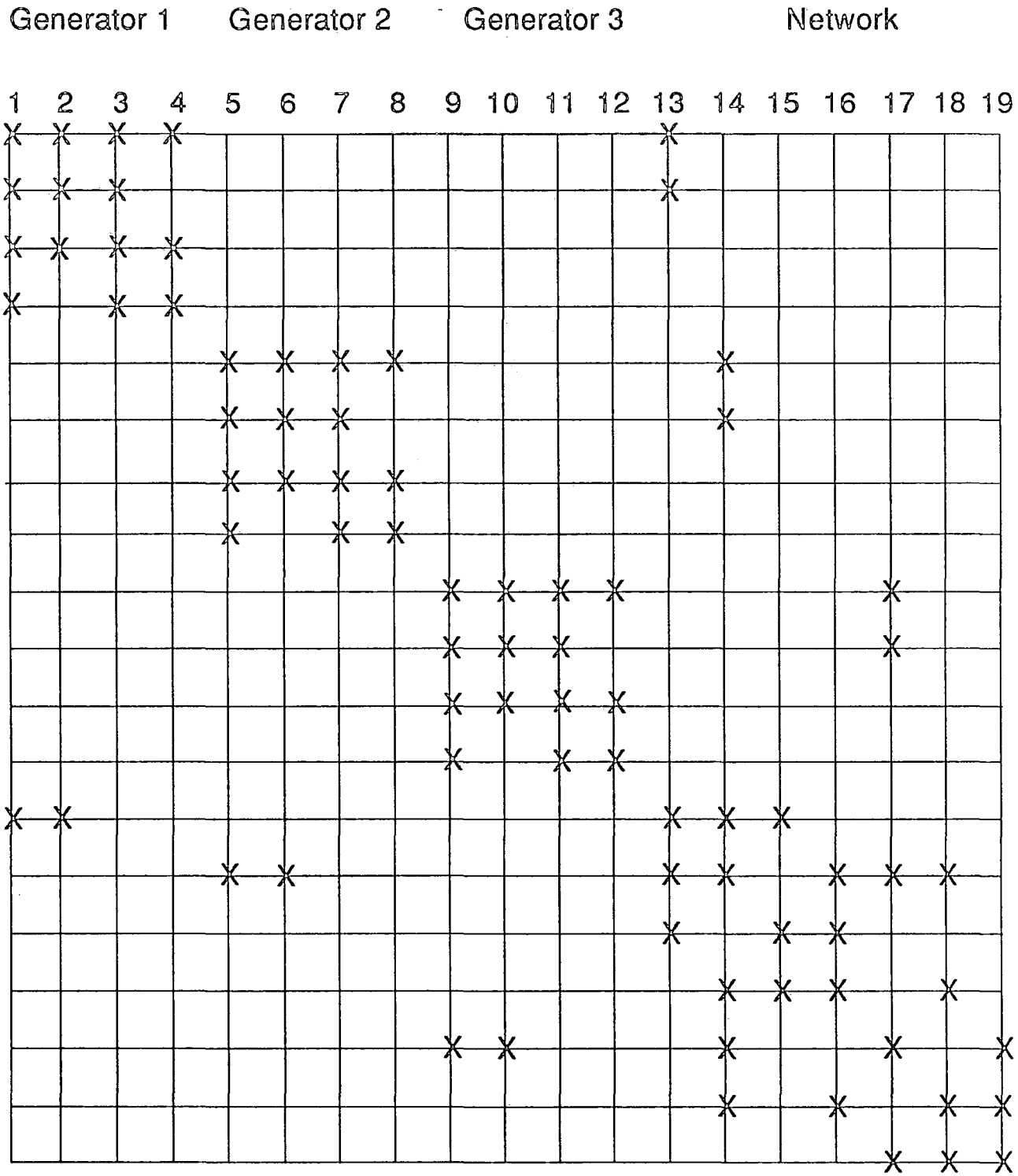


Fig. 3.5 Modified Jacobian matrix of the example network

Although the dynamic components are not related to a physical network, using the coefficient matrix associated to these elements a network-type structure can be deduced, as shown in figure (3.4). The nodes related to the generators refer to rows and columns of the dynamic sub-Jacobian which is represented by the $4 * 4$ diagonal blocks in figure (3.5).

The process of storing the non zero elements and the corresponding indices involved in this example is shown in tables (3.1) and (3.2). It should be noted that the right half of table (3.2) is a continuation of its left half.

3.6.3.2 Simulation and ordering

In order to minimise the number of fill-ins and preserve the sparsity a scheme referred to as optimal ordering has to be used. However, it is practically impossible to obtain an exact optimum [11,25,117]. Sophisticated ordering schemes are generally time consuming and complicated to programme. Therefore, their benefit in reducing the number of arithmetical operations by minimizing the number of new generated elements is lost [4,11,117].

A compromise between the least number of operations, and processing time and memory space is necessary. Thus a sub-optimal ordering process which has a performance close to that of the optimal ordering would be enough. In order to choose an efficient scheme, the effect of the ordering on the factorisation and solution should be considered. Therefore, it is not adequate to select a very fast ordering method which would produce inefficient reduction and solution.

Many ways of ordering sparse matrices are available. Among these there is the bandwidth minimisation method, dynamic ordering, static ordering etc. The former option involves more fill-ins, but this does not alter the matrix structure [52]. Since this is the case and this approach cannot be used for all matrices it is rarely used in power system network problems. The static ordering is more suitable for the solution of systems where the coefficient matrix remains constant [25]. The least number of non zeros dynamic ordering technique is the most commonly used for the pre-elimination process [11,111].

Location	LCOL	NOZE	NSEQ	DE
1	31	5	14	$a_{1\ 1}$
2	1	4	7	$a_{2\ 2}$
3	2	4	8	$a_{3\ 3}$
4	3	3	1	$a_{4\ 4}$
5	38	5	15	$a_{5\ 5}$
6	8	4	9	$a_{6\ 6}$
7	9	4	10	$a_{7\ 7}$
8	10	3	2	$a_{8\ 8}$
9	45	5	16	$a_{9\ 9}$
10	15	4	11	$a_{10\ 10}$
11	16	4	12	$a_{11\ 11}$
12	17	3	3	$a_{12\ 12}$
13	4	5	17	$a_{13\ 13}$
14	11	7	19	$a_{14\ 14}$
15	23	3	4	$a_{15\ 15}$
16	24	4	13	$a_{16\ 16}$
17	18	5	18	$a_{17\ 17}$
18	27	3	5	$a_{18\ 18}$
19	29	3	6	$a_{19\ 19}$

Table 3.1 Storage of a sparse matrix

The ordering scheme is the most time consuming process in the solution of sets of linear equations especially for large systems. However, since the ordering is carried out only once for each coefficient matrix structure, the overall solution of the repetitive solutions where the Jacobian matrix is updated at each iteration is speeded-up greatly.

The dynamic ordering consists of simulating the order in which the reduction is going to be carried out. It starts with a pivotal search during

Location	ITAG	LNXT	CE	RE
1	1	35	$a_{1\,2}$	$a_{2\,1}$
2	1	5	$a_{1\,3}$	$a_{3\,1}$
3	1	7	$a_{1\,4}$	$a_{4\,1}$
4	1	6	$a_{1\,13}$	$a_{13\,1}$
5	2	37	$a_{2\,3}$	$a_{3\,2}$
6	2	52	$a_{2\,13}$	$a_{13\,2}$
7	3	0	$a_{3\,4}$	$a_{4\,3}$
8	5	42	$a_{5\,6}$	$a_{6\,5}$
9	5	12	$a_{5\,7}$	$a_{7\,5}$
10	5	14	$a_{5\,8}$	$a_{8\,5}$
11	5	13	$a_{5\,14}$	$a_{14\,5}$
12	6	44	$a_{6\,7}$	$a_{7\,6}$
13	6	22	$a_{6\,14}$	$a_{14\,6}$
14	7	0	$a_{7\,8}$	$a_{8\,7}$
15	9	49	$a_{9\,10}$	$a_{10\,9}$
16	9	19	$a_{9\,11}$	$a_{11\,9}$
17	9	21	$a_{9\,12}$	$a_{12\,9}$
18	9	20	$a_{9\,17}$	$a_{17\,9}$
19	10	51	$a_{10\,11}$	$a_{11\,10}$
20	10	26	$a_{10\,17}$	$a_{17\,10}$
21	11	0	$a_{11\,12}$	$a_{12\,11}$
22	13	54	$a_{13\,14}$	$a_{14\,13}$
23	13	55	$a_{13\,15}$	$a_{15\,13}$
24	14	25	$a_{14\,16}$	$a_{16\,14}$
25	15	58	$a_{15\,16}$	$a_{16\,15}$
26	14	0	$a_{14\,17}$	$a_{17\,14}$
27	14	28	$a_{14\,18}$	$a_{18\,14}$
28	16	0	$a_{16\,18}$	$a_{18\,16}$
29	17	30	$a_{17\,19}$	$a_{19\,17}$

Location	ITAG	LNXT	CE	RE
30	18	0	$a_{18\,19}$	$a_{19\,18}$
31	2	32	$a_{2\,1}$	$a_{1\,2}$
32	3	33	$a_{3\,1}$	$a_{1\,3}$
33	4	34	$a_{4\,1}$	$a_{1\,4}$
34	13	0	$a_{13\,1}$	$a_{1\,13}$
35	3	36	$a_{3\,2}$	$a_{2\,3}$
36	13	0	$a_{13\,2}$	$a_{2\,13}$
37	4	0	$a_{4\,3}$	$a_{3\,4}$
38	6	39	$a_{6\,5}$	$a_{5\,6}$
39	7	40	$a_{7\,5}$	$a_{5\,7}$
40	8	41	$a_{8\,5}$	$a_{5\,8}$
41	14	0	$a_{14\,5}$	$a_{5\,14}$
42	7	43	$a_{7\,6}$	$a_{6\,7}$
43	14	0	$a_{14\,6}$	$a_{6\,14}$
44	8	0	$a_{8\,7}$	$a_{7\,8}$
45	10	46	$a_{10\,9}$	$a_{9\,10}$
46	11	47	$a_{11\,9}$	$a_{9\,11}$
47	12	48	$a_{12\,9}$	$a_{9\,12}$
48	17	0	$a_{17\,9}$	$a_{9\,17}$
49	11	50	$a_{11\,10}$	$a_{10\,11}$
50	17	0	$a_{17\,10}$	$a_{10\,17}$
51	12	0	$a_{12\,11}$	$a_{11\,12}$
52	14	53	$a_{14\,13}$	$a_{13\,14}$
53	15	0	$a_{15\,13}$	$a_{13\,15}$
54	16	56	$a_{16\,14}$	$a_{14\,16}$
55	16	0	$a_{16\,15}$	$a_{15\,16}$
56	17	57	$a_{17\,14}$	$a_{14\,17}$
57	18	0	$a_{18\,14}$	$a_{14\,18}$
58	18	0	$a_{18\,16}$	$a_{16\,18}$

Table 3.2 Storage of a sparse matrix

which the pivotal column is selected. This column is defined as the one with the least number of non zero elements that has not been pivotal before. If a few columns have the same number of non zero elements, the pivotal column is chosen arbitrarily by considering the ascending column index. The advantage of this process is minimising the number of fill-ins generated during the reduction phase.

Before considering the next pivotal search, a simulation of the elimination of the present pivot must be carried out, this is because during the elimination some elements are deleted and some may be generated. Therefore, the number of non zero elements is changing through out the process.

The factor matrix elements are to be stored in the deleted element locations which correspond to their indices. For instance r_{12} is stored in a_{12} location when r_{12} is calculated and a_{12} is deleted.

3.6.3.3 Reduction of the ordered matrices

The programming of the reduction of the ordered matrix is not as simple as the conventional factorisation of the original coefficient matrix. This process is upset because the coefficient matrix is not fully stored as an $n * n$ matrix but only the non zero elements are stored in a compact form. Although this compacting requires additional programming and execution time simply to identify each element, it significantly reduces the storage space. This would allow for storing very large matrices without any restrictions. Furthermore, the extra computational time for indexing is small because the indexing and the simulation of the factorisation is carried out only once for every unaltered matrix structure as shown in figure (3.2). It is this aspect of programming which can improve the efficiency of an algorithm, because in some cases, even if sparsity techniques are used, the solution may not be very efficient.

The factorisation is thus reduced to the arithmetical determination of the different factor matrix elements and the reduced coefficient matrix. The pivotal columns and rows of the reduced matrix are eliminated. These free positions can be used to store the newly processed factor matrix elements. The elements

involved are identified by their indices. This process is required only once if all non zero elements remain constant. However, this is often not the case if the problem is solved repeatedly, unless the Jacobian matrix is held constant between one iteration and the next.

3.6.3.4 Direct solution

This subprogram is concerned with the determination of the unknown vector by a successive multiplication of the factor matrices by a vector. The initial vector in this operation is the known vector b which is represented by the right hand side member of equation (3.5) in this case. The vector elements obtained are successively stored in the locations of the original vector.

Compared to the execution time of the different subroutines of the *bifactorisation* technique the solution routine is the one which takes the least computational time. The ordering and simulation is the slowest process. Consequently the separation of the *bifactorisation* algorithm into different independent subprograms is crucial, especially for iterative solutions of large systems.

3.7 DECOMPOSED METHODS

The development of decomposed methods started as early as 1958 when attempts have been made to achieve better computing performance by using multiple network tearing and other decomposed algorithms to solve large mathematical problems. The use of these techniques in well defined problems has been motivated by the rapid development of microprocessors which are expected to match the performance of present day main-frames, if used in parallel, at a fraction of the cost [80,92].

Decomposed methods can be defined as the simultaneous use of two or more processing elements of the same capabilities to solve a problem. These processing elements can be minicomputers, microprocessors or even main-frames. Parallelism can also be exploited by using machines with different capabilities. This possibility is described in chapters 5 and 6 where an Array Processor FPS 5205 is used in parallel with a Perkin Elmer PE 3230 minicomputer. The

powerful machine can be used to solve the majority of the problem and the less powerful one can be used as a host computer to perform minor tasks. This section is devoted to the former alternative where machines of the same nature and capability ought to be used. This requires the decomposition of the problem to be solved in parallel into smaller sub-problems

There are two fundamental methods for decomposing systems. The partitioned matrix method and the diakoptics approach. The first method consists of partitioning the coefficient matrix into sub-matrices. The second approach consists of tearing the power system into small subsystems. In general, compared to matrix partitioning the diakoptical method requires less storage and computational time [11,80]. Furthermore, from the point of view of programming effort, tearing the network into subsystems is more advantageous because these sub-networks can be treated, as a first step, as isolated from each other. Then in a subsequent step the solution is modified to take the interconnection into consideration. In this way a program developed for the single processor approach can be directly implemented in the decentralised algorithm with only minor changes to account for the tie nodes injected currents into the isolated areas. These techniques could be even more efficient if the partitioning or the splitting-up is done at points with the minimum of coupling. These can be identified by observing the network topology, or automatically by using special programs [11,67,92,106].

3.7.1 Decoupled technique configurations

Decomposed methods are categorised as multiple instruction stream multiple data stream systems where many instructions can be carried out on different sets of data simultaneously. They are subdivided into tightly coupled schemes and loosely coupled ones. The selection of these methods may be dictated by the purpose of the study and the nature of the problem. For instance, a loosely coupled method can be more suitable for processing the different grid areas which may be considered as independent entities. In this case each area has its own data and private memory, therefore the processors can be geographically remote from each other. The data transfer between processors is accomplished via hardware links. Supplementary code and instructions are

necessary for this function and the synchronisation of processors. In this case the problem of data corruption may occur.

The tightly coupled configuration (also referred to as multiprocessor) consists of a few processors sharing a common memory or having bus links connecting their local memories. These systems may also require the existence of an identical operating system for task distribution, synchronisation etc., and the use of the same peripherals.

3.7.2 Network tearing

Two forms of diakoptics are possible: the conventional method cutting branches linking two areas, and the node-tearing technique. These two methods not only differ in the way the network is partitioned but also in the selection of the coordinating variables. In the first case the coordination variables are tie line currents. In the second the voltage at the torn nodes are used as coordinating variables. The combination of the two techniques is also possible [116].

Since in most power systems the usual network split occurs at the transmission line level, the use of the line tearing is more often adopted. A problem associated with the reference node may occur if a branch-cutting method is used and some of the areas do not include any earthed equipment such as shunts etc. In this case the voltages in each area are calculated referring to a local reference node or voltage. The problem arises when the relative voltage and phase angle between two areas have to be evaluated to calculate the power and current flows through the tie lines. This problem is easily overcome by considering node tearing, since the node busbars have the same voltage [41,116]. In our case this approach is not necessary since it is unlikely that this problem would arise. Power systems generally include a variety of elements which are linked to the ground. Among these are the shunt compensators, the line charging, the loads etc.

Another way of decomposing the power system simulator is the use of dynamic/algebraic decomposition. This is usually used when the partitioned

methods for computing the differential and algebraic equations are adopted. In this case the dynamic and network models are solved separately by different numerical methods [41] and different processors.

This chapter has highlighted the most suitable algorithms to be used for real time dynamic simulation. These algorithms are fast stable and quite accurate for the range of time of interest. The next chapter will concentrate on the modelling of the different simulated elements of the network. The simplifications adopted to help speeding-up the simulator are also mentioned and examples on the relationship between the numerical algorithms and the mathematical models are presented.

CHAPTER 4

MATHEMATICAL MODELLING OF POWER SYSTEMS

4.1 INTRODUCTION

This chapter describes the modelling of generating units and their connection to the loads through the transmission lines. It explains the function of the components of a power system and their contribution towards maintaining the stability and reliability of the system. Power system local control is mainly ensured by excitors, prime movers, boilers at the generating units, and by breaker switching at the substations. Other local controllers include transformer tap-changing and phase-shifting, and VAr compensators. The features and characteristics of these power system components are fully described.

The physical components of electrical power systems can be represented by mathematical models. In order to develop a flexible simulator suitable for operation under a wide range of dynamic operating conditions, and achieve an acceptable compromise between the conflicting requirements of accuracy and execution speed, it is always desirable to choose the computationally simplest models.

The power systems behaviour is described by two sets of equations.

(1) A set of differential equations simulating the dynamic behaviour of the generating units, and their local control systems and plant such as the governors, boilers etc. These equations are given by the following general expression

$$[\dot{Y}] = [f([Y]^t, [X]^t)] \tag{4.1}$$

(2) A set of algebraic equations representing the network steady state. These equations are given by the following equation

$$[g([Y]^t, [X]^t)] = 0 \tag{4.2}$$

Where the vector $[Y]$ includes the state variables related to the dynamic components of the electrical system, and the vector $[X]$ consists of the network state variables.

The models described in this chapter are based on those presented in reference 93, with some revisions and extensions.

4.2 DYNAMIC MODELS

The different elements of a power system are represented by simplified mathematical models. This is because the time scale adopted does not allow for the consideration of the subtransient and fast transient studies. Therefore, these models cannot provide a realistic response of the system during the very short periods following the disturbances. However, they are efficient for mid- and long-term dynamic studies which are the purpose of this project. The objective of these simplifications is to allow the adopted mathematical models to contribute to speeding up the simulator. The consideration of both short-term and long-term studies would slow down the simulator, because these combined studies will require either the use of two or more different time steps, or the execution of the short- and long-term dynamics by two separate programs. In addition, these time steps should be small enough to avoid the stiffness problem, and maintain numerical stability. Since this simulator does not require detailed modelling of the different physical plants, the dynamic elements are simplified to low order differential equations.

4.2.1 Generator models

For large scale studies the electrical behaviour of all three phase balanced machines is simplified to an equivalent circuit in the rotor direct and quadrature axis [106]. To speed-up the simulator the following assumptions and simplifications of the generator models have been adopted:

(1) The stator is represented by the positive sequence impedance, and considered to be operating at steady state. Thus it is modelled as an algebraic equation similar to the network models.

(2) The subtransient and transient states are not considered and the generator variables are assumed to vary slowly from one state to another.

(3) The saturation of the magnetic circuits of the generator is not included.

(4) By suitable arrangement of the stator and rotor winding and of the magnetic circuit in the alternators, harmonics are not generated except in very small percentages under transient conditions [93]. Harmonics may therefore be neglected.

(5) In most of the modern synchronous generators related to thermo-turbines saliency is neglected. Therefore, the direct and quadrature axis reactances can be assumed equal.

(6) For the sake of simplicity of the models the generator parameters are taken as independent of the variation of the frequency. However, this can be included if required [93].

(7) It has been common practice to model the damper windings as an artificial coefficient when these are not considered [15,44,68,93].

(8) In the balanced three phase systems the single phase variables are based on positive sequence components.

Consequently, the complex circuits of the synchronous generator can be simplified into the conventional equivalent circuit given in figure (4.1).

The single phase vector diagram corresponding to this equivalent circuit is shown in figure (4.2).

4.2.1.1 Electrical equations

The relationship between the different variables are illustrated by the following expressions. It should be noted that the generators reference frame is fixed to the rotor, and only the positive sequence components are modelled.

$$\bar{I}_g = I_d + jI_q \tag{4.3}$$

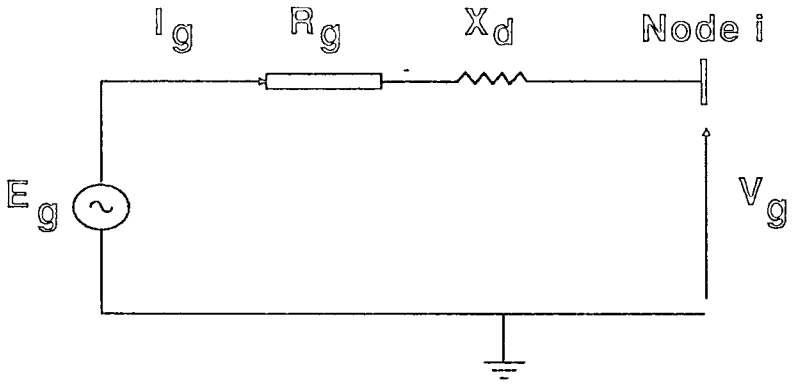


Fig. 4.1 Generator equivalent circuit

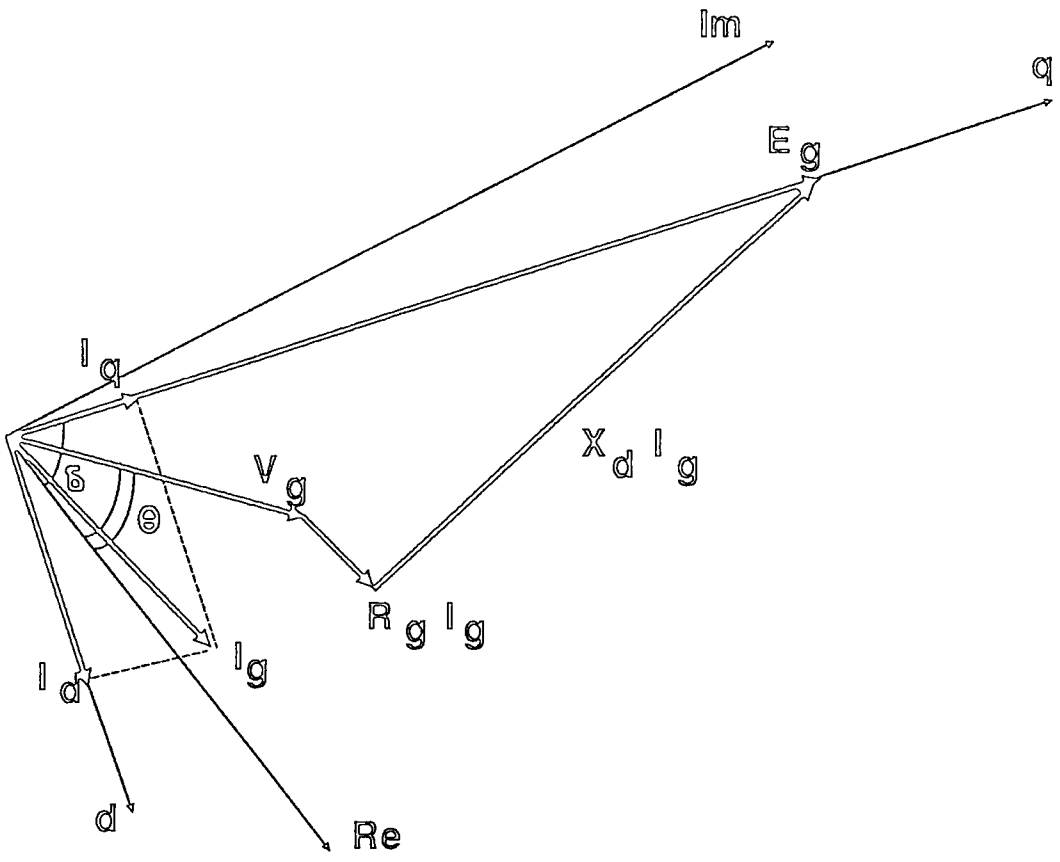


Fig. 4.2 Approximate positive sequence diagram

$$\bar{V}_g = V_d + jV_q \tag{4.4}$$

The above-mentioned assumption (5) allows the simplification of the equivalent positive sequence impedance as follows

$$\bar{Z}_g = R_g + jX_d \tag{4.5}$$

Therefore the terminal voltage can be expressed as

$$\bar{V}_g = \bar{E}_g - \bar{Z}_g * \bar{I}_g \tag{4.6}$$

Replacing $\bar{V}_g$, $\bar{Z}_g$, and $\bar{I}_g$ by their expressions, equation (4.6) becomes

$$V_d + jV_q = j|E_g| - (R_g + jX_d)(I_d + jI_q) \tag{4.7}$$

From this equation the direct and quadrature components of the terminal voltage $\bar{V}_g$ can be deduced

$$\begin{aligned} V_d &= -R_g * I_d + X_d * I_q \qquad (a) \\ V_q &= |E_g| - (R_g * I_q + X_d * I_d) \qquad (b) \end{aligned} \tag{4.8}$$

From these two equations the direct and the quadrature components of the current $\bar{I}_g$ can be determined

$$\begin{aligned} I_d &= \frac{1}{|Z_g|^2}[X_d(|E_g| - V_q) - R_g * V_d] \qquad (a) \\ I_q &= \frac{1}{|Z_g|^2}[R_g(|E_g| - V_q) + X_d * V_d] \qquad (b) \end{aligned} \tag{4.9}$$

The apparent power $\bar{S}_g$ is given by

$$\bar{S}_g = \bar{V}_g * \bar{I}_g^* = P_g + jQ_g \tag{4.10}$$

Substituting $\bar{V}_g$ and $\bar{I}_g$ by their components in equation (4.10) will result in the following expressions for the active and reactive power respectively.

$$\begin{aligned} P_g &= V_d * I_d + V_q * I_q \qquad (a) \\ Q_g &= V_q * I_d + V_d * I_q \qquad (b) \end{aligned} \tag{4.11}$$

Substituting (4.9) into (4.11) gives

$$P_g = \frac{|E_g|}{|Z_g|^2}(X_d * V_d + R_g * V_q) - |V_g|^2 * \frac{R_g}{|Z_g|^2} \qquad (a)$$
$$Q_g = \frac{|E_g|}{|Z_g|^2}(X_d * V_q - R_g * V_d) - |V_g|^2 * \frac{X_d}{|Z_g|^2} \qquad (b) \qquad (4.12)$$

The interface between the network and the generators requires the expression of the generator variables in the network reference frame. For instance, the direct and quadrature coordinates of the terminal voltage are expressed as follows

$$V_d = |V_g| * \sin(\delta - \theta) \qquad (a)$$
$$V_q = |V_g| * \cos(\delta - \theta) \qquad (b) \qquad (4.13)$$

in the generator reference frame, and it is given by this expression

$$V_{rg} = |V_g| * \cos(\theta) \qquad (a)$$
$$V_{img} = |V_g| * \sin(\theta) \qquad (b) \qquad (4.14)$$

in the network reference frame. Therefore, the active and reactive power of the generators can be expressed by the following equations in this reference frame

$$P_g = \frac{|E_g|}{|Z_g|^2}[V_{rg}(X_d * \sin(\delta) + R_g * \cos(\delta)) + V_{img}(R_g * \sin(\delta) - X_d * \cos(\delta))] - \frac{R_g}{|Z_g|^2} * |V_g|^2 \qquad (a)$$
$$Q_g = \frac{|E_g|}{|Z_g|^2}[V_{rg}(X_d * \cos(\delta) - R_g * \sin(\delta)) + V_{img}(X_d * \sin(\delta) + R_g * \cos(\delta))] - \frac{X_d}{|Z_g|^2} * |V_g|^2 \qquad (b) \qquad (4.15)$$

Where

$\bar{I}_g$ is the positive sequence injected current at the generator g busbar in (p.u.).

$\bar{V}_g$ is the positive sequence terminal voltage of generator g in (p.u.).

$\bar{E}_g$ is the positive sequence internal voltage of generator g in (p.u.).

$\bar{Z}_g$ is the positive sequence impedance of generator g in (p.u.).

$\bar{S}_g$ is the positive sequence apparent power of generator g in (p.u.).

P_g is the positive sequence active power of generator g in (p.u.).

Q_g is the positive sequence reactive power of generator g in (p.u.).

4.2.1.2 Electromechanical equations

The motion of the generator is caused by a mechanical torque Tm developed by the turbine. The generator produces an opposing torque T_g as shown in figure (4.3). In a thermal power station the mechanical power P_m is caused by the steam flow through the turbine. The electrical power P_g is generated by the interaction between the rotor and the stator magnetic fields. The steady state, where the rotational speed ω is constant, is due to the equality between these two torques. Whenever there is an over-speed caused by an increase of the mechanical torque, or an under-speed due to an increase in the electrical torque, the whole system is affected. To overcome such an imbalance the rotating speed should be kept as close as possible to its normal value. This can be realised by the damper windings and governor action. Therefore, the variation of the rotating speed depends also on the damping torque which is created by the damper windings. This torque damps the local oscillations.

$$P_m - P_g = M * \frac{d\omega}{dt} + D(\omega - \omega_a) \tag{4.16}$$

$$\frac{d\omega}{dt} = \frac{1}{M}[P_m - P_g - D(\omega - \omega_a)] \tag{4.17}$$

Where M is given by

$$M = \frac{(2 * H)}{2\pi * f_o} \tag{4.18}$$

Substituting M in equation (4.17) gives

$$\frac{d\omega}{dt} = \frac{\pi * f_o}{H}[P_m - P_g - D(\omega - \omega_a)] \tag{4.19}$$

The rotor angle δ is related to the rotating speed ω by the following expression

$$\frac{d\delta}{dt} = \Delta\omega \tag{4.20}$$

Where

H is the inertia constant in (s).

P_m is the mechanical power in (p.u.).

P_g is the electrical power in (p.u.).

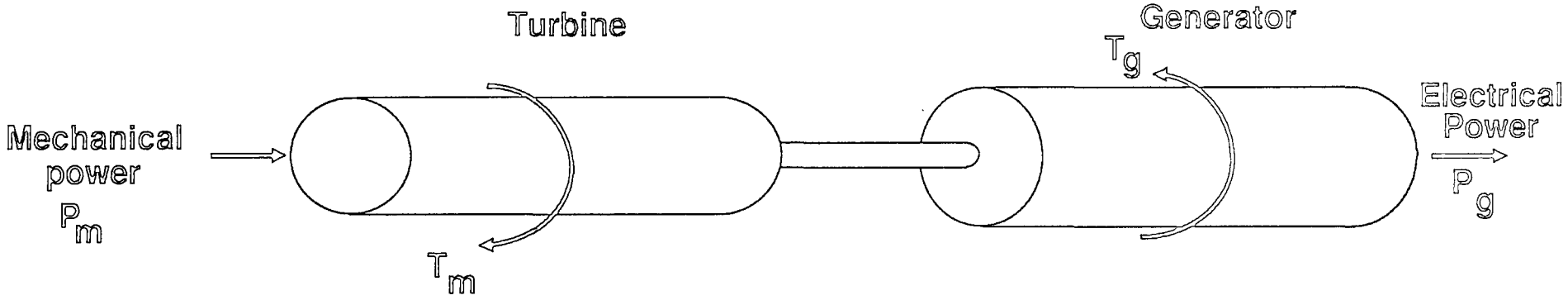


Fig. 4.3 Turbine/generator interaction

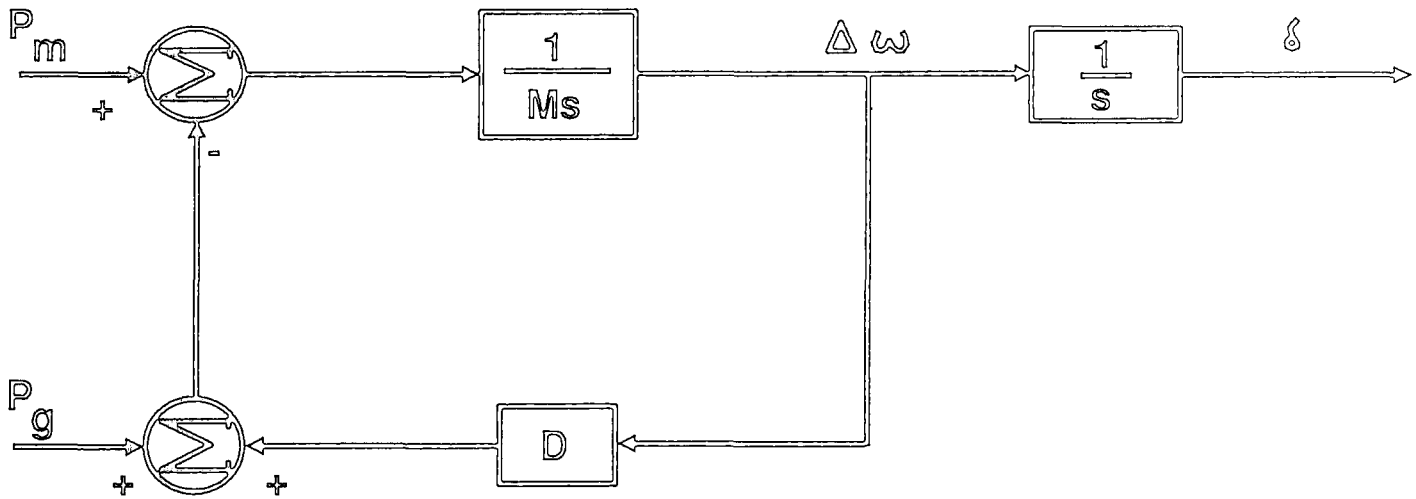


Fig. 4.4 Block diagram for a generator

$\omega_a = 2 * \pi * f_a$ is the average angular speed of the system or island in (electrical rad/s).

f_o is the synchronous frequency in (Hz).

The block diagram corresponding to the electromechanical interaction of the generator is illustrated by figure (4.4).

4.2.2 Models of the generating unit local controllers

A control system is joined to the generator, to reduce the effect of transient disturbances on power system efficiency. This restores the normal operating conditions rapidly. The contribution of the different controllers towards the stability of the system depends on their time constants. A small time constant implies a faster contribution to alleviate the disturbance. Since the load is constantly changing, this requires a continuous control of the generator.

The load frequency control provides the generating units with the most economical power set points and regulates the frequency of the system. The set points are telemetered from the control centre to the generating units.

4.2.2.1 Automatic Voltage Regulator (AVR)

The voltage regulator senses the variation of the terminal voltage, and acts accordingly on the excitation system. As a result the field current of the generator is adjusted. This in turn regulates the generator terminal voltage by affecting the stator internal voltage $\bar{E}_g$. The adopted excitor model is based on the widely used type 1 model proposed by the IEEE committee [62].

It should be noted that where the response is very fast, the component model should be revised to a simpler form for inclusion [15]. This reduces the stiffness problem, and maintains numerical stability [106]. Therefore, the transfer function representing the AVR has been simplified to its gain K_a, neglecting its time constant which is outside the range of the simulator time step. Also the exciter and feed back loop time constants are too small to be considered. Responses faster than the 1/5 Δt are usually not detected by the dynamic simulator [15]. Furthermore, the effect of saturation and the stabilisers are not

included. The resulting block diagram of the generator excitation system is shown in figure (4.5). Although this simplified model has no actual physical meaning, however, it is quite efficient for dynamic simulation.

The following equations correspond to figure (4.5).

$$\Delta V = V_{ref} - |V|_g \tag{4.21}$$

$$E_{fd} = K_a * \Delta V \tag{4.22}$$

$$\frac{d|E_g|}{dt} = \frac{(E_{fd} - |E_g|)}{T'_{do}} \tag{4.23}$$

Substituting ΔV and E_{fd} by their expressions in equation (4.23) leads to

$$\frac{d|E_g|}{dt} = \frac{K_a(V_{ref} - |V|_g) - |E_g|}{T'_{do}} \tag{4.24}$$

$$V_{R\,\min} \leq E_{fd} \leq V_{R\,\max} \tag{4.25}$$

The standard manner of dealing with the discontinuous functions, whenever the limits are exceeded, consists of substituting the limited quantities by their maximum or minimum values [73,106]. In such cases, expression (4.24) becomes

$$\frac{d|E_g|}{dt} = \frac{(V_R - |E_g|)}{T'_{do}} \tag{4.26}$$

Where

V_R is the regulator voltage in (p.u.). It can be either $V_{R\,\min}$ or $V_{R\,\max}$

$|E_g|$ is the open circuit terminal voltage magnitude in (p.u.).

E_{fd} is the field voltage in (p.u.).

T'_{do} is the open-circuit field direct axis transient time constant in (s).

V_{ref} is the reference voltage in (p.u.).

At steady state and under the initial conditions equation (4.24) becomes

$$0 = \frac{K_a(V_{ref} - |V_{go}|) - |E_{go}|}{T'_{do}} \tag{4.27}$$

$$V_{ref} = \frac{|E_{go}|}{K_a} + |V_{go}| \tag{4.28}$$

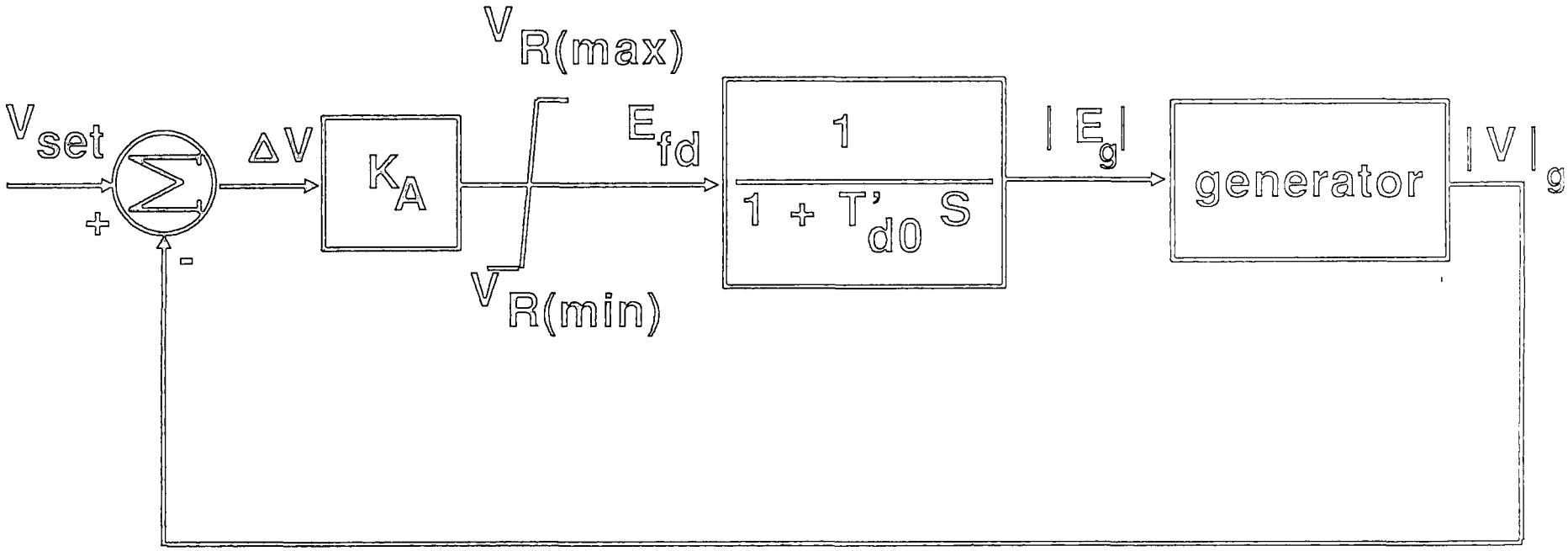


Fig. 4.5 AVR block diagram

4.2.2.2 Governor model

The changes in the generated electrical power are mainly caused by the variation of the load demand. This may cause the frequency of the system to reach unacceptable values. The main function of the governor is to sense the variation of the frequency and compare it against a reference value. If a mismatch is detected, the turbine inlet valves are adjusted so that the changes in the mechanical power compensate for the load variation. Consequently, the frequency is brought back to the normal conditions. The system chosen allows for the sharing of the load variation between the different generating units within the multi-machine system. The curves shown in figure (4.6) illustrate how the load variation is shared between two units according to the governor's gain. In this figure the gain is assumed to be the same for both governors. Therefore, the load variation is equally shared between the generating units.

The adopted governor model shown in figure (4.7) is similar to the one proposed by De Mello et al. [20].

The mathematical interpretation of this diagram is as follows

$$\Delta f = f - f_{set} \tag{4.29}$$

$$\Delta P = P_{set} - \frac{\Delta f}{R} \tag{4.30}$$

$$\frac{dP_{gv}}{dt} = \frac{(\Delta P - P_{gv})}{T_c} \tag{4.31}$$

$$P_{gv\,\min} \leq P_{gv} \leq P_{gv\,\max} \tag{4.32}$$

Where

R is the governor droop in (p.u.).

T_c is the governor time constant in (s).

P_{set} is the power set in (p.u.).

P_{gv} is the governor output in (p.u.).

The load frequency control provides the governor with frequency set points and power set points.

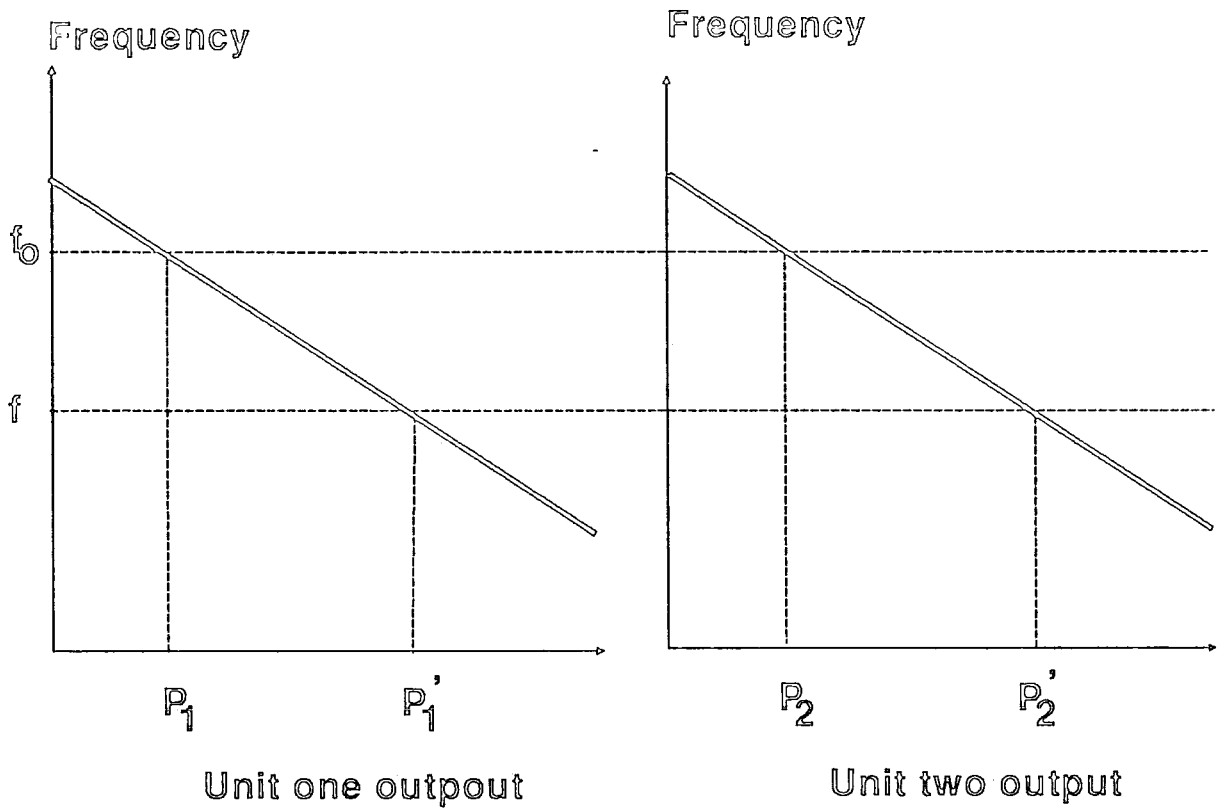


Fig. 4.6 Allocation of load variation to generators

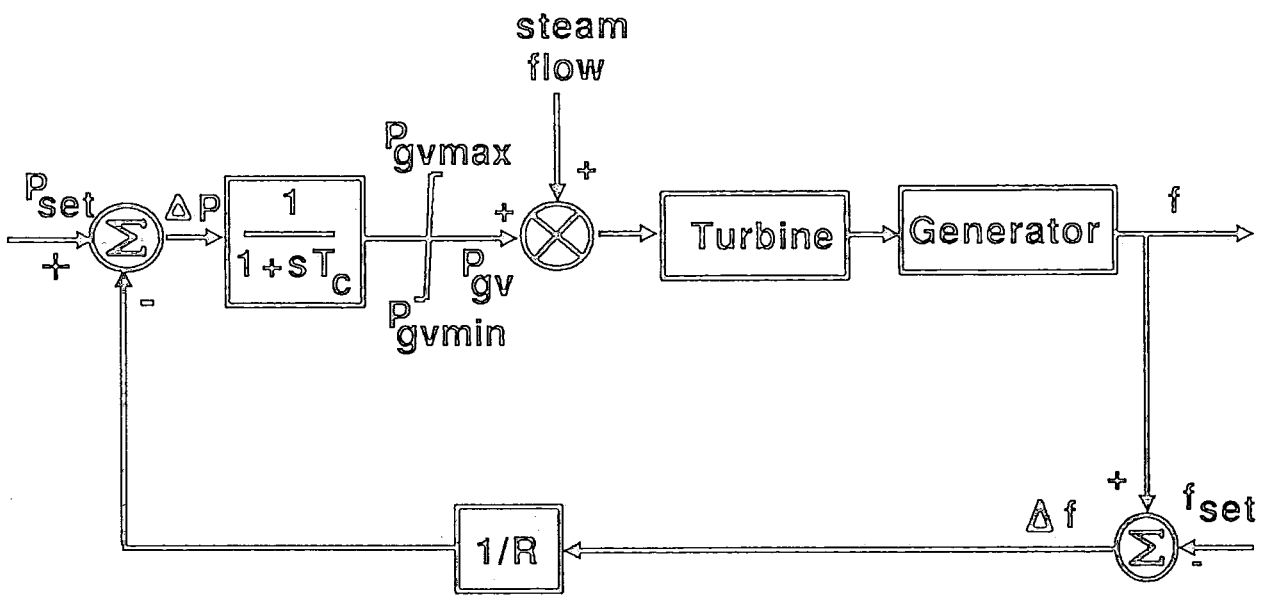


Fig. 4.7 Block diagram for the adopted governor

When the position valve attains its limits, P_{gv} can take either the maximum or the minimum value. In this case equation (4.31) becomes null. This can impose important constraints for short-term simulation [106].

4.2.2.3 Steam turbine model

The common steam turbine configurations together with their typical parameters are presented by the IEEE committee in reference 63. The adopted system is also widely used in the field of transient stability studies [3,20,110]. This model is a tandem compound single reheat system. Other models described in the literature are either too detailed or very approximated. Figure (4.8) shows the block diagram of the adopted steam turbine model.

This diagram shows that the governor allows a certain quantity of steam to flow through the different stages of the turbine. This leads to the adjustment of the mechanical power and therefore the correction of the frequency. The gradual response of the turbine is approximately represented by the different time constants. From this diagram the mechanical power can be expressed as follows

$$P_m = P_{hp} * F_{hp} + P_{ip} * F_{ip} + P_{lp} * F_{lp} \tag{4.33}$$

Where $F_{hp} + F_{ip} + F_{lp} = 1$

$$\frac{dP_{hp}}{dt} = \frac{(\Delta P - P_{hp})}{T_{ch}} \tag{4.34}$$

$$\Delta P = P_{gv} + K_1 \Delta(DP) \tag{4.35}$$

$$\frac{dP_{ip}}{dt} = \frac{(P_{hp} - P_{ip})}{T_{rh}} \tag{4.36}$$

$$\frac{dP_{lp}}{dt} = \frac{(P_{ip} - P_{lp})}{T_{co}} \tag{4.37}$$

Where

P_m is the mechanical power in (p.u.).

P_{hp} is the turbine high pressure in (p.u.).

P_{ip} is the turbine intermediate pressure in (p.u.).

P_{lp} is the turbine low pressure in (p.u.).

F_{hp} is the turbine high pressure factor in (p.u.).

F_{ip} is the turbine intermediate pressure factor in (p.u.).

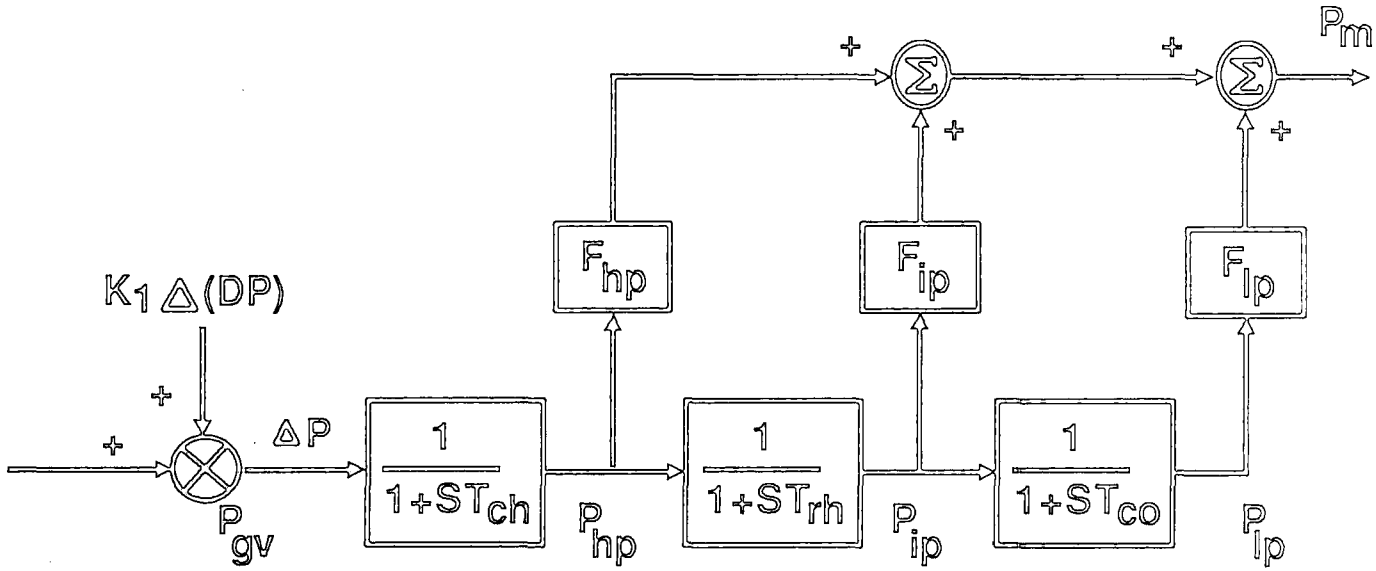


Fig. 4.8 Block diagram for the adopted steam turbine

F_{lp} is the turbine low pressure factor in (p.u.).

T_{ch} is the steam chest and high pressure cylinder time constant in (s).

T_{rh} is the reheater and the intermediate pressure cylinder time constant in (s).

T_{co} is the cross over and the low pressure cylinder time constant in (s).

K_1 and $\Delta(DP)$ are related to the boiler. They are defined in the following section.

4.2.2.4 Boiler model

The boiler is considered as the key link in the energy transformation process in a power station. From the chemical energy of the fuel it produces steam which is transformed into mechanical power by the turbine. The mechanical energy is changed into electrical power by the generator.

Chemical ⇒	Thermal ⇒	Mechanical ⇒	Electrical
energy	energy	energy	energy
(Fuel)	(Steam)	(Pm)	(Pg)

Due to its time delays and lags, the boiler does not contribute immediately after the occurrence of the disturbance to restore the normal operating conditions. Rather, it produces the required amount of steam after a few minutes. The demand change is initially supported by the storage capacity of the boiler.

Detailed representation of the boiler requires the inclusion of the non-linear models of a large number of its components. Such details are vital for studies such as design, but they are not desirable in the mid- and long-term simulation where hundreds of plants must be represented. Therefore, only those elements which have a great effect on the overall operating conditions of the network are considered. A survey of several low-order boiler models has been provided by Anderson in reference 3.

Three response modes of the boiler and the turbine to the power output variation are described in the literature [3,64]. The first mode is the turbine leading mode (boiler following) where the changes in generation are first detected

by the turbine and the boiler controls react accordingly. This mode is fast, but usually results in large deviation of the controlled variables, especially during severe disturbances. The second mode is the boiler leading (turbine following) mode, where the boiler senses the variation of the load and adjusts its variables. This adjustment is detected by the turbine, which changes the position of its valves to the new load level. This sort of control mode is safe but slow. A compromise between the desire for fast generation response and the desire for boiler safety led to a third option called the coordinated control mode.

The adopted model for this study is similar to the one suggested by De Mello et al. [20]. This model is relevant to coal fired plants. The advantage of such boilers is the relatively fast response to the disturbances ensured by the boiler's energy storage. Since the boiler controllers response is not fast enough to compensate for pressure variations caused by governor action, the stored energy enables the pressure at the turbine valve to be held constant without any control action during the initial seconds following a disturbance. The governor uses the frequency and power set points to determine the required valve position in order to allow the right amount of steam to flow through the different turbine cylinders. The power and frequency set points are also used as references by the boiler to generate the required quantity of steam, using the necessary amount of fuel and water.

Figure (4.9) illustrates the adopted boiler and its location within the loop of the generating unit's local controllers.

In order to model the boiler control system clearly, a further subdivision of its transfer functions is needed as shown in figure (4.10).

From this diagram equations representing the physical behaviour of the boiler can be derived.

$$A_1 = K_i * \Delta(FRS) \tag{4.38}$$

$$\frac{dA_2}{dt} = A_1 + T_b * \frac{dA_1}{dt} \tag{4.39}$$

$$\frac{d(\Delta(FI))}{dt} = \frac{(A_2 + T_a * \frac{dA_2}{dt} - \Delta(FI))}{T'_a} \tag{4.40}$$

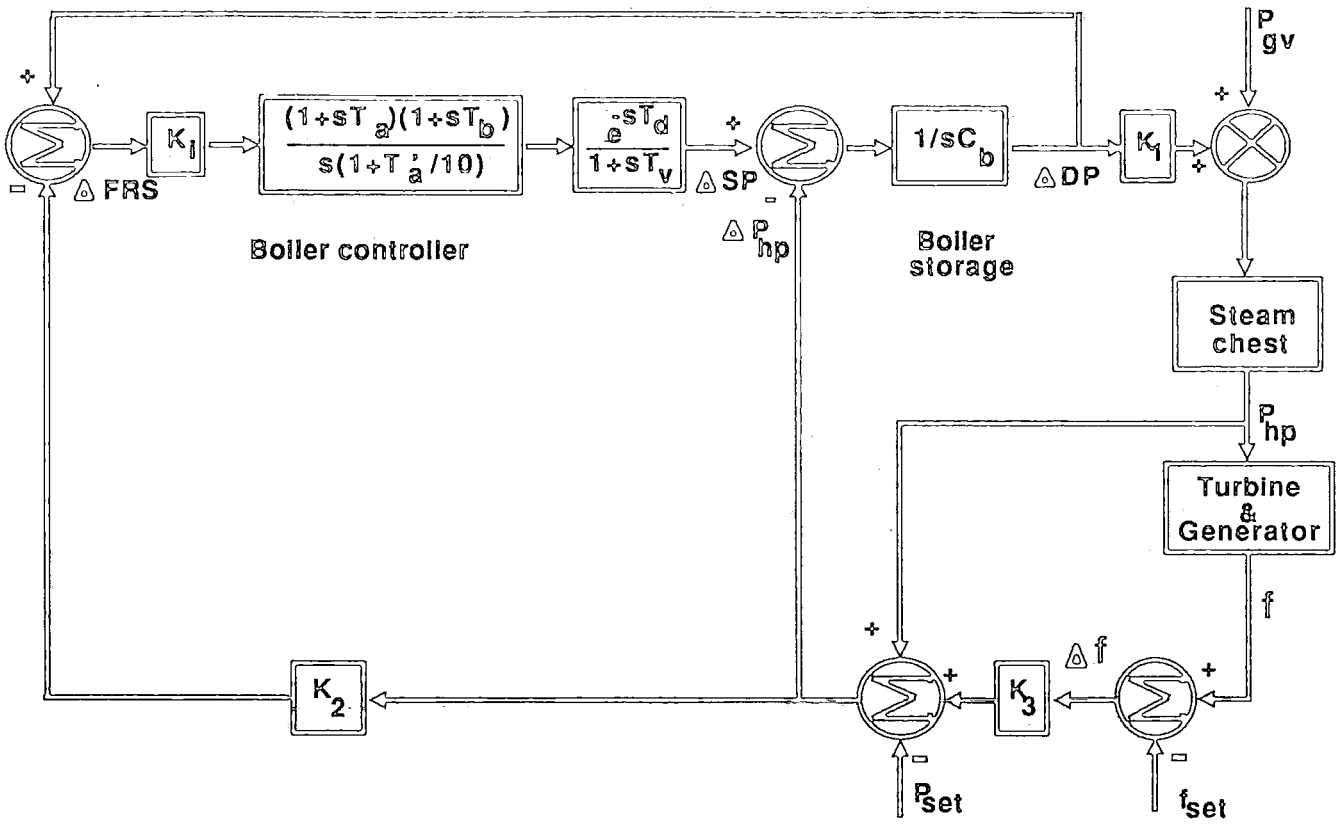


Fig. 4.9 Block diagram for the adopted boiler

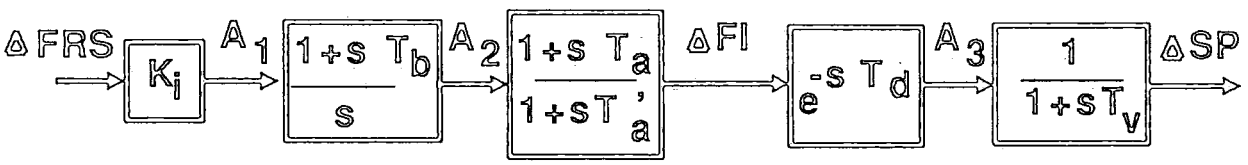


Fig. 4.10 Expansion of the boiler controller transfer functions

$$T'_a = \frac{T_a}{10}$$

$$A_3 = \Delta(FI) \tag{4.41}$$

$$\frac{d(\Delta(SP))}{dt} = \frac{(A_3 - \Delta(SP))}{T_v} \tag{4.42}$$

$$\frac{d(\Delta(DP))}{dt} = \frac{(\Delta(SP) - \Delta P_{hp})}{C_b} \tag{4.43}$$

$$\Delta P_{hp} = P_{hp} - (P_{set} - K_3 * \Delta f) \tag{4.44}$$

$$\Delta(FRS) = \Delta(DP) - K_2 * \Delta P_{hp} \tag{4.45}$$

Where

$\Delta(FI)$ is the variation of fire intensity in (p.u.).

$\Delta(SP)$ is the steam production variation in (p.u.).

$\Delta(DP)$ is the drum pressure variation in (p.u.).

$\Delta(FRS)$ is the firing rate setting variation in (p.u.).

A_1, A_2 and A_3 are auxiliary variables.

T_a and T_b are the combustion controller time constants in (s).

T_d is the fuel time delay in (s).

T_v is the fuel time constant in (s).

C_b is the boiler storage time constant in (s).

K_1 and K_i are amplifying factors.

K_2 is a factor denoting the pressure drop.

K_3 represents the effect of frequency on the boiler response.

4.3 ALGEBRAIC MODELS

A power system network includes the transmission system, compensators, generators, loads etc. The following models represent the static behaviour of these different elements. The dynamic components are linked to the rest of the network through interfacing variables such as current injections into nodes.

The subtransient and transient phenomena of the network are not included in this study because they are very fast events which do not last longer than a few milliseconds.

Algebraic models are based on Kirchhoff's first law expressed in equation (4.46) which is of the same form as the general network equation (4.2).

$$\sum_{i=1}^{n} \bar{I}_i = 0 \tag{4.46}$$

Where

n is the number of energised elements within a given node.

$\bar{I}_i$ is the current of the energised elements connected to a given node.

This current is expressed by Ohm's law as follows

$$\bar{I} = [\bar{Y}] * \bar{V} \tag{4.47}$$

Where

$[\bar{Y}]$ is the admittance.

$\bar{V}$ is the voltage.

4.3.1 Generator model

In order to model the interface between the generating units and the rest of the network, equations representing such operation must be of the same type, and they must be referred to the same reference frame as mentioned in section (4.2.1.1).

The algebraic equations representing the static behaviour of the synchronous generators is derived from figure (4.11 (a)).

By network analysis the injected current at the generator busbar i is given by

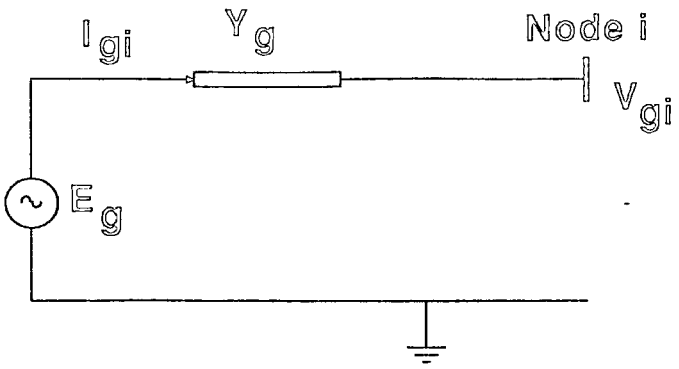
$$\bar{I}_{gi} = \bar{Y}_{gi} * (\bar{V}_{gi} - \bar{E}_{gi}) \tag{4.48}$$

Where $\bar{E}_g$ is expressed as follows in the network reference frame

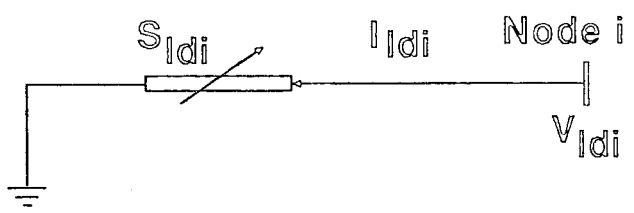
$$\bar{E}_g = |E_g| (\cos(\delta) + j \sin(\delta)) = E_{rg} + jE_{img} \tag{4.49}$$

and $\bar{Y}_g$ is given as follows

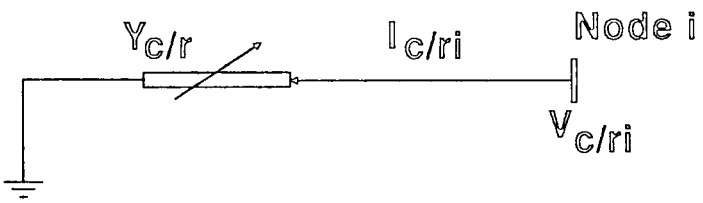
$$\bar{Y}_g = \frac{1}{(R_g + jX_d)} = \frac{R_g - jX_d}{|Z_g|^2} = G_g - jB_g \tag{4.50}$$



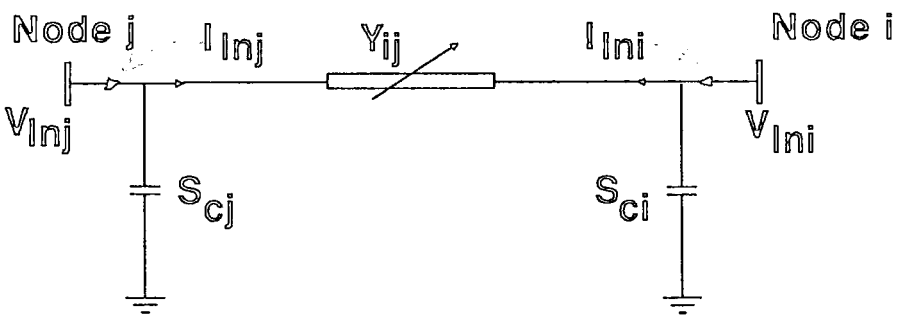
a) Generator model



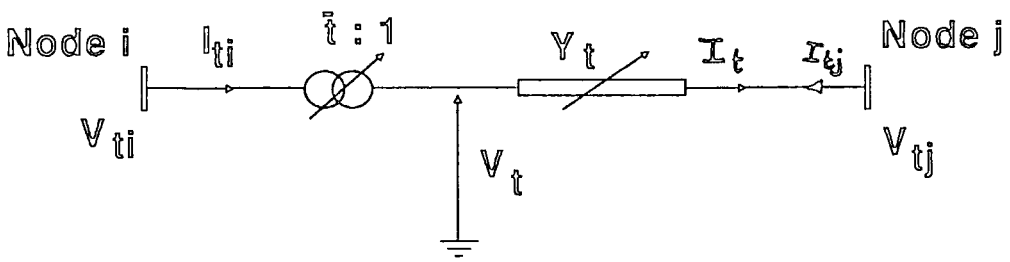
b) Load model



c) Capacitor/reactor model



d) Line model



e) Transformer model

Fig. 4.11 Network models

Substituting these two expressions into equation (4.48), and replacing the complex variable $\bar{V}_{gi}$ by its real and imaginary coordinates given in equation (4.14) will lead to the following expression of the real and imaginary coordinates of the current injected by the generator g at its busbar i.

$$\begin{bmatrix} I_r \\ I_{im} \end{bmatrix}_{gi} = \begin{bmatrix} G & B \\ -B & G \end{bmatrix}_{gi} \times \begin{bmatrix} V_r \\ V_{im} \end{bmatrix}_{gi} + |E_g|_i \times \begin{bmatrix} -G & -B \\ B & -G \end{bmatrix}_{gi} \times \begin{bmatrix} \cos(\delta) \\ \sin(\delta) \end{bmatrix}_{gi} \quad (4.51)$$

4.3.2 Load model

Early transient stability studies emphasised the importance of generators, with little attention to the loads. During the last two decades great efforts have been made to develop and improve the load models [6,17,83,88,95,99]. This is due to the realisation of the contribution of loads towards damping the inter-area oscillations together with the prime mover and load frequency control (the damper winding mainly affects the generating units local swings). Although the load parameters are obtained for small variations of frequency and voltage, and the dynamic behaviour of the induction motors is neglected, these models have greatly improved the simulation accuracy [88].

Most loads include diverse equipment of varying composition. Therefore, an equivalent model is necessary. Following the classification and the observation of the behaviour of the various loads, a second order polynomial expression relating the power demand to the voltage magnitude has been elaborated. This approach has been adopted by Concordia et al.[17] and Lopez [80]. This can be satisfactory if the frequency effect on the loads can be neglected.

Practically, the consideration of the frequency effect on the load is more difficult than the consideration of the effect of the voltage. This is due to the complexity of performing field tests, and interpreting the results. Also these tests are meant to be carried out at constant voltages, and this is practically not possible.

In reference 99 the authors have shown, through tests, that the effect of the voltage angle variation on the reactive load is negligible and its effect on the active load is of a transient nature. Consequently, this variable can be

neglected and only the voltage magnitude and the frequency at the load busbar is assumed to affect the load. The aggregate load in a substation is expressed by the following non-linear models

$$P_{ld} = \sum_{j=1}^{n} \left\{ P_{oj} * \frac{|V|}{|V_{oj}|}^{p_{vj}} * [1 + p_{fj} * \frac{(f_j - f_o)}{f_o}] \right\} \quad (a)$$
$$Q_{ld} = \sum_{j=1}^{n} \left\{ Q_{oj} * \frac{|V|}{|V_{oj}|}^{q_{vj}} * [1 + q_{fj} * \frac{(f_j - f_o)}{f_o}] \right\} \quad (b) \qquad (4.52)$$

Where n is the number of types of loads.

Although determining the composition of the aggregate load at each node is not an easy task, the overall coefficients p_v and q_v can be estimated under certain conditions such as small variations of the frequency and voltage. In this case the load ld at node i can be defined as follows, if $|V_{oi}|$ is assumed to be equal to 1 (p.u.) [95].

$$S_{ldi} = A_i * |V_{ldi}|^{p_{vi}} + jB_i * |V_{ldi}|^{q_{vi}} \qquad (4.53)$$

Where A_i and B_i are given by

$$A_i = P_{oi} * [1 + p_{fi} \frac{(f_i - f_o)}{f_o}] \quad (a)$$
$$B_i = Q_{oi} * [1 + q_{fi} \frac{(f_i - f_o)}{f_o}] \quad (b) \qquad (4.54)$$

Where

$$p_{fi} = \frac{dP_i}{df_i}; \qquad q_{fi} = \frac{dQ_i}{df_i}; \qquad p_{vi} = \frac{dP_i}{dV_i}; \qquad q_{vi} = \frac{dQ_i}{dV_i}$$

Typical values of the load coefficients p_{vi} and q_{vi} are 0 for a constant power, 1 for a constant current, and 2 for a constant impedance load [83].

It is assumed that the load exhibits a constant impedance characteristic whenever the voltage is below some predetermined value, typically 0.8 or 0.7 (p.u.) [4,17,73,93]. This approach is necessary due to the lack of data for larger voltage and frequency variations. This simplification is feasible since the motors behave as constant impedances at low voltages.

To include the load in the algebraic set of equations, the aggregate load is considered as a single component consuming power from busbar i as shown in figure (4.11 (b)). It should be noted that, due to the small time constants of most of the load components, loads are represented as static models. Although motors have non negligible time constants, their dynamics are not included in the system since they are not represented individually.

The injected current at the load busbar i is expressed as follows

$$\bar{I}^{*}_{ldi} = \frac{\bar{S}_{ldi}}{\bar{V}_{ldi}} = \frac{A_i * |V_{ldi}|^{p_{vi}} + jB_i * |V_{ldi}|^{q_{vi}}}{\bar{V}_{ldi}} \tag{4.55}$$

The matrix form of this equation for a load ld connected to a busbar i is as follows

$$\begin{bmatrix} I_r \\ I_{im} \end{bmatrix}_{ldi} = \begin{bmatrix} A * |V|^{pv-2} & B * |V|^{qv-2} \\ -B * |V|^{qv-2} & A * |V|^{pv-2} \end{bmatrix}_{ldi} \times \begin{bmatrix} V_r \\ V_{im} \end{bmatrix}_{ldi} \tag{4.56}$$

Where

S_{ldi} is the apparent power consumed by load ld at its busbar i in (p.u.).

P_{oi} is the load initial active power at busbar i in (p.u.).

Q_{oi} is the load initial reactive power at busbar i in (p.u.).

V_{oi} is the load initial voltage at busbar i in (p.u.).

p_{vi} is the sensitivity of the load active power with respect to voltage variation at busbar i.

q_{vi} is the sensitivity of the load reactive power with respect to voltage variation at busbar i.

p_{fi} is the sensitivity of the load active power with respect to frequency variation at busbar i.

q_{fi} is the sensitivity of the load reactive power with respect to frequency variation at busbar i.

4.3.3 Compensator models

The generator reactive power capability generally is enough to meet the required load reactive demand. However, since the generators are the main reactive source in the electrical systems it is desirable to keep this capability as a reserve to respond quickly to disturbances such as generator outages. Furthermore, when the load is heavy and remote from the generating plants the

reactive power losses become very important. Although the high voltage lines contribute significantly to the system shunt capacitance, this is not enough to meet such losses and support the voltage of the loads.

When the load is light reactive losses are relatively small. Together with the line charging effect this can cause the reactive power to exceed the VAr requirements, and the voltage to rise. This reactive power can be absorbed by generators, but it is preferable to preserve the generators absorbing capability for sudden contingencies.

Therefore, the installation of compensators has become indispensable to improve the load power factors and the voltage profile.

Synchronous compensators and series capacitors are used for very long overhead lines. The former are installed at very large industrial motor loads to improve their power factor. However, because they require frequent maintenance and they are liable to become unstable their use is not desirable [50]. Furthermore, since the loads are not modelled individually no motors are represented in the present simulator. The series capacitors are a means to reduce the line reactance and losses. These are boosting the voltage not through supplying reactive power but by reducing the losses.

Inductive series compensators are used for very short lines to increase their inductance. These can be included in the transmission line models [68]. The static VAr compensators (SVC) are used in circumstances where high speed response is required [70]. These devices can be included in this algorithm if the necessary data is available.

The shunt capacitors are the most relevant plant to this simulator. This is because their response time frame is within the time range of the present simulator, and they can overcome adequately the problem of low bus voltages at heavy load levels. Although these devices are not fast enough to improve system transient stability, because of their flexibility in regulating the voltage, they are the most commonly used form of compensation [39]. They can be

installed at or near the load sites. In the case of light load the excess of reactive power can be absorbed by shunt reactors. This leads to a reduction of the voltage. A simplified representation of these two elements is given in figure (4.11 (c)). These devices can be controlled remotely, automatically or manually. In this project they are operated manually by switching them on or off using a scenario file.

Transformer tap-changing is another means of regulating reactive power in the system through regulating the voltage level. This is described separately in section (4.3.5).

The shunt capacitor and reactor parameters are frequency dependent. This dependency is illustrated by the following equations

Capacitor modelling:

$$\bar{Y}_{ci} = jy_{ci} = \frac{jB_{oi} * f_i}{f_o} \tag{4.57}$$

$$Q_{ci} = y_{ci} * |V_{ci}|^2 \tag{4.58}$$

$$\bar{I}_{ci} = \bar{V}_{ci} * \bar{Y}_{ci} = (jy_c) * (V_r + jV_{im})_{ci} \tag{4.59}$$

The matrix form of a capacitor c linked to a busbar i is indicated below

$$\begin{bmatrix} I_r \\ I_{im} \end{bmatrix}_{ci} = \begin{bmatrix} 0 & -y \\ y & 0 \end{bmatrix}_{ci} \times \begin{bmatrix} V_r \\ V_{im} \end{bmatrix}_{ci} \tag{4.60}$$

Reactor modelling:

$$\bar{Y}_{ri} = -jy_{ri} = \frac{-jf_o}{(X_{oi} * f_i)} \tag{4.61}$$

$$Q_{ri} = y_{ri} * |V_{ri}|^2 \tag{4.62}$$

$$\bar{I}_{ri} = \bar{Y}_{ri} * \bar{V}_{ri} \tag{4.63}$$

The matrix form of this equation at the busbar i is as follows

$$\begin{bmatrix} I_r \\ I_{im} \end{bmatrix}_{ri} = \begin{bmatrix} 0 & y \\ -y & 0 \end{bmatrix}_{ri} \times \begin{bmatrix} V_r \\ V_{im} \end{bmatrix}_{ri} \tag{4.64}$$

Where

$\bar{Y}_{c/r}$ is the capacitor/reactor admittance in (p.u.).

$Q_{(c/r)}$ is the reactive power of the capacitor/reactor in (p.u.).

$I_{r(ci/ri)}$ is the real current injected at the capacitor/reactor busbar i in (p.u.).

$I_{im(ci/ri)}$ is the imaginary current injected at the capacitor/reactor busbar i in (p.u.).

B_o is the capacitor's susceptance at nominal frequency in (p.u.).

X_o is the reactor's reactance at nominal frequency in (p.u.).

f_i and f_o are the frequency at busbar i and the normal frequency in (Hz).

4.3.4 Transmission line model

The transient state of the transmission lines is not included in this study. It has been a common practice to model the lines as a conventional Π circuit at steady state. This modelling is carried out for a balanced representation only. The single line circuit of a transmission line is given in figure (4.11 (d)).

The line reactance and susceptance are frequency dependent. In order to model this effect properly the frequencies at the sending and receiving end of the line are averaged to give a more accurate model.

The capacitances C_i and C_j at the two ends of a line are generally the same. Therefore the susceptances S_{ci} and S_{cj} can be expressed as follows

$$S_{ci} = S_{cj} = \frac{S_o * \omega_{ij}}{\omega_o} = S_c \tag{4.65}$$

$$S_{oi} = S_{oj} = C_i * \omega_o = S_o \tag{4.66}$$

$$\bar{Y}_{ci} = \bar{Y}_{cj} = jS_c \tag{4.67}$$

$$\bar{Y}_{ij} = \frac{1}{R_{ij} + jX_{ij}} = G_{ij} - jB_{ij} \tag{4.68}$$

By network analysis the current at busbars i and j can be expressed as follows

$$\bar{I}_{lni} = \bar{Y}_{ci} * \bar{V}_{lni} + \bar{Y}_{ij} * (\bar{V}_{lni} - \bar{V}_{lnj}) \tag{4.69}$$

In the same way the components of $\bar{I}_{lnj}$ are given by

$$\bar{I}_{lnj} = \bar{Y}_{cj} * \bar{V}_{lnj} + \bar{Y}_{ji} * (\bar{V}_{lnj} - \bar{V}_{lni}) \tag{4.70}$$

It should be noted that for a line $\bar{Y}_{ij} = \bar{Y}_{ji}$. The matrix form of a line ln connected between nodes i and j is

$$\begin{bmatrix} I_{ri} \\ I_{imi} \\ I_{rj} \\ I_{imj} \end{bmatrix}_{ln} = \begin{bmatrix} G_{ij} & -S_c + B_{ij} & -G_{ij} & -B_{ij} \\ S_c - B_{ij} & G_{ij} & B_{ij} & -G_{ij} \\ -G_{ij} & -B_{ij} & G_{ij} & -S_c + B_{ij} \\ B_{ij} & -G_{ij} & S_c - B_{ij} & G_{ij} \end{bmatrix}_{ln} \times \begin{bmatrix} V_{ri} \\ V_{imi} \\ V_{rj} \\ V_{imj} \end{bmatrix}_{ln} \tag{4.71}$$

Where

$i = 1, 2 \ldots$ and $j = 1, 2, \ldots$ with $i \neq j$.

R_{ij} is the line ln resistance in (p.u.)

X_{ij} is the line ln reactance in (p.u.).

$C_{i/j}$ is the ln charging at the two ends i and j in (p.u.).

G_{ij} is the line ln conductance in (p.u.).

B_{ij} is the line ln series susceptance in (p.u.).

$S_{c(i/j)}$ are the line ln sending and receiving end shunt susceptances at the angular speed $\omega(i/j)$ in (p.u.).

$S_{o(i/j)}$ are the line ln sending and receiving end susceptances at the nominal angular speed ω_o in (p.u.).

ω_{ij} is the average angular speed in (rad/s).

$V_{r(i/j)}$ and $V_{im(i/j)}$ are the voltage coordinates at busbars i and j joined by line ln in (p.u.).

$I_{rln(i/j)}$ and $I_{imln(i/j)}$ are the current coordinates at busbars i and j in (p.u.).

4.3.5 Transformer model

The voltage, active and reactive power are mainly controlled locally at the generating units. However, at the remote busbars, due to losses in the transmission network, these variables need to be readjusted. This can be done by various equipment such as compensators and transformers. The latter are equipped with tap-changing and phase-shifting systems. These controllers sense

the voltage and active power respectively, compare them with their corresponding set points, and initiate tap-changing and phase-shifting accordingly. Tap-changing is a means for controlling the voltage and the corresponding reactive power. Whereas the phase-shifting is a device which controls the phase angle and the corresponding active power [57]. In this case reactive and active power are controlled indirectly through controlling the voltage and the phase angle respectively.

To simplify the transformer representation, the conventional approximations have been assumed. Since the exciting current is very small compared to the rated current, it can be neglected without affecting much the accuracy of the model. Therefore, the transformer is simplified to its leakage impedance and an ideal transformer in series. Since both the tap-changing and phase-shifting are considered, the turn ratio is complex. The schematic diagram of a single line model of the transformer is illustrated in figure (4.11 (e)). The variables involved are referred to the sending busbar i.

$$\bar{V}_t = \bar{t} * \bar{V}_{ti} \tag{4.72}$$

By network analysis we have

$$\bar{I}_{tj} = \bar{Y}_t * (\bar{V}_{tj} - \bar{V}_t) \tag{4.73}$$

Substituting $\bar{V}_t$ by its expression gives

$$\bar{I}_{tj} = \bar{Y}_t * (\bar{V}_{tj} - \bar{t} * \bar{V}_{ti}) \tag{4.74}$$

By energy conservation and network analysis we have

$$\frac{-\bar{I}_{tj}}{\bar{I}_{ti}} = \frac{1}{\bar{t}^*} \tag{4.75}$$

This implies that

$$\bar{I}_{ti} = -\bar{I}_{tj} * \bar{t}^* \tag{4.76}$$

Replacing $\bar{I}_{tj}$ by its expression in this equation leads to

$$\bar{I}_{ti} = \bar{Y}_t * (\bar{t}^* * \bar{t} * \bar{V}_{ti} - \bar{t}^* * \bar{V}_{tj}) \tag{4.77}$$

Where * denotes the conjugate and

$$\bar{t}^* * \bar{t} = |t|^2 \tag{4.78}$$

The expression for $\bar{I}_{ti}$ becomes

$$\bar{I}_{ti} = \bar{Y}_t * (|t|^2 * \bar{V}_{ti} - \bar{t}^* * \bar{V}_{tj}) \tag{4.79}$$

$$\bar{Y}_t = \frac{1}{R_t + jX_t} = G_t - jB_t \tag{4.80}$$

$$\bar{t} = |t|(\cos(\phi) + j\sin(\phi)) = t_r + jt_{im} \tag{4.81}$$

Replacing all the variables by their coordinates in $\bar{I}_{ti}$ and $\bar{I}_{tj}$ and putting the equations in matrix form we obtain the following expression

$$\begin{bmatrix} I_{ri} \\ I_{imi} \\ I_{rj} \\ I_{imj} \end{bmatrix}_t = \begin{bmatrix} |t|^2G & |t|^2B & -t_rG + t_iB & -t_iG - t_rB \\ -|t|^2B & |t|^2G & t_iG + t_rB & -t_rG + t_iB \\ -t_rG - t_iB & t_iG - t_rB & G & B \\ -t_iG + t_rB & -t_rG - t_iB & -B & G \end{bmatrix}_t \times \begin{bmatrix} V_{ri} \\ V_{imi} \\ V_{rj} \\ V_{imj} \end{bmatrix}_t \tag{4.82}$$

Where $i = 1, 2 \ldots$ and $j = 1, 2 \ldots$ with $i \neq j$

The tap-changing and phase-shifting are updated whenever the voltage at the controlled busbars and the power flow is different to the set points. Mescua [81] has suggested the following iterative way of adjusting the tap-changing and phase-shifting

$$\begin{aligned} |t_k| &= |t_{k-1}| + a * \Delta V \qquad (a) \\ \phi_k &= \phi_{k-1} + b * \Delta P \qquad (b) \end{aligned} \tag{4.83}$$

$$\Delta V = |V_i| - V_{set} \tag{4.84}$$

$$\Delta P = P_i - P_{set} \tag{4.85}$$

Where

$|t|_{(k/k-1)}$ is the turn ratio magnitude at iteration k and $k-1$ respectively.
$\phi_{(k/k-1)}$ is the turn ratio phase angle at iteration k and $k-1$ respectively.
$\bar{V}_i$ and P_i are the voltage and the active power at the controlled busbar in (p.u.).
V_{set} and P_{set} are the voltage and the active power set points respectively in (p.u.).

a and b are accelerating factors.

Since tap and phase steps are discrete and cannot be incremented by other than one step at a time, they are incremented or decremented by one step whenever the controlled voltage and power flow deviate from the target values. This can be achieved manually, by remote control or automatically. In the case of remote control operation, a target step position is sent from the control centre to the transformer operating equipment which varies the tap-changing and phase-shifting accordingly. In the case of automatic control the variation of the bus voltage and power flow are detected and the necessary actions take place. The response of the control equipment of transformers to control signals is not instantaneous. It is in the range of a few seconds. This restricts the effect of tap-changing and phase-shifting during fast disturbances.

4.4 ALGEBRAISATION OF THE DIFFERENTIAL EQUATIONS

The simultaneous solution of the dynamic and static subsystems requires the algebraisation of the former. This can be achieved by utilising the implicit trapezoidal method. Therefore, the differential equations are transformed to take a similar form to the algebraic equation given in expression (4.2).

In this equation the transpose of the dynamic state variables vector $[Y]_i$ and the static state variables vector $[X]_i$ are given as follows

$$[Y]^t_i = (\omega_i, \delta_i, |E_g|_i, P_{gvi}, P_{hpi}, P_{ipi}, P_{lpi}, \Delta DP_i) \tag{4.86}$$

$$[X]^t_i = (Vri, Vimi) \tag{4.87}$$

The algebraisation of equation (4.37) for instance can be derived from equation (3.1).

$$P_{lp_t} = P_{lp_{(t-\Delta t)}} + \frac{\Delta t}{2}(\dot{P}_{lp_t} + \dot{P}_{lp_{(t-\Delta t)}}) \tag{4.88}$$

Substituting $\dot{P}_{lp}$ by expression (4.37) gives

$$P_{lp_t} = P_{lp_{(t-\Delta t)}} + \frac{\Delta t}{2T_{co}}(P_{ip_t} - P_{lp_t} + P_{ip_{(t-\Delta t)}} - P_{lp_{(t-\Delta t)}}) \tag{4.89}$$

$$P_{lp_t}(1+\frac{\Delta t}{2T_{co}}) - P_{ip_t}(\frac{\Delta t}{2T_{co}}) = P_{lp_{(t-\Delta t)}}(1-\frac{\Delta t}{2T_{co}}) + P_{ip_{(t-\Delta t)}}(\frac{\Delta t}{2T_{co}}) \quad (4.90)$$

This equation can be rewritten in a general form as follows

$$h(P_{ip}, P_{lp}) = h(P_{ip}, P_{lp})_t + h(P_{ip}, P_{lp})_{(t-\Delta t)} + C = 0 \quad (4.91)$$

Where

$$h(P_{ip}, P_{lp})_t = P_{lp_t}(1+\frac{\Delta t}{2T_{co}}) - P_{ip_t}(\frac{\Delta t}{2T_{co}}) \quad (4.92)$$

$$h(P_{ip}, P_{lp})_{(t-\Delta t)} = P_{lp_{(t-\Delta t)}}(1-\frac{\Delta t}{2T_{co}}) + P_{ip_{(t-\Delta t)}}(\frac{\Delta t}{2T_{co}}) \quad (4.93)$$

$$C = 0$$

The same process can be applied to the rest of the dynamic equations as shown in Appendix A.

Equations representing the boiler are not algebraised by the trapezoidal method. A more approximate technique is used to solve these equations because their time constants are of the order of minutes. Therefore, it is adequate to solve them once for each time step rather than solving them at each iteration. This reduces the number of differential equations to be solved and the order of the Jacobian matrix without significantly affecting the overall accuracy [93]. This method can be described by the following simple expression

$$[\dot{Y}] = \frac{([Y]_t - [Y]_{(t-\Delta t)})}{\Delta t} \quad (4.94)$$

4.5 FORMATION OF THE JACOBIAN MATRIX

The set of algebraic equations described in the previous sections have to be solved to obtain the unknown variables at each time step. The Newton-Raphson (N-R) method which is a widely used technique in the field of power system stability studies and load-flow solutions [24,106] is adopted.

As described in chapter 3 the N-R method consists of solving iteratively the following set of equations

$$-[J]_k^{-1} \times \begin{bmatrix} [h([Y]^t,[X]^t)] \\ [g([Y]^t,[X]^t)] \end{bmatrix}_k = \begin{bmatrix} [\Delta Y] \\ [\Delta X] \end{bmatrix}_k \tag{4.95}$$

Where

Function $[h([Y]^t,[X]^t)]$ represents the set of the algebraised dynamic functions. An example of these functions is the one related to the turbine low pressure given in the previous section for a subset s of state variables $[Y]_s{}^t = (P_{ip}, P_{lp})$.

The set of linear functions $[g([Y]^t,[X]^t)]$ is given by the sum of the currents at each node i as expressed in equation (4.46) in section (4.3).

The $N * N$ Jacobian matrix $[J]$ is composed of the partial derivatives of the dynamic equations $[h([Y]^t,[X]^t)]$ with respect to $[Y]$ and $[X]$, and the derivatives of the static equation $[g([Y]^t,[X]^t)]$ with respect to $[Y]$ and $[X]$.

The order m of the vector $[\Delta Y]$ and $[h([Y]^t,[X]^t)]$ is given by $m = N_d * N_g$.

The order m' of the vector $[\Delta X]$ and $[g([Y]^t,[X]^t)]$ is given by $m' = N_n$.

With N_d the number of differential equations for each generator, N_g the number of active generators in the system, and N_n the number of active nodes in the network. The order N of the Jacobian matrix satisfies this relation $N = m + m'$.

4.5.1 Dynamic sub-Jacobian

This sub-Jacobian includes the partial derivatives of the generator dynamic equations with respect to the dynamic state variables given by the general expressions $[\frac{d[h([Y]^t,[X]^t)]}{d[Y]}]$ in equation (3.6).

As an example, the Jacobian elements related to equation (4.92) are given below

$$\frac{d(h(P_{ip}, P_{lp}))}{d(P_{lp})} = 1 + \frac{\Delta t}{2T_{co}} \tag{4.96}$$

$$\frac{d(h(P_{ip}, P_{lp}))}{d(P_{ip})} = -\frac{\Delta t}{2T_{co}} \tag{4.97}$$

Where

equation (4.96) is a diagonal element of the Jacobian matrix and
equation (4.97) is an off-diagonal element of the same matrix.

These elements represent the 7^{th} row of the Jacobian matrix given in equation (4.104) where equation $h(P_{ip}, P_{lp})$ is renamed h_7. The rest of the non zero elements of the dynamic sub-Jacobian are given in appendix A.

4.5.2 Interfacing sub-Jacobian

This represents the dependence of the dynamic models on the network state variables (voltages) and the dependence of the static models on the dynamic state variables. These are represented by the two sets of derivatives $[\frac{d[h([Y]^t,[X]^t)]}{d[X]}]$ and $[\frac{d[g([Y]^t,[X]^t)]}{d[Y]}]$ respectively.

The Jacobian elements related to the dynamic/static and static/dynamic interface are developed in appendix A.

4.5.3 Algebraic sub-Jacobian

The algebraic sub-Jacobian is formed by the set of derivatives of the static equations with respect to the network state variables $[\frac{d[g([Y]^t,[X]^t)]}{d[X]}]$. This expression includes the derivatives of the injected current of the different components of the network such as generators, loads, shunts, transformers, and lines with respect to the corresponding real and imaginary voltages.

Theoretically a node can include all network elements. These can be linked to different sections of busbars within the substations. Since the frequency is assumed to be constant during a time step, the derivatives of frequency dependent elements with respect to the frequency are not included in the Jacobian matrix. The voltage dependent static equations are all represented by their admittances in the Jacobian matrix because the current varies linearly with the voltage except for the loads which are represented by non-linear equations.

4.5.3.1 Line elements

Differentiating the line injected currents given in equation (4.71) by their corresponding real and imaginary voltages results in the following expression

$$\begin{bmatrix} \frac{dI_{ri}}{dV_{ri}} & \frac{dI_{ri}}{dV_{imi}} & \frac{dI_{ri}}{dV_{rj}} & \frac{dI_{ri}}{dV_{imj}} \\ \frac{dI_{imi}}{dV_{ri}} & \frac{dI_{imi}}{dV_{imi}} & \frac{dI_{imi}}{dV_{rj}} & \frac{dI_{imi}}{dV_{imj}} \\ \frac{dI_{rj}}{dV_{ri}} & \frac{dI_{rj}}{dV_{imi}} & \frac{dI_{rj}}{dV_{rj}} & \frac{dI_{rj}}{dV_{imj}} \\ \frac{dI_{imj}}{dV_{ri}} & \frac{dI_{imj}}{dV_{imi}} & \frac{dI_{imj}}{dV_{rj}} & \frac{dI_{imj}}{dV_{imj}} \end{bmatrix}_{ln} = \begin{bmatrix} G_{ij} & -S_c + B_{ij} & -G_{ij} & -B_{ij} \\ S_c - B_{ij} & G_{ij} & B_{ij} & -G_{ij} \\ -G_{ij} & -B_{ij} & G_{ij} & -S_c + B_{ij} \\ B_{ij} & -G_{ij} & S_c - B_{ij} & G_{ij} \end{bmatrix}_{ln} \tag{4.98}$$

4.5.3.2 Transformer elements

Transformer models are similar to those of lines, except for the consideration of the complex turn ratio. The derivatives of the current at the sending and receiving end of the transformers with respect to the corresponding voltages are deduced from equation (4.82)

$$\begin{bmatrix} \frac{dI_{ri}}{dV_{ri}} & \frac{dI_{ri}}{dV_{imi}} & \frac{dI_{ri}}{dV_{rj}} & \frac{dI_{ri}}{dV_{imj}} \\ \frac{dI_{imi}}{dV_{ri}} & \frac{dI_{imi}}{dV_{imi}} & \frac{dI_{imi}}{dV_{rj}} & \frac{dI_{imi}}{dV_{imj}} \\ \frac{dI_{rj}}{dV_{ri}} & \frac{dI_{rj}}{dV_{imi}} & \frac{dI_{rj}}{dV_{rj}} & \frac{dI_{rj}}{dV_{imj}} \\ \frac{dI_{imj}}{dV_{ri}} & \frac{dI_{imj}}{dV_{imi}} & \frac{dI_{imj}}{dV_{rj}} & \frac{dI_{imj}}{dV_{imj}} \end{bmatrix}_t = \begin{bmatrix} |t|^2 G & |t|^2 B & a & b \\ -|t|^2 B & |t|^2 G & -b & a \\ c & d & G & B \\ -d & c & -B & G \end{bmatrix}_t \tag{4.99}$$

Where a, b, c, d are expressed as follows

$a = -t_r G + t_i B; \quad b = -t_i G - t_r B; \quad c = -t_r G - t_i B; \quad d = t_i G - t_r B$

4.5.3.3 Generator static elements

The static elements of the generator included in the network sub-Jacobian are derived from equation (4.51).

The diagonal block of elements is given by

$$\begin{bmatrix} \frac{dI_r}{dV_r} & \frac{dI_r}{dV_{im}} \\ \frac{dI_{im}}{dV_r} & \frac{dI_{im}}{dV_{im}} \end{bmatrix}_{gi} = \begin{bmatrix} G & B \\ -B & G \end{bmatrix}_{gi} \tag{4.100}$$

4.5.3.4 Load elements

The loads are modelled as non-linear frequency and voltage dependent equations. The derivatives of the injected current at the load busbars with respect to their voltages are added to the diagonal elements of the nodes to which the loads ld are connected. These derivatives are based on equation (4.56).

$$\begin{bmatrix} \frac{dI_r}{dV_r} \\ \frac{dI_r}{dV_{im}} \\ \frac{dI_{im}}{dV_r} \\ \frac{dI_{im}}{dV_{im}} \end{bmatrix}_{ldi} = \begin{bmatrix} M & N & 0 \\ 0 & M & N \\ -N & M & 0 \\ 0 & -N & M \end{bmatrix}_{ldi} \times \begin{bmatrix} {V_r}^2 \\ V_r V_{im} \\ {V_{im}}^2 \end{bmatrix}_{ldi} + \begin{bmatrix} A|V|^{pv-2} \\ B|V|^{qv-2} \\ -B|V|^{qv-2} \\ A|V|^{pv-2} \end{bmatrix}_{ldi} \tag{4.101}$$

Where M and N are

$$M = A(p_v - 2)\frac{|V|^{pv-2}}{|V|^2} \quad \text{and} \quad N = B(q_v - 2)\frac{|V|^{qv-2}}{|V|^2}$$

It should be noted that this model is a revision of the one presented in reference 93.

4.5.3.5 Compensator elements

The compensator elements added to the diagonal block of elements of the static sub-Jacobian are derived from equation (4.60) and (4.64). These elements are represented by the shunt capacitor and reactor admittances respectively.

Equations representing shunt capacitors are given below

$$\begin{bmatrix} \frac{dI_r}{dV_r} & \frac{dI_r}{dV_{im}} \\ \frac{dI_{im}}{dV_r} & \frac{dI_{im}}{dV_{im}} \end{bmatrix}_{ci} = \begin{bmatrix} 0 & -y \\ y & 0 \end{bmatrix}_{ci} \tag{4.102}$$

The shunt reactor elements are

$$\begin{bmatrix} \frac{dI_r}{dV_r} & \frac{dI_r}{dV_{im}} \\ \frac{dI_{im}}{dV_r} & \frac{dI_{im}}{dV_{im}} \end{bmatrix}_{ri} = \begin{bmatrix} 0 & y \\ -y & 0 \end{bmatrix}_{ri} \tag{4.103}$$

Equation (4.104) shows the lay out of the Jacobian matrix. The different derivatives in this matrix are defined by equations (4.96 and 4.97) and (A.1-A.77). The two diagonal sub-Jacobians (block I and IV) represent the dynamic and the network elements respectively. The two off-diagonal sub-Jacobians (block II and III) include the interfacing elements between the generators and the rest of the network and vice-versa. This matrix represents the details of generator one linked to node i.

4.6 POWER FLOW CALCULATION

The transmission lines are the means for transporting electrical power from the power stations where it is generated to the remotely sited loads where it is consumed. These lines are subject to many constraints and limitations. In order to function properly without affecting the security and reliability of the system, certain restrictions such as limitations of the power flow through the lines are imposed.

4.6.1 Line power flow

By considering figure (4.11(d)) and equation (4.71) the power flow at node i and j is obtainable. The apparent powers $\bar{S}_i$ and $\bar{S}_j$ are given by

$$\bar{S}_i = \bar{V}_i * \bar{I}_i^* = V_i(I_{ri} - jI_{imi}) = P_i + jQ_i$$

$$\bar{S}_j = \bar{V}_j * \bar{I}_j^* = V_j(I_{rj} - jI_{imj}) = P_j + jQ_j$$

Replacing $\bar{V}_{i/j}$ by their real and imaginary components, and substituting $I_{ri/j}$ and $I_{imi/j}$ by their expressions given in equation (4.71), the active and

$$
\begin{array}{c}
\\
\\
\\
\\
\\
I, II \\
\\
\\
\\
\\
\\
\\
\\
\\
\\
\\
\\
\\
III, IV \\
\\
\\
\\
\\
\end{array}
\left(\begin{array}{ccccccccccc|ccccccc}
\frac{dh_1}{d\omega} & \frac{dh_1}{d\delta} & \frac{dh_1}{d|E_g|} & 0 & \frac{dh_1}{dP_{hp}} & \frac{dh_1}{dP_{ip}} & \frac{dh_1}{dP_{lp}} & 0 & \dots & 0 & \dots & \dots & \frac{dh_1}{dV_{ri}} & \frac{dh_1}{dV_{imi}} & \dots & 0 & \dots & 0 \\
\frac{dh_2}{d\omega} & \frac{dh_2}{d\delta} & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \dots & \dots & 0 & 0 & & & & \vdots \\
0 & 0 & \frac{dh_3}{d|E_g|} & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \dots & \dots & \frac{dh_3}{dV_{ri}} & \frac{dh_3}{dV_{imi}} & & & & \vdots \\
\frac{dh_4}{d\omega} & 0 & 0 & \frac{dh_4}{dP_{gv}} & 0 & 0 & 0 & 0 & \dots & 0 & \dots & \dots & 0 & 0 & & & & \vdots \\
0 & 0 & 0 & \frac{dh_5}{dP_{gv}} & \frac{dh_5}{dP_{hp}} & 0 & 0 & \frac{dh_5}{d(\Delta DP)} & \dots & 0 & \dots & \dots & 0 & 0 & & 0 & & \vdots \\
0 & 0 & 0 & 0 & \frac{dh_6}{dP_{hp}} & \frac{dh_6}{dP_{ip}} & 0 & 0 & \dots & 0 & \dots & \dots & 0 & 0 & & & & \vdots \\
0 & 0 & 0 & 0 & 0 & \frac{dh_7}{dP_{ip}} & \frac{dh_7}{dP_{lp}} & 0 & \dots & 0 & \dots & \dots & 0 & 0 & & & & \vdots \\
\frac{dh_8}{d\omega} & 0 & 0 & 0 & \frac{dh_8}{dP_{hp}} & 0 & 0 & \frac{dh_8}{d(\Delta DP)} & \dots & 0 & \dots & \dots & 0 & 0 & & & & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & & & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & Gns. & & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & & \ddots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
- & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & \frac{dI_{rig}}{d\delta} & \frac{dI_{rig}}{d|E_g|} & 0 & 0 & 0 & 0 & 0 & \vdots & \vdots & \vdots & \dots & \frac{dI_{ri}}{dV_{ri}} & \frac{dI_{ri}}{dV_{imi}} & \dots & \frac{dI_{ri}}{dV_{rj}} & \frac{dI_{ri}}{dV_{imj}} & \dots \\
0 & \frac{dI_{imig}}{d\delta} & \frac{dI_{imig}}{d|E_g|} & 0 & 0 & 0 & 0 & 0 & \vdots & \vdots & \vdots & \dots & \frac{dI_{imi}}{dV_{ri}} & \frac{dI_{imi}}{dV_{imi}} & \dots & \frac{dI_{imi}}{dV_{rj}} & \frac{dI_{imi}}{dV_{imj}} & \dots \\
\vdots & & & & & & & & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \ddots & \dots & \dots & \dots \\
\vdots & & & & 0 & & & & \vdots & \vdots & \vdots & \dots & \frac{dI_{rj}}{dV_{ri}} & \frac{dI_{rj}}{dV_{imi}} & \dots & \frac{dI_{rj}}{dV_{rj}} & \frac{dI_{rj}}{dV_{imj}} & \dots \\
\vdots & & & & & & & & \vdots & \vdots & \vdots & \dots & \frac{dI_{imj}}{dV_{ri}} & \frac{dI_{imj}}{dV_{imi}} & \dots & \frac{dI_{imj}}{dV_{rj}} & \frac{dI_{imj}}{dV_{imj}} & \dots \\
0 & & & & & & & & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}\right)
$$

(4.104)

reactive power at node i and j can be expressed as follows

$$P_i = V_{ri}(G_{ij}V_{rij} + B_{ij}V_{imij}) + V_{imi}(G_{ij}V_{imij} - B_{ij}V_{rij}) \qquad (a)$$
$$Q_i = -S_c|V_i|^2 + V_{ri}(B_{ij}V_{rij} - G_{ij}V_{imij}) + V_{imi}(G_{ij}V_{rij} + B_{ij}V_{imij}) \qquad (b)$$

(4.105)

$$P_j = V_{rj}(G_{ij}V_{rji} + B_{ij}V_{imji}) + V_{imj}(G_{ij}V_{imji} - B_{ij}V_{rji}) \qquad (a)$$
$$Q_j = -S_c|V_j|^2 + V_{rj}(B_{ij}V_{rji} - G_{ij}V_{imji}) + V_{imj}(G_{ij}V_{rji} + B_{ij}V_{imji}) \qquad (b)$$

(4.106)

where

$$V_{rij} = V_{ri} - V_{rj}$$
$$V_{imij} = V_{imi} - V_{imj}$$
$$V_{rji} = V_{rj} - V_{ri}$$
$$V_{imji} = V_{imj} - V_{imi}$$

4.6.2 Transformer power flow

Apparent power at node i and j linked through a transformer is given by the following equations

$$\bar{S}_i = \bar{V}_i * \bar{I}_i^* = \bar{V}_i(I_{ri} - jI_{imi}) = P_i + jQ_i$$

$$\bar{S}_j = \bar{V}_j * \bar{I}_j^* = \bar{V}_j(I_{rj} - jI_{imj}) = P_j + jQ_j$$

Replacing the currents at node i and node j by their components given in equation (4.82), and substituting the voltages at nodes i and j by their real and imaginary components, would result in the following expressions of active and reactive power at the two ends.

$$P_i = I_{t1i} * V_{ri} - I_{t2i} * V_{imi}$$
$$Q_i = I_{t1i} * V_{imi} + I_{t2i} * V_{ri} \qquad (a)$$
$$P_j = I_{t1j} * V_{rj} - I_{t2j} * V_{imj}$$
$$Q_j = I_{t1j} * V_{imj} + I_{t2j} * V_{rj} \qquad (b)$$

(4.107)

Where

$$I_{t1i} = G_t * (|t|^2 * V_{ri} - t_r * V_{rj} - t_i * V_{imj}) + B_t * (|t|^2 * V_{imi} + t_i * V_{rj} - t_r * V_{imj})$$
$$I_{t2i} = B_t * (|t|^2 * V_{ri} - t_r * V_{rj} - t_i * V_{imj}) - G_t * (|t|^2 * V_{imi} - t_r * V_{imj} + t_i * V_{rj})$$
$$I_{t1j} = G_t * [V_{rj} - (t_r * V_{ri} - t_i * V_{imi})] + B_t * [V_{imj} - (t_i * V_{ri} + t_r * V_{imi})]$$
$$I_{t2j} = -G_t * [V_{imj} - (t_i * V_{ri} - t_r * V_{imi})] + B_t * (V_{rj} - (t_r * V_{ri} - t_i * V_{imi})]$$

4.7 SIMULATION OF PROTECTION

Most of protection relays operate very quickly. Therefore, these fast acting operations can only be simulated as an instantaneous action by the dynamic simulator. This simplified protection includes: the generator over-speed and under-speed protection, the load under-frequency protection, and the transmission line overload protection.

4.7.1 Generator over-speed protection

When certain system events occur, a mismatch between the demand and the generation of energy takes place causing a variation of the generators rotating speed. If this variation is not critical the governors can respond efficiently by acting upon the turbine valves. However, if the over-speed reaches a critical limit (about 10% of the nominal value [34]), the valves are shut instantaneously before any structural damage of the generating units takes place.

4.7.2 Generator under-speed protection

If the load under-frequency protection (described below) fails to react when the frequency decreases to undesirable values, when a critical limit is reached the generator under-speed protection disconnects the generator.

4.7.3 Load under-frequency protection

When a system or an island is overloaded, the frequency of the system starts falling until a new load/generation balance is reached. If the generators cannot operate safely at this level, they have to trip. These control actions are undesirable especially in the case of small islands because this may lead to a blackout of that subsystem. The load shedding relays have set points which would allow them to react before the generators critical state is reached. The

frequency can be used as a measure to assess the amount of the load to be shed [87]. Therefore the under-frequency relays have graded frequency settings. On the other hand the relay settings must also consider the load priority factors. The loads which tolerate low frequencies, and require a relatively permanent supply are the last to be disconnected, whereas the loads which are very sensitive to the frequency deviation are set to be disconnected first. The action of the simulated under-frequency relays is instantaneous. Whenever the predetermined frequency limit is reached the breaker at the load busbar is immediately switched off. Usually, the lowest frequency deviation is of the order of -5% of the nominal value. However, if the load can stand a frequency out of this range, it can be supplied until the frequency gets to the corresponding critical limit.

4.7.4 Line overload protection

The adopted overload line protection uses the inverse-time relays. These relays provide adjustable operating conditions with a time margin. As their name indicates, they exhibit inverse characteristics of the time versus the current or power flow [33]. Actually these devices are used for over-current protection during short circuit disturbances. But, they can be used for both over-current and overload protection if the appropriate current and time settings are selected [33].

The principle of this protection is to accumulate the integrated error for a certain time. If the fault persists and the accumulated error reaches a critical value, then the corresponding breaker is switched off, thus disconnecting the overloaded line.

4.8 EXACT TOPOLOGY DETERMINATION

This program is one of the supporting software packages of the simulator. The relationship between these two algorithms is based on transferring the state of the different switches whenever a physical change of the network occurs. This switching can be caused either purposefully by sending commands to open or close any of the breakers, or can happen as a result of clearing a disturbance.

In both cases switch states are altered and a flag is updated in the simulator program. This is detected by the topology program which would update the physical configuration of the network accordingly. When the simulator receives the new structure of the system, it includes the energised plants again and it dismisses the disconnected ones from the calculation.

4.9 ISLANDING AND SYNCHRONISATION

Certain severe disturbances may cause the network to split into several independent islands. The topology analysis program is capable of detecting this phenomena immediately, and communicating the new configuration of the system to the simulator and the control package. These are robust enough to deal with such events without any numerical instability or inaccuracies.

Islanding is an undesirable phenomena because it often leads to an energy deficit within the independent islands. This can also lead to a general blackout, or to a smaller extent, a blackout of some of the islands. As a result of the mismatch between the demand and the generation, an imbalance of the frequencies within the different islands occurs. In order to re-connect the different islands a re-synchronisation must take place. The reduction of the difference between the island frequencies can be achieved either, by shedding a certain amount of the load, if the frequency is below the nominal value, or by decreasing the power supply if the frequency is above the frequency set point. Further, the voltage magnitudes and phase angles at both ends of the tie lines must be of the same order. If these conditions are satisfied, the different islands can be recombined into an overall system. The topology analysis program is able to detect this event and inform the control system.

4.10 STARTING UP AND SHUTTING DOWN GENERATORS

Unit commitment is the package which decides whether the different generators in the system are to function normally, run-down to be disconnected, or started-up if they are to be re-connected. These decisions are sent to the dispatch program where new power targets are set. If a disconnection or a

connection of a generator is required, its output power target is set to its minimum level. The load frequency control ramps down the output of the generator to be disconnected to its new target in a predetermined target time. When the minimum value is reached, the simulator reduces the power to zero and switches off the generator breaker. In the case of starting up the generator, first the simulator has to re-synchronise the generator and increase its power to its minimum. Then the load frequency control has the task of ramping up this generator's output to meet the requirements of the system before a target time is elapsed.

4.11 TELEMETRY AND MEASUREMENTS

When the network solution is obtained, the updated data is telemetered to the control package as is the case in real life operational control. The real data is usually corrupt because of transducer and data communication errors. To account for this phenomena two types of errors are simulated. The first are the static errors which are mainly caused by the transducers. The second type are the dynamic errors which are related to the noise and other factors which affect the data in their communication channel. Since the value of these errors cannot be determined analytically, a random error theory is used. This is based on random number generation [77].

The random number together with the transducer precision P result in the following expression for the static error for a given full scale reading.

$$E_s = P_s * RN * V_{max} \tag{4.108}$$

Dynamic errors are varying with the time. Whenever a measurement is required the random error is evaluated. These errors are also dependent on percentage errors and the maximum scale of the measured value.

$$E_{dt} = P_d * RN_t * V_{max} \tag{4.109}$$

Where

E_s is the static error.

E_{dt} is the dynamic error at time t.

P_s is the static percentage error for the measured quantity.

P_d is the dynamic percentage error for the measured quantity.

V_{max} the maximum value on the scale of the transducer.

RN is the random number.

Using the calculated values at time t and the two equations given above the measured values can be modelled as follows

$$V_{meast} = V_{calt} + E_{dt} + E_s \tag{4.110}$$

Where

V_{meast} is the measured value at time t.

V_{calt} is the calculated variable at time t.

In addition to this analogue bad data simulation, gross errors are added manually (in the scenario file) to the measured variables if required. Furthermore, a digital bad data simulation has been modelled. This simulates erroneous switch states.

This chapter and the previous one presented the selected models and methods in order to build up a fast dynamic simulator. All the requirements, conditions and simplifications are discussed. The next chapter presents the different steps in developing a dynamic simulator on digital computers. The method previously used to solve the power system set of linear equations is described.

CHAPTER 5

CONVENTIONAL SIMULATOR STRUCTURE

5.1 INTRODUCTION

This chapter describes the use of the original algorithm (based on the *Harwell* library routines) which has been implemented within the simulator for solving the linear set of equations. The structure of the time cycle and the iterative process involved in the dynamic simulation are also highlighted. The interaction between the simulator and its environment is included. This consists of performing the measurements and telemetry in the simulator and receiving the updated quantities of the system such as the topology, loading conditions, set points etc.

The general structure of the simulator and the sequence in which the tasks mentioned above are carried out are shown in figure (5.1). Figure (5.1) shows the overall structure of the centralised simulator implemented on the Perkin Elmer PE 3230 minicomputer. In the case of the parallel simulator which runs on the Array Processor FPS 5205 hosted by the Perkin Elmer PE 3230 these tasks are subdivided between the two machines following their capabilities.

5.2 SIMULATOR INITIALISATION

The simulator program is initialised by acquiring the necessary static and dynamic data from the various simulator support packages such as the physical data, the loader, the exact topology determination, the load-flow, the control signals etc. These have to run before the simulator starts, and the initial data has to be loaded into different common blocks which can be accessed by the

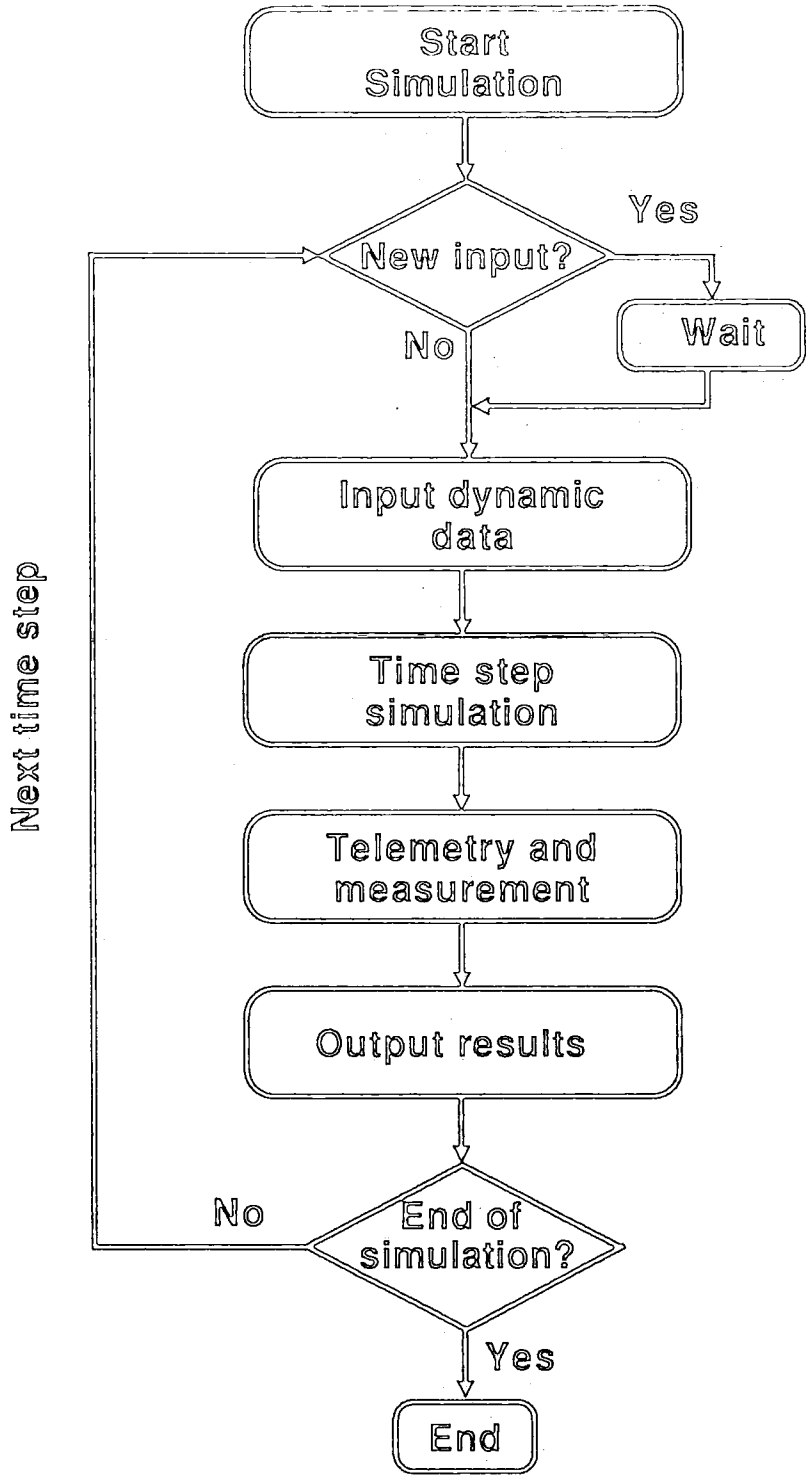


Fig. 5.1 The simulator configuration

simulator and periodically updated by these tasks. For instance, the loader updates its variables every 10 seconds, and the control signals are updated every 5 seconds. The topology is updated whenever a scenario command is received or the protection has acted on faulted elements. The simulator updates its values usually every second, although larger or smaller time steps can be chosen.

The initial node voltage magnitudes and phase angles are provided by the load-flow package together with the initial power set points and generating unit active and reactive powers. The loading conditions are determined by the loader. The initial structure of the network is given by the exact topology determination package.

After receiving the initial data the simulator evaluates the steady state initial operating conditions. Knowing the network parameters and structure, the initial set points and node voltages, and the loading conditions and generation levels for each load and generator, it is possible to evaluate the local initial operating state of the simulator. Since it is very important to avoid any unnecessary calculation, some of the local static values are calculated once at this stage to avoid repeated operations.

Initially the transformer turn ratios are set according to their tap-changing and phase-shifting initial positions. Knowing the complex terminal voltage of each machine and its apparent power, as well as its parameters, it is possible to evaluate the initial voltage behind the synchronous reactance and the rotor angle by performing a simple calculation. The initial generator terminal voltage and voltage behind the synchronous reactance can be used to evaluate the initial AVR set point as shown in equation (4.28). This value is normally set by the control package, however, since the initial voltage behind the synchronous reactance is not available in the control package, the AVR reference voltage V_{ref} is calculated initially by the simulator. The initial mechanical power and turbine quantities are set equal to the electrical power, and the variation of the boiler variables are set to zero.

5.3 THE TIME STEP LOOP

After initialising the simulator the dynamic time cycle can be performed. The different steps included in this loop are illustrated in the flow chart given in figure (5.2). This process starts by receiving the dynamic data of the previous time step and from the simulator support programs. According to the latest topology state, unit commitment, and load frequency control requirements the system dependent and independent variables are initialised for the new time step. In the case of islanding, the simulator is capable of assigning these variables to their areas and treating these islands as self-contained networks over the islanding period without losing stability.

At some point in the time step a decision point is introduced that determines when convergence is obtained and enough iterations have been performed. During this iterative loop (inner loop) the Jacobian matrix is updated following the latest values of the state variables. Usually the system converges after one or two iterations at steady state and it does not take more than four or five iterations under severe transients. When the system converges and the state variables are evaluated the simulation of the time step carries on by updating the dependent variables such as the active and reactive power of the generators, loads, shunts, the frequency at each active busbar and the average system or island frequency, the loading conditions for the next time step, and so on.

The slow varying differential equations of the boiler are also evaluated at this stage using a simple rectangular integration formula given by equation (4.94).

By knowing the power flow and the frequencies the protection can be initiated. A tripping takes place if any of the controlled variables are unacceptable.

The re-synchronisation of independent islands and generator start-up and shut-down are also included in the time cycle and activated whenever this is

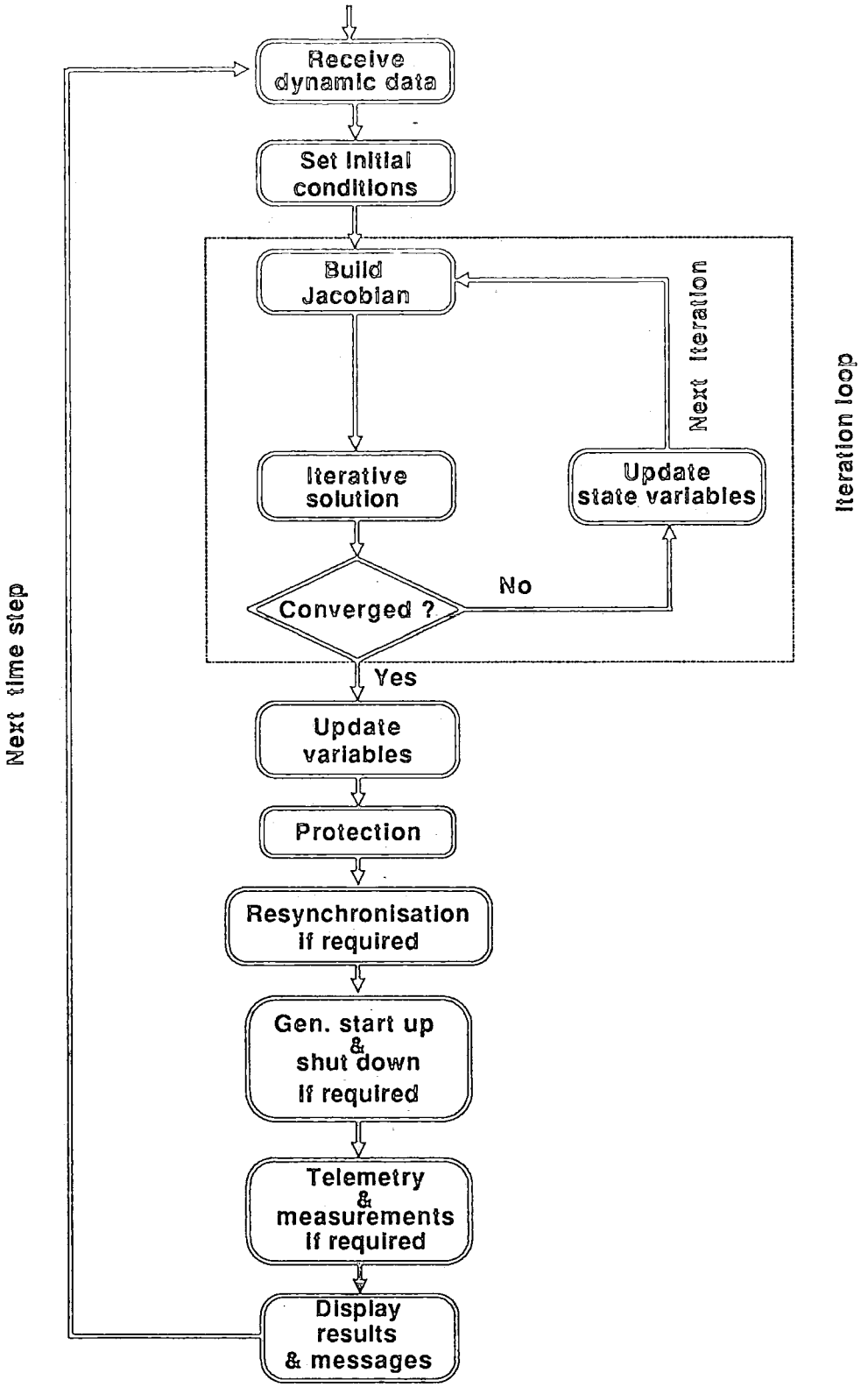


Fig. 5.2 The simulator time loop

required.

The last step in this external loop is the evaluation of the measurement and telemetry if this is required by the control package.

It should be noted that all inactive elements of the power system are dismissed from this calculation until they are re-energised again. This prevents the simulator from performing unnecessary operations.

5.3.1 The iteration loop

This loop starts with the construction of the Jacobian matrix, and the solution of the resulting set of equations, if necessary, and the state variable updating, together with the production of error messages if any abnormal conditions such as the divergence of the Newton-Raphson system, violation of the voltage limits etc. occur.

The above mentioned steps can be added to the simplified iteration loop given in figure (5.2) to illustrate the sequence in which the process of iteratively evaluating the state variables takes place (figure (5.3)).

5.3.1.1 Formation of the Jacobian matrix

The Newton-Raphson method and the theory of the formation of the Jacobian matrix are described in detail in chapters 3 and 4. The equations of the different non zero elements of the sparse Jacobian matrix are illustrated in section (4.5) and appendix A. The present subsection emphasises the practical aspects of the problem and gives details of the computer implementation.

Using equations (4.96-4.103), and (A.1-A.77) the Jacobian matrix non zero elements are determined together with their locations. Equation (4.104) illustrates the form of the Jacobian matrix, its non zero elements and their positions within the matrix. For the sake of clarity this equation represents the details of one generator only and two nodes, however, it can be easily generalised for the whole system. This matrix is subdivided into four blocks. The two diagonal blocks (I,IV) represent the dynamic and algebraic diagonal sub-Jacobians,

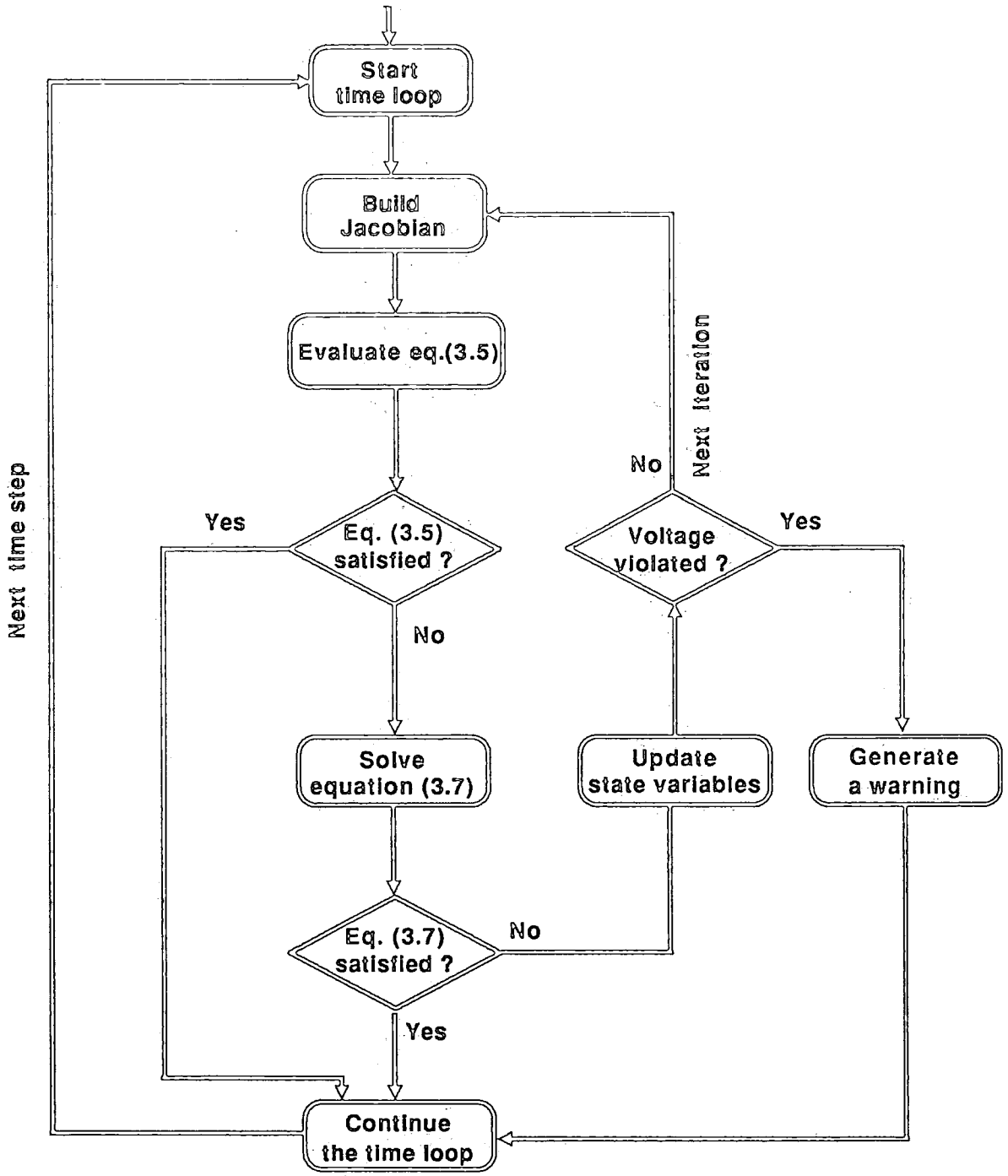


Fig. 5.3 The simulator iterative loop

the off-diagonal elements of block (*II*) represent the dynamic/algebraic interface and the off-diagonal sub-Jacobian of block (*III*) represents algebraic/dynamic interface between the generator and the network respectively.

Since only four elements are included in each of the off-diagonal blocks (*II* and *III*) these sub-Jacobians are very sparse. The limited number of these equations gives only $8 * N_g$ non zero elements. The dimension of these second and third blocks is given by $(N_d * N_g) * n$ and $n * (N_d * N_g)$ which is equal to $48 * 30$ and $30 * 48$ for the 30 node network.

Where

$N_d = 8$ is the number of the differential equations included in the Jacobian matrix.

N_g is the number of generators.

n is the number of nodes.

This shows that the interfacing sub-matrices will be even more sparse for larger networks.

All the active generators are positioned in sequence. Since there is no interface between the generators, the dynamic sub-Jacobian of each generator is in the form of an independent diagonal sub-matrix. Each of the dynamic diagonal sub-Jacobians contains only 21 non zero elements. The number of non zero elements of the global dynamic sub-Jacobian is $21 * N_g$ out of the total number of elements given in this equation $N_{dg} = (N_d * N_g)^2$.

The sparsity of the static sub-Jacobian depends directly on the line connectivity and this is usually not very dense. Generally a node is not connected to more than four or five lines. The number of components other than lines and transformers linked to a single node does not affect the sparsity of the system. The number of static non zero elements of a system can be expressed by $N_l * 2 + n$ where N_l is the number of lines. The complex representation of the network equations necessitates $4 * (N_l * 2 + n)$ elements.

The sparsity of the 30 node, 118 node, and 141 node systems is given in table (5.1)

n	N_g	N_l	No. of non zeros	Total elements	Sparsity (%)
30	6	41	622	11664	5.33
118	19	179	2455	150544	1.63
141	12	204	2544	142884	1.78

Table 5.1 Sparsity of the different test networks

Thus enormous storage space is saved by storing the non zero elements only in a compact vector instead of storing the whole matrix as a two dimensional array. The matrix is stored row-wise. The storage includes the non zero element values as well as their addresses and row and column numbers. Other integer variables such as the number of non zero elements in each column are also stored.

The Jacobian matrix requires all the static and dynamic equations to be dependent on the state variables or constants. Therefore, all dependent variables are replaced by their expressions which are functions of the independent variables. Furthermore, these equations must be referred to the same reference frame, as mentioned in chapter 4, so that their interface will be modelled correctly.

The non zero elements are updated iteratively since some of them depend on the state variables which are re-calculated at the end of each iteration.

A) Generator dynamic sub-Jacobian

Each generating unit is represented by 14 equations, where only 8 are included in the Jacobian matrix. The rest of these equations are the slow varying models of the boiler which do not need frequent solution, as stated in chapter 4. These are evaluated at the end of each time step using the rectangular integration method. This approach has the advantage of reducing the number of non zero elements of the Jacobian matrix, and the number of state variables without sacrificing the computational performance and accuracy

since the time constants of these models allow for the consideration of these dynamic variables as constants within a time step.

Some of the dynamic variables are subject to constraints and discontinuities if limiters are modelled. The AVR limits are dealt with by setting the field voltage at its limit whenever this value is attained. The Jacobian elements related to the limited variable are zero since in this case the variable is set to its limit which is constant. The governor limits were not included in this version.

The dynamic state variables are ordered arbitrarily. The Jacobian elements related to the dynamic plant are represented by the partial derivatives of the algebraised functions of each model with respect to the state variables they depend upon. Most of the generating unit models are linear and are related to only one of the state variables, except the non-linear AVR and swing models. The swing equation is related to all generator and network state variables as illustrated in equation (4.104). The position of these elements is deduced from this equation and depend on the state variables order.

Due to the linearity of most of the dynamic equations, their related non zero elements are constants. These elements ought to be assigned to their corresponding positions in the overall Jacobian matrix once. However, since in certain cases retrieving data from the memory may be more time consuming than assigning a value to a variable, this process is carried out repeatedly to preserve the integrity of the computing code. Furthermore, if the devices corresponding to these terms are disconnected from the system they should be dismissed from the Jacobian matrix. The evaluation of the linear function given by equation (3.5) is obtained simply by multiplying the different Jacobian elements by the relevant state variables. However, to satisfy this equation in the case of non-linear models a further manipulation of the right hand side member of this equation is necessary.

B) Network sub-Jacobian

This sub-matrix includes the static non-linear generator and load equations, the transmission line, transformer, and the compensator models. This sub-Jacobian is represented by the different element admittances except for the non-impedance loads. These are modelled as constant impedances only under certain circumstances.

All network Jacobian elements are obtained by differentiating the complex components of the current at each node with respect to the complex bus voltage thus forming sets of $2 * 2$ sub-matrices. Some elements of these sub-matrices such as the load sub-Jacobians are voltage dependent. Since this state variable is altered at each iteration the iterative updating of these elements is necessary. The generator algebraic (diagonal) elements are constant. Other elements such as those related to the lines and shunts depend on the bus frequency which is evaluated once at the end of each time step. These elements could be updated for each cycle.

The nodal self admittance is obtained by adding all the incident line and transformer admittances at each node including load, generator and compensator sub-Jacobians. The mutual admittances include the transmission lines and transformers only since these are the only means which link two nodes. The position of these elements depends on the node in which their busbars are located. This can be expressed as follows

$$row = N_d * N_g + node(I) * 2 \tag{5.1}$$

for the diagonal elements where the rows and columns are equal, and

$$row = N_d * N_g + snode(J) * 2$$

$$column = N_d * N_g + rnode(J) * 2 \tag{5.2}$$

for the off-diagonal elements.

Where

$node(I)$ is the node which includes busbar I (any of the plants considered can be linked to any busbar I within the node),

$snode(J)$ is the sending node of line J,
$rnode(J)$ is the receiving node of line J.

a) Line and transformer programming

The transformers and lines are represented by the conventional Π circuit. The lines are considered as transformers with a turn ratio equal to 1. The charging elements for transformers are usually neglected.

The tap-changing and phase-shifting are activated whenever the voltage set points and power set points are not satisfied. A time delay is included in the model to simulate the time taken by the controlling devices to respond to the variation of the bus voltage and power flow. If the transformer tap-changing and phase-shifting are controlled manually, the operator increments or decrements the number of steps whenever the controlled bus voltage and power flow limits are exceeded. This is simulated by accessing the common block and changing the values of the corresponding variables.

The Jacobian elements related to these two plant items are represented by the summation of all admittances of the lines and/or transformers linked to the sending and receiving nodes including their charging. The charging of a line is equally divided between its two ends. These non zero elements are given by equations (4.98) and (4.99). Their position is given by expressions (5.1) and (5.2).

b) Load programming

Since dynamic models of the loads are not included in this study, they are represented as an aggregate non-linear static model linking the power consumed to the voltage at the busbars. This model does not depend on state variables other than the voltage. Therefore, they are contributing to the formation of the diagonal static sub-Jacobian only. Their sub-Jacobian elements given by equation (4.101) are added to the rest of the non zero elements at their nodes.

Some of the loads are modelled as constant impedances ($p_v = q_v = 2$). In this case equation (4.56) representing the load current at node i becomes:

$$\begin{bmatrix} I_r \\ I_{im} \end{bmatrix}_{ldi} = \begin{bmatrix} A & B \\ -B & A \end{bmatrix}_{ldi} \times \begin{bmatrix} V_r \\ V_{im} \end{bmatrix}_{ldi} \tag{5.3}$$

and the Jacobian elements of this equation are:

$$\begin{bmatrix} \frac{dI_r}{dV_r} & \frac{dI_r}{dV_{im}} \\ \frac{dI_{im}}{dV_r} & \frac{dI_{im}}{dV_{im}} \end{bmatrix}_{ldi} = \begin{bmatrix} A & B \\ -B & A \end{bmatrix}_{ldi}$$

A and B are defined in equation (4.54)

Strictly speaking these elements are not constant since they are frequency dependent. Their values are considered constant during a time step, but they vary from one step to another with the variation of the bus frequency.

The load is also represented by this simplified model if the voltage decreases below 0.8 (p.u.). This is because motors usually stall if the voltage is around this value and can be considered as an impedance. In a discussion of reference 99 the adoption of this measure is justified by the fact that below these limits the load-flow is less likely to converge. Therefore, this is adopted to allow the simulation to run even under severe conditions.

The load model does not satisfy equation (3.5) because it is not linear. Therefore, extra manipulation of the right hand side member of this equation is necessary.

The model given in chapter 4 is a modification of the one which is included in the standard OCEPS simulator. It was noticed that some terms were missing in the sub-Jacobian elements related to the loads.

c) Shunt compensator programming

The expression of the compensators sub-Jacobian is given in equations (4.102-4.103). This model represents the injected current differentiated with respect to the voltage of the capacitors linked to node i. These elements are

also included in the diagonal static sub-Jacobian since they are related to one node only and are independent of the rest of the network. Like the line, transformer, and the load admittances the shunt compensator admittances are modelled as frequency dependent.

d) Generator static elements

These elements include the variation of the injected stator current at the generator busbar with the terminal voltage, the voltage behind the synchronous reactance, and the rotor angle. The partial derivative of the generator injected current with respect to the bus voltage represents the generator admittance. These elements are included in the diagonal sub-Jacobian which includes the derivatives of all currents with respect to the bus voltages at the corresponding nodes. This admittance is modelled as a constant, but it can easily be modelled as a frequency dependent variable, if this is necessary.

The diagonal elements are given by equation (4.100) and their location depends on the number of active generators in the system and the node at which the generator is linked, as expressed in equation (5.1).

The relationship between the stator current and the rest of the state variables (rotor angle and the magnitude of the voltage behind the synchronous reactance) is included in the interfacing elements between the static equations and the dynamic system and is given in the subsection below.

C) Interfacing sub-Jacobian

The interface between the generator and the network is manifest by the generator injected current through the stator at node i and the dependence of certain dynamic variables on the network state variables. The dynamic/static off-diagonal sub-Jacobian relates the rotating angular speed to the terminal voltage, and illustrates the dependence of the AVR control closed loop on this state variable. The static/dynamic off-diagonal sub-Jacobian is illustrated by the dependence of the generator current on the rotor angle and the voltage behind the synchronous reactance.

The expressions for the static/dynamic interface elements are given by equations (A.60) and (A.61) and the dynamic/static mutual relation is illustrated by equations (A.58) and (A.59) given in appendix A. As shown in equation (4.104), the position of the elements included in the off-diagonal block (*II*) can be deduced from expression (5.1). The position of the elements of block (*III*) can be obtained by exchanging the rows and columns of the corresponding elements from block (*II*).

5.3.1.2 Solution of the system by Harwell routines

The next step after building the Jacobian and evaluating the right hand side member of equation (3.5), is testing this equation to check if it is within a predetermined tolerance, which is set to 0.005 (p.u.). The system is allowed to converge to an approximate value since obtaining a more accurate solution is time consuming. If the above mentioned condition is not satisfied the use of a sparsity-directed numerical method to solve equation (3.5) becomes unavoidable. In this case the state variable corrections should also be within the tolerance (0.001 (p.u.)) for the system to converge. A new solution is found (new state variables) and the iterative process is repeated until convergence, or until the allowed number of iterations for each time step is exceeded. In both cases the cycle is finished by evaluating the dependent variables and the required messages.

The tolerances corresponding to the linearised functions and the state variable corrections should be selected in such away that an acceptably accurate solution is obtained and over-convergence (i.e. unnecessary extra iterations) is avoided. This depends on the purpose of the study under consideration.

The standard OCEPS simulation uses the *Harwell* library for solving the set of simultaneous linear equations (which have the form $Ax = b$) given by equation (3.5). To evaluate the unknown state variable mismatch x, the known vector b is calculated. It is represented by the linear functions, and the coefficient matrix is the Jacobian matrix.

The *Harwell* library includes a routine based on the LU decomposition sparsity technique to perform the ordering and the factorisation. The ordering of the coefficient matrix considers a compromise between numerical stability and the number of fill-ins to be generated at each solution. This routine is called repeatedly since the performance of the stability test depends on the values of the Jacobian matrix which are updated iteratively. Another routine which deals with the determination of the unknown vector x by using the result of the first routine is also included in this library. The unknown vector is evaluated by forward and backward substitution as mentioned in chapter 3 in equations (3.13-3.17).

a) Ordering and factorisation

The *Harwell* library uses the LU sparsity-directed factorisation method to solve the linear set of equations. The different steps carried out to order and factorise the Jacobian matrix stored in the form of a vector are briefly outlined below.

Initialise the different non zero element indices and tables.
Scan through the matrix (row-wise).
Neglect very small non zero elements and move last non zero elements in the storage to the vacant positions created.
Set the various indices and tables.
Store the Jacobian matrix column-wise.
Start the pivoting (search columns or rows which have not been selected as pivots).
Scan through columns with the least number of non zeros Nz.
Find the potential pivotal row (following the cheapest column/row combination).
Find the largest non zero element in the potential pivotal row.
Perform stability test (is the potential pivot greater than a fraction of the largest element in the row).
If stability test successful;
select this potential pivotal row and column as the actual pivotal row and column,

update the ordering list,
perform the reordering process,
perform the factorisation:
search non pivot rows included in pivotal column for elements to be reduced,
scan the pivot row for fill-ins,
calculate the factor matrix $[U]$,
remove pivotal row and column elements.
Neglect small elements and store the accepted ones,
update the different tables,
eliminate the pivotal column from the pivoting list,
next pivot.
Endif.
If pivot not found;
perform a similar process row-wise.
Endif.
Next column with Nz non zeros.
If last column with Nz non zeros;
increment Nz by one.
Endif.

It should be noted that the factor matrix $[L]$ is equal to the negated reduced matrix, therefore this factor matrix is known implicitly.

b) Direct solution

The unknown state variable mismatch is determined by a separate subroutine. This process consists of performing the forward and backward substitution. This can be briefly described as follows:

Multiply the linear function (the known vector b) by the inverse of the factor matrix $[L]$.
Multiply the vector obtained by the inverse of the factor matrix $[U]$.

The vector obtained is stored in the same location as the known vector. This is achieved by overwriting the elements of this vector during the solution process. This measure helps in saving storage space, since preserving the known vector is not necessary.

5.3.2 Variable updating

The state variables are updated at the end of each iteration and used as initial conditions for the next iteration. The voltage level at each busbar is checked after each iteration. If the set limits are violated a warning is displayed and the transformer tap-changing is incremented or decremented after a certain delay. If the system converges or the maximum number of iterations is exceeded the dependent variables such as the active and reactive power, the bus frequency etc. are calculated. The boiler variables are calculated by using the rectangular numerical method given by equation (4.94). The generator dependent variables are then updated. The generator active and reactive power are given by expression (4.15), the mechanical power is evaluated by using equation (4.33), and the generator frequency is calculated by $f = \frac{\omega}{2\pi}$. The phase angle is updated for each active busbar using the complex components of the bus voltage. The variation of the bus frequency is related to the variation of the phase angle as shown in this expression

$$f_{bus} = f_o + \frac{\Delta\theta}{2\pi\Delta t} \tag{5.4}$$

Following the variation of this dependent variable all frequency dependent admittances are updated using the models described in chapter 4. After determining the new admittances all values dependent on these are subsequently calculated. These variables include the active and reactive power flow, the active and reactive demand and the reactive power generated by the shunt capacitors.

Two subroutines are not performed at each time step but only when it is necessary. These are the shutting down and starting up of the generators, and the re-synchronisation of a split system. These two functions are included in the flow chart of figure (5.2).

5.3.3 Generators start-up and shut-down

The generator starting up and shutting down is initiated by unit commitment. This sets a flag to -1 if a generator is to be removed and 1 if the re-connection of a generator is required. This flag is sent to the simulator from the unit commitment package through the control package communicated data. If a start-up of a generator is necessary, the generator is re-synchronised by the simulator which then ramps up its active power to its minimum output. By contrast, shutting down a generator is done by gradually decreasing its output to the minimum by the load frequency control. The simulator reduces then the power to zero and disconnects the generator at the target time set by unit commitment. This smooth manipulation of the active power is necessary to avoid sudden changes of generation and its consequential undesirable results.

5.3.4 Synchronisation

The synchronisation is carried out if the scenario file sends a command for re-connecting two independent islands. A synchronising target time is received by the simulator together with the frequency, phase angle and voltage magnitude tolerances. The simulator selects the two ends of the line to be re-synchronised, checks if the three necessary conditions for the re-synchronisation of a line are satisfied and performs the actual re-connection. These conditions consist of reducing the difference between the frequencies, the phase angles and voltage magnitudes at the two ends (located in two different islands) of the line to the above mentioned tolerances. If these conditions are not satisfied within the specified target time, the re-synchronisation is aborted. This limitation is imposed because the use of scenario files necessitates specific times for each action. However, a manual re-synchronisation can be restarted if necessary.

5.3.5 Protection simulation

After determining the state variables and the dependent variables of all the simulated network elements the evaluation of the protection becomes possible. The generator over- and under-speed protection depend on the angular speed, the load under-frequency protection depends on the bus frequency and the circuit overload protection depends mainly on the power flow of the transmission lines and transformers. If any of these variables violates the preset conditions

the corresponding element is tripped and a message is added to the list of abnormalities.

It should be noted that the protection models for generator over-speed and load under-frequency which have been described are a revision of those in the standard simulator which uses the same model for the three types of protection presented.

5.3.6 Telemetry and measurements

After providing all the necessary data, the measurements can be simulated together with the telemetry. These two tasks are performed whenever there is a topology change or the state estimator and data validation tasks require new sets of data. In the case of topology changes the telemetry and measurement is not carried out until the dynamic data corresponding to the latest network structure is evaluated.

The measurement function corrupts the selected simulated variables by adding to the calculated values the static and dynamic errors mentioned in chapter 4 in equations (4.108) and (4.109). If the calculated value is null, its corresponding measurement is set equal to the static error. This task can also simulate the failure of transducers, if commands for such operations are received from the scenario file. Digital errors representing the erroneous communicated switch states are also included.

When the measurement is evaluated the telemetry is updated. Then the data can be transferred to the control package through the SCADA system.

5.4 OUTPUT RESULTS

The simulator is equipped with a graphics package which plots the main variables on graphic screens. This enables the user to ~~analyse~~ the correctness of the results obtained from simulating given events. This also allows for the assessment of the simulator robustness and adequacy in handling various events.

5.5 IMPLEMENTATION OF THE SIMULATOR ON PE 3230

First the developed algorithm was implemented on a minicomputer PE 3230. This machine is a serial 32 bit general-purpose computer where all operations are performed sequentially. It has a capacity of 0.5 Mflops, and a 1 Mips processor. This machine is not particularly efficient in handling floating-point operations. The general structure of the program is given in figure (5.1). The details of the time and iterative loops are shown in figure (5.2) and (5.3).

In order to preserve the validity of the simulated variables over a time step flags are set to prevent other tasks from accessing the common blocks currently used by the simulator. When the supporting packages are running the validity of the common blocks is preserved by putting the simulator in a waiting mode as shown in figure (5.1). This happens whenever the exact topology, the loader, or the control signals are invoked.

The tests carried out on this machine using the *Harwell* library routines showed the incapability of this algorithm in realising real-time simulation even for the small size network of 30 nodes [90,92].

5.6 IMPLEMENTATION OF THE SIMULATOR ON FPS 5205

The utilisation of Perkin Elmer (PE) 3230 and 3220 minicomputers has revealed their limitations in simulating even the relatively small IEEE 30 node test network in real-time [90,92]. The availability of a high speed (167ns cycle time) pipelined floating-point processor FPS 5205 Array Processor (AP) has provided a powerful tool to be used in parallel with the PE 3230 minicomputer so that more processing speed can be achieved. Array processors are a form of special-purpose hardware device mainly designed to efficiently handle arithmetical sequences of operations. They are single instruction multiple data machines (SIMD). Their highly-parallel structure eliminates the overheads caused by array indexing, loop counting and data fetching and allows the simultaneous performance of these functions and the arithmetical operations. Compared to general-purpose computers, this Array Processor allows much higher execution

speed [42] at an estimated computational power of about 12 Mflops when register to register operations are performed. The Array Processor characteristics are expected to satisfy the requirements of numerical power system simulators which have the task of evaluating hundreds of variables iteratively and require high precision. Not all computations performed are floating-point operations; pointer manipulations are heavily employed in sparse matrices. Some logical operations are also required.

The structure of the AP suggests its suitability for solving power system numerical models which involve floating-point operations and large matrices and vectors. Since this machine is incapable of performing IO operations, a host computer is used in order to communicate with other tasks and peripherals. Besides this function the host computer can perform less intensive tasks in parallel with the AP while it is waiting for the latter to finish its calculations. In this way the host computer idle time can be exploited and the computation burden of the AP can be alleviated to a certain extent. The host computer, therefore, provides additional resources in the form of computational power and data memory. This enhances the Array Processor performance.

As shown in figure (5.4), the simulator tasks have been subdivided in the following way between the Array Processor and the host computer:

The AP performs the Jacobian building as well as the solution of the subsequent sets of linear equations together with the dependent and independent variable updating, and emergency protection (if the updated quantities violate their limits). It also carries out the synchronisation of the system if it is islanded and performs generators start-up and shut-down when a request is issued.

The host computer deals with the operations concerned mainly with IO, and acts as an intermediary between the AP and its environment. It presents the AP with the topology changes, load variation, updated set points etc. On the other hand, it ensures communication with the

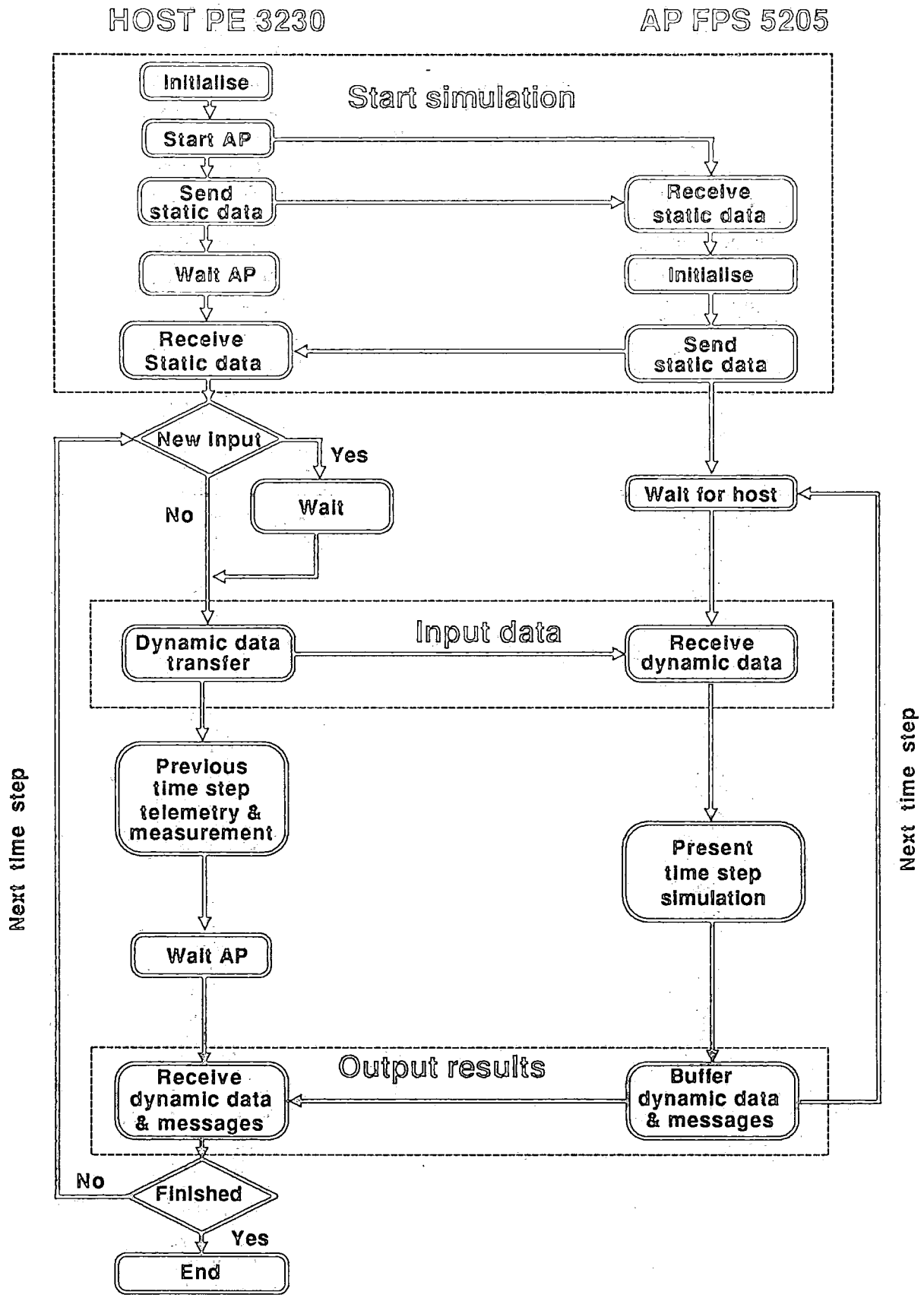


Fig. 5.4 AP/Host parallel processing

SCADA machine (Supervisory Control and Data Acquisition), communicates the latest network events coming from the simulator to the topology subprogram to update the structure of the system. The host computer performs the simulation of measurement and telemetry by corrupting selected data evaluated by the AP before sending it to the control package via the SCADA system. The host computer communication with the AP is achieved through Direct Memory Access (DMA) and its communication with other tasks is through shared common blocks.

The benefit from arranging the two machine functions in this way is in exploiting their potentialities to the maximum and minimising data transfer. The two programs must be structured in such a way that the number of synchronisations which involve waiting by one of the programs is minimised.

Before starting any of these functions, the simulator requires initialising. The initialisation of the AP machine is followed by opening and loading the AP simulator and specifying the data addresses in the memory to facilitate data transfer between this machine and the host.

5.6.1 Initialisation of the Array Processor program

After initialising the common blocks and the AP machine and specifying the portion of memory to be used for transferring the data, the latter can be buffered and sent to its predetermined address so that the actual simulator can be initialised. This is performed only once for each simulator run. The initial values such as the parameters of the different plants of the network, its connectivity, node voltages and so on are communicated from the host computer to the AP where the actual numerical simulation takes place. Some of this communicated data, such as the parameters, are static. Since this is the case, this data need not be buffered, rather it is sent from the host computer to the AP individually. The dynamic data needed by the AP program includes the generator set points, the breaker and switch states, the loading factors etc., and the evaluated data consists of the node voltages, the frequency at each busbar, the power consumed by each load, the generating unit variables such as the electrical and mechanical power, the different pressures, and so on.

The initial values provided by the supporting packages are performed in the same manner as described in section (5.2). The calculation of the simulator local initial conditions are also performed as in section (5.2).

5.6.2 Array Processor control

There are several mechanisms for controlling and synchronising an AP program with a host program. An AP program is started by the host calling the CPRUN routine. The AP program stops itself by using a FORTRAN PAUSE statement, with which an optional identifying number can be returned to the host. The host may wait for an AP program to pause by calling the function ICPVAL, which returns the pause number specified by the AP. It can continue the AP program after a pause by calling the CPCONT routine.

Data communication between the two machines is achieved through direct memory access (DMA). The host generates all (DMA) operations by using the EXPUT routine when sending the data from this machine to the AP and the EXGET routine if the data is received from the latter. The arguments of these routines include the addresses of the data to be transferred, their quantity, and the type of conversion. The (DMA) facility allows the IO board to access the specified addresses and transfer the data while the CPU carries on with the calculation.

Two ways of communicating data to and from the Array Processor (AP) are available. The first one consists of specifying the name of the variable and a search for its address in the memory. This approach involves a large overhead time caused by the use of the EXPUT and EXGET routines for each variable. The second way is more efficient since the variables are buffered into a real and an integer buffer and communicated by calling the two routines once at each time step. In this case the initial addresses and the size of the buffers are specified for both machines thus eliminating the overhead necessary to find the location of each variable and significantly reducing the CPU time. The static data however is transferred individually using the first process so that its allocated memory is not left idle. Although this process is slow, since it

is carried out only once this elapsed time is not an obstacle for the simulator and it saves memory space.

The integer variables have the same structure in both machines, therefore they are transferred without any conversion. Real variables need conversion since the AP words are 38 bits long whereas the host computer words are 32 bits long. This re-formatting takes place in the AP whether the data is sent or received by this machine.

Since the AP has no output facilities, messages should also be communicated to the host computer to be displayed and processed. The messages generated in the AP are also sent to the host computer in a packed form. The host computer specifies the required address for this function in the same manner as data addressing. The AP uses a subroutine called FWRITE which gathers all messages in a $(4 * number\ of\ messages)$ array. Four indicates the number of arguments used in this subroutine. The first argument is a code related to the type of the message and the rest of the arguments indicate the variables involved. One or two of the last arguments can be set to zero if only one or two variables are to be transferred. This array is then communicated to the host computer in the same way as the data.

5.6.3 The host computer time loop

After initialising the AP and the host program, performing the tasks which do not need repeated calculation such as the allocation of addresses to the data and messages, the communication of the static data, and the evaluation of any static errors to be implemented on the measurements, the host program starts its time loop.

As shown in figure (5.4) the host time loop starts by sending the dynamic data. In order to preserve synchronism between the simulator and the supporting packages, and avoid erroneous calculation, the host program has to wait for these tasks to update the common blocks and the network structure whenever a change is needed. This is done by setting flags which warn the host program that the common blocks are being updated and cannot

be accessed until the present calculation is over. When these functions finish their calculation, the results are included with the dynamic data and sent to the AP which is also in a waiting mode.

While the AP is evaluating the numerical calculation, the host computer concurrently performs the telemetry and measurement calculations. When these routines are active the physical data and telemetry common blocks are locked to prevent other tasks from accessing them and changing the variables which are currently being updated by the host computer. This process preserves the validity of the data and allows for synchronism between the different programs accessing the same common blocks. These are released as soon as the host finishes its calculation.

5.7 CONCLUSION

In the *Harwell* approach the ordering is time consuming since two conditions are imposed for the selection of the pivot. These conditions are:

(1) The pivot element should not be small in comparison with other elements in the pivotal row.
(2) The number of the non zero elements in the pivotal column and row must be as small as possible.

The first condition is satisfied by not allowing the pivot element to be smaller than a fraction of the largest element in the pivotal row. Since this depends on the values of these non zero elements which are evaluated iteratively, the ordering also requires an iterative solution.

The power system Jacobian diagonal elements are usually large enough, therefore such a compromise between the minimum number of non zeros in the pivotal column and the value of the pivot is not necessary. The only task of the ordering scheme should be minimising the number of fill-ins by selecting the pivot as the column or the row with the minimum number of non zeros. This does not depend on the values of non zero elements, but depends on

the number of these terms in each column or row. Consequently, this process should be carried out separately from the factorisation which depends on the values of the Jacobian matrix and needs to be evaluated at each iteration.

A technique which takes advantage of these characteristics, thus saving enormous computational time is the *bifactorisation* approach described in the following chapter and in chapter 3.

Originally the *Harwell* algorithm was implemented on the Perkin Elmer PE 3230 minicomputer. An Array Processor was then used with the expectation that this powerful machine which has a rating of 12 Mflops would improve the original simulator by a factor of at least 12. The results obtained were rather disappointing. The 30 node test network execution time was speeded up by a factor of only three times [90]. This is due to the nature of the large amount of scalar index calculations required for each floating-point operation and the scalar nature of the calculation. This mix of computing cannot fully exploit the pipelined architecture of Array Processors. This drawback implies the need for a sparsity algorithm that reduces the large indexing overhead. Such a technique is described in the following chapter.

CHAPTER 6

A NEW ALGORITHM APPLIED TO PARALLEL AND CENTRALISED SIMULATORS

6.1 INTRODUCTION

In this chapter a novel algorithm is introduced for parallel and centralised simulation. This novelty has been achieved by using a very efficient approach based on the Zollenkopf *bifactorisation* technique [117] which groups the non zero elements of the coefficient matrix to be inverted into $2 * 2$ sub-matrices. The developed technique is used for solving the set of linear equations. Also some of the existing power system models are modified and some new models are introduced for further improvement of the system stability.

The implementation of the Newton-Raphson method with a quadratic rate of convergence necessitates updating the Jacobian matrix at each iteration. The inversion of this matrix using sparse techniques at each iteration is very expensive. To overcome this problem Stott [106] suggested a compromise and used the same Jacobian matrix for several successive iterations. However, this affects one of the most significant characteristics of the method; i.e. the quadratic convergence. Since this compromise affects the (already approximate) accuracy of the method it is judged more advantageous to seek faster sparsity techniques rather than adopting this type of compromise.

The investigation in this chapter is based on suggestions and observations presented in references 68 and 90. The major change is the algorithm used for solving the set of simultaneous linear equations. It was estimated that this change would improve the speed of the simulator by a factor of 8 to 10 since most of the execution time is spent in solving these equations repeatedly.

Consideration of the *Harwell* subroutine LA05A which is used for ordering and factorising the Jacobian matrix revealed that most of the calculation is repeated unnecessarily. Among these drawbacks are:

(1) The adoption of the stability test, which is not necessary since the power system Jacobian matrix has naturally strong diagonal elements. The only task of the ordering would then be minimising the number of fill-ins without any need for compromise.
(2) The ordering can be carried out without moving the pivotal rows and columns within the matrix, it is sufficient to store the reordered column or row pointers in a table.
(3) The factorisation would better be performed separately from the ordering scheme since it depends on the values of the non zero elements which are updated iteratively and this requires a repeated manipulation.

In this case the ordering would need to be re-computed only if the structure of the matrix changes, since it involves indices and tables which are related to the structure of the matrix only.

To permit the solution of the linear equations to execute efficiently, the *bifactorisation* approach described in chapter 3 has been selected. This method separates the ordering scheme from the factorisation and splits the factorisation task into a dynamic part which involves the evaluation of the factor matrices and a static part which deals with the location of the factorised elements. This indexing varies only if a topology change occurs. Therefore, the indexing part of the program has been appended to the ordering subroutine. This feature has also been exploited by Woo [113] and Rainbolt et al.[94]. Reference 113 applied this approach to the LU decomposition technique and reference 94 used it for a matrix factorised into an upper triangular form by Gauss elimination.

Since both the *Harwell* library and the *bifactorisation* method are based on the Gauss elimination technique the amount of the arithmetical calculation involved is not expected to be very different. The difference resides in the

programming process which takes advantage of the nature of the problem to be solved and its features and characteristics.

To highlight the performance of the *bifactorisation* method, it has been implemented on both the parallel simulator running on the Array Processor FPS 5205 hosted by a minicomputer PE 3230, and the host only centralised version.

6.2 APPLICATION OF THE 2 * 2 SUB-MATRIX ALGORITHM

Another feature of power system models is that the network sub-Jacobians are all grouped into 2 * 2 sub-matrices to represent the derivatives of the complex currents with respect to the complex bus voltages. This characteristic is exploited for load-flow studies by Dodson [23] and Irving et al. [66], and for transient stability studies by Rainbolt et al. [94]. Dodson applied this scheme to the LU decomposition method and Irving applied it to the *bifactorisation* technique.

Dodson's algorithm requires less memory space unless 75 % of the involved equations are scalars. The scalar representation requires less memory space for the matrix elements, but, more integer pointers. In the case of the 2 * 2 blocks algorithm one pointer is sufficient for 4 elements.

Traditionally, when computers execute instructions sequentially, efficient methods are those which involve the least number of arithmetic operations. However, with modern machines which can execute many operations in parallel this ~~criteria~~ criterion is no longer valid. Considering these points Dodson presented three possibilities of computing the set of simultaneous linear equations:

(1) All non zero elements of the Jacobian matrix are treated as scalars even if they present particular patterns such as 2 * 1, 1 * 2, or 2 * 2. In this case pointers for each element are necessary. No zero and one elements (necessary to make up the chosen pattern) are stored and no unnecessary null operations and multiplications by unity are included.

(2) All non zero elements whether they are scalars, $1 * 2$, or $2 * 1$ sub-matrices are represented in $2 * 2$ sub-matrices. This approach reduces the number of pointers by at most 75 % if all elements are $2 * 2$ sub-matrices. Scalar elements are represented as $\begin{bmatrix} a_{ii} & 0 \\ 0 & 1 \end{bmatrix}$ or $\begin{bmatrix} a_{ij} & 0 \\ 0 & 0 \end{bmatrix}$ depending whether they are diagonal or off-diagonal elements. The $1 * 2$ and $2 * 1$ elements are stored as $\begin{bmatrix} a_{i,i} & a_{i,i+1} \\ 0 & 1 \end{bmatrix}$ or $\begin{bmatrix} a_{i,j} & a_{i,j+1} \\ 0 & 0 \end{bmatrix}$, and $\begin{bmatrix} a_{i,i} & 0 \\ a_{i+1,i} & 1 \end{bmatrix}$ or $\begin{bmatrix} a_{i,j} & 0 \\ a_{i+1,j} & 0 \end{bmatrix}$ respectively. The diagonal elements include ones instead of zeros to preserve the non singularity of the matrix required by matrix inversion. In this case the storage of zero elements and their arithmetic manipulation are unavoidable.

(3) This option consists of handling the different sub-matrices in their natural pattern using a mixture of sizes. In this case also compared to the scalar scheme the number of pointers is reduced, and no null operations or extra storage space is necessary.

At the first glance the third option may appear to be the best option since it takes advantage of the natural structure of the matrix and does not involve any penalties in terms of both storage and number of operations. However, the mixture of sizes is not suitable for pipelined computers since this does not exhibit regular data structure. Furthermore, this scheme is not flexible as far as programming is concerned. It has been suggested that the second option is the most convenient approach. It is expected to execute the calculation at a higher computational rate, especially when implemented to the AP which requires a small number of integer calculations [42]. The application of the second scheme greatly reduces the number of such operations.

This approach has been tested on the parallel dynamic simulator. The network equations are naturally $2 * 2$ sub-matrices. The generators and the interfacing scalar non zero elements were represented in a similar way to option (2). The dynamic vectors were stored as $\begin{bmatrix} v_i \\ 0 \end{bmatrix}$ and the static vectors were stored as $\begin{bmatrix} v_i \\ v_{i+1} \end{bmatrix}$. Extra rows and columns were needed for each scalar. The execution time of the AP in this case was 0.225 seconds, about half of the standard simulator timing, and the total time was about 0.3 seconds which

is about 1.8 times faster than the overall execution time of the conventional parallel simulator.

Consideration of the structure of the Jacobian matrix revealed that the number of extra stored zeros can be greatly reduced if the dynamic and interfacing sub-Jacobian elements are naturally grouped into $2 * 2$ sub-matrices following their location in the matrix. No additional 1s need be stored since all the diagonal elements are non-null. Four neighbouring elements can be gathered, as shown in equation (6.1), to form a $2 * 2$ block. The zero elements which are stored to satisfy the symmetry of the structure of the matrix required by the *bifactorisation* technique are represented by crosses. Those which are included in the $2 * 2$ sub-matrices are represented by zeros. As a result of this grouping there is no need for artificial expansion of the matrix dimension, and the order of the matrix of $2 * 2$ blocks is exactly half that of the original Jacobian matrix. The $2 * 2$ blocks can be full as is the case of the network elements. Dodson's algorithm requires the expansion of the scalar, the $2 * 1$ and $1 * 2$ elements. Therefore, the approach adopted here not only reduces the storage space, it also reduces the number of zero operations.

The vector representation in the case of the Dodson algorithm can be stored either as $\begin{bmatrix} V_i \\ 0 \end{bmatrix}$ or $\begin{bmatrix} V_i \\ V_{i+1} \end{bmatrix}$ depending on the structure of the matrix, whereas in the adopted method their dimension is halved and they are represented as full $2 * 1$ sub-vectors. This scheme reduces the sparsity of the system, but this reduction is trivial when compared to the benefits offered. Table (6.1) represents the sparsity percentage when the non zero elements are grouped into $2 * 2$ sub-matrices. In this case the number of dynamic $2 * 2$ non zero blocks is 14 for each generator and the number of the interface $2 * 2$ sub-matrices are 4 as shown ~~in equation (6.1)~~ on page 159. Table (6.1) is equivalent to table (5.1) which represents the sparsity of the same systems when the non zero elements are treated as scalars.

$$
\begin{bmatrix}
\frac{dh_1}{d\omega} & \frac{dh_1}{d\delta} & \frac{dh_1}{d|E_g|} & X & \frac{dh_1}{dP_{hp}} & \frac{dh_1}{dP_{ip}} & \frac{dh_1}{dP_{lp}} & X & \cdots & \cdots & \frac{dh_1}{dV_{ri}} & \frac{dh_1}{dV_{imi}} & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\
\frac{dh_2}{d\omega} & \frac{dh_2}{d\delta} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & X & X & & & & & & \vdots \\
X & 0 & \frac{dh_3}{d|E_g|} & 0 & 0 & 0 & & & \cdots & \cdots & \frac{dh_3}{dV_{ri}} & \frac{dh_3}{dV_{imi}} & & & & & & \vdots \\
\frac{dh_4}{d\omega} & 0 & 0 & \frac{dh_4}{dP_{gv}} & X & 0 & & & \cdots & \cdots & 0 & 0 & & & & & & \vdots \\
X & 0 & 0 & \frac{dh_5}{dP_{gv}} & \frac{dh_5}{dP_{hp}} & X & 0 & \frac{dh_5}{d(\Delta DP)} & \cdots & \cdots & & & & & 0 & & & \vdots \\
X & 0 & 0 & 0 & \frac{dh_6}{dP_{hp}} & \frac{dh_6}{dP_{ip}} & X & 0 & \cdots & \cdots & & & & & & & & \vdots \\
X & 0 & & & 0 & \frac{dh_7}{dP_{ip}} & \frac{dh_7}{dP_{lp}} & 0 & \cdots & \cdots & & & & & & & & \vdots \\
\frac{dh_8}{d\omega} & 0 & & & \frac{dh_8}{dP_{hp}} & 0 & 0 & \frac{dh_8}{d(\Delta DP)} & \cdots & \cdots & & & & & & & & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
X & \frac{dI_{rig}}{d\delta} & \frac{dI_{rig}}{d|E_g|} & 0 & & & & & \vdots & \cdots & \frac{dI_{ri}}{dV_{ri}} & \frac{dI_{ri}}{dV_{imi}} & \cdots & & \frac{dI_{ri}}{dV_{rj}} & \frac{dI_{ri}}{dV_{imj}} & & \cdots \\
X & \frac{dI_{imig}}{d\delta} & \frac{dI_{imig}}{d|E_g|} & 0 & & & & & \vdots & \cdots & \frac{dI_{imi}}{dV_{ri}} & \frac{dI_{imi}}{dV_{imi}} & \cdots & & \frac{dI_{imi}}{dV_{rj}} & \frac{dI_{imi}}{dV_{imj}} & & \cdots \\
\vdots & & & & & & & & \vdots & \cdots & \vdots & \vdots & \ddots & & \cdots & \cdots & & \cdots \\
\vdots & & & & 0 & & & & \vdots & \cdots & \frac{dI_{rj}}{dV_{ri}} & \frac{dI_{rj}}{dV_{imi}} & \cdots & & \frac{dI_{rj}}{dV_{rj}} & \frac{dI_{rj}}{dV_{imj}} & & \cdots \\
\vdots & & & & & & & & \vdots & \vdots & \frac{dI_{imj}}{dV_{ri}} & \frac{dI_{imj}}{dV_{imi}} & \cdots & & \frac{dI_{imj}}{dV_{rj}} & \frac{dI_{imj}}{dV_{imj}} & & \cdots \\
0 & & & & & & & & \vdots & \cdots & \vdots & \vdots & \vdots & & \vdots & \vdots & & \ddots
\end{bmatrix}
\tag{6.1}
$$

n	N_g	N_l	No. of non zeros	Total elements	Sparsity (%)
30	6	41	220	2916	7.5
118	19	179	818	37636	2.17
141	12	204	765	35721	2.14

Table 6.1 Sparsity of 3 test networks using the $2 * 2$ sub-matrix algorithm

The reduction of the sparsity may be attributed mainly to the storage of zero elements to maintain the symmetry of the matrix and form the $2 * 2$ blocks. From table (6.1) it can be noticed that the sparsity factor is better for larger systems. This is expected to result in a better performance of the *bifactorisation* algorithm for large networks.

6.3 MODIFICATIONS OF THE CONVENTIONAL SIMULATOR

Based on observations of the trends plotted from the conventional simulator, and the consideration of reference 68 and the program, the mathematical models of the power system presented in reference 93 have been revised.

6.3.1 Extrapolation

The extrapolated values are expected to provide better starting guesses for the Newton-Raphson iterative algorithm to allow for faster convergence. This was first implemented by Irving [68]. This approach has also been tested on the present simulator and has shown an improvement of approximately 20 % in solution speed by reducing the number of iterations. Two state variables have been extrapolated, the rotor angle and the bus voltage. The voltage magnitude and its phase angle are extrapolated independently as suggested by Stott in reference 106.

The extrapolation of these variables is represented as a function of the frequency. The models for this process are given by equations (3.10-3.12).

6.3.2 Modification of the Jacobian matrix

The major modifications applied to the Jacobian matrix are: the method of storing the non zero elements, the modification of some models and the introduction of previously neglected functions.

The Jacobian matrix is subdivided into diagonal elements, the upper triangle non zero elements and the lower triangle elements which use the rows and columns in an opposite sense. The non zero elements are grouped into $2 * 2$ sub-matrices as stated earlier. These $2 * 2$ sub-matrices may be full or sparse. They can even be null if zero elements, with zero neighbouring terms, have to be stored to satisfy the matrix incidence symmetry necessary for the *bifactorisation* method. The state variables and other vectors involved in this calculation are stored as $2 * 1$ vectors.

In order to solve the whole Jacobian matrix as one system, the dynamic and the interface sub-Jacobians are represented as a network-type system following their matrix structure. Each diagonal non zero element is represented by a node, with a path connecting it to all nodes (non zero elements) in the row. The diagonal elements are equivalent to the nodal self admittance and the off-diagonal terms are equivalent to the mutual admittance. The rows and columns are taken as the sending and receiving ends of a fictitious branch.

6.3.2.1 Introduction of governor constraints

By observing the behaviour of simulated generator active powers it was noticed that the governor limiters were neglected in the program [61]. These constraints have been included by forcing the governor output to take the lower or the higher limit whenever one of these limits is violated. Under these conditions the Jacobian elements related to the governor are null.

These state variables should be checked outside the iteration loop at each time step to avoid testing a temporary value instead of the final governor output. This avoids imposing a limit erroneously.

6.3.2.2 Modification of the load model

The loader program sends new load levels (relevant to the beginning of a time step) to the simulator every 10 seconds. Therefore, the simulator calculations related to this task also have to be performed at the beginning of the time step, otherwise the Jacobian matrix would be using non updated values at each time step following the variation of the total load, leading to erroneous results.

Considering the FORTRAN code related to the load model, it has been noticed that some terms were not included in the Jacobian matrix. The model has been corrected to that in equation (4.101). This correction has improved the performance of the simulator by reducing the average number of iterations from 3.2 to 2.8 when tested on the centralised version.

6.3.2.3 Introduction of shunt reactors

Networks may be subject to reactive power deficit as well as reactive power excess. Both problems have an impact on the voltage profile. Therefore, the consideration of shunt reactors can be as important as the shunt capacitors. The models representing the Jacobian elements related to these devices are given in equation (4.103).

6.3.2.4 Voltage limits

The consideration of the curves and events obtained from running a given scenario demonstrated that some breakers were activated by numerical instability. This occurred because the bus voltage was tested at the end of each iteration before the final convergence of the system, without considering that these limits could be easily exceeded during the transient state. The seriousness of the problem of cancelling the calculation and starting another time step is the use of these state variables as initial guesses for the Newton-Raphson iteration. Consequently, this phenomena may be propagated for many time steps leading to a cascade of bus disconnections resulting in de-energising lines and generators and even a blackout in the extreme case.

This problem is general to all limited variables. Therefore, care should be taken when placing these models in the program. This approach is illustrated in figure (6.1) which is a modification of figure (5.3).

6.3.2.5 Bus frequency determination

The modification of the bus frequency and the frequency dependant variables consisted of performing this calculation at the beginning of the time step instead of at the end. This is necessary because the new topology is sent to the simulator with the rest of the dynamic data before the numerical calculation is started. If the frequency evaluation and the topology updating are out of step the results of the simulator can be erroneous; especially if an islanding occurs and the variables affected by this phenomena (such as the average frequency of the system) do not consider the new structure of the network during the first step following this event.

6.3.2.6 Protection

The protection simulation has been enhanced based on suggestions from the CEGB. These enhancements include widening the range of frequency over which the load under-frequency protection operates and modifying this model to simulate a device which acts instantaneously. The generator over-speed protection tripping has also been replaced by instantaneous actions. A generator under-speed protection has been added to prevent the generator from operating at unrealistic speed levels. The description of these tasks was presented in chapter 4.

6.3.3 Use of the Zollenkopf algorithm

The nature of the power system problems offers a number of advantages which can improve the efficiency of the numerical technique used to solve the set of linear equations by developing a special-purpose algorithm. The fact that the network topology does not change often enables the developer of the algorithm to subdivide it in such a way that the calculation related to the structure of the matrix (network) can be separated from the arithmetical computation of the non zero elements which are updated iteratively. Furthermore, some particular aspects of the matrix such as the potential predominance of the number of full

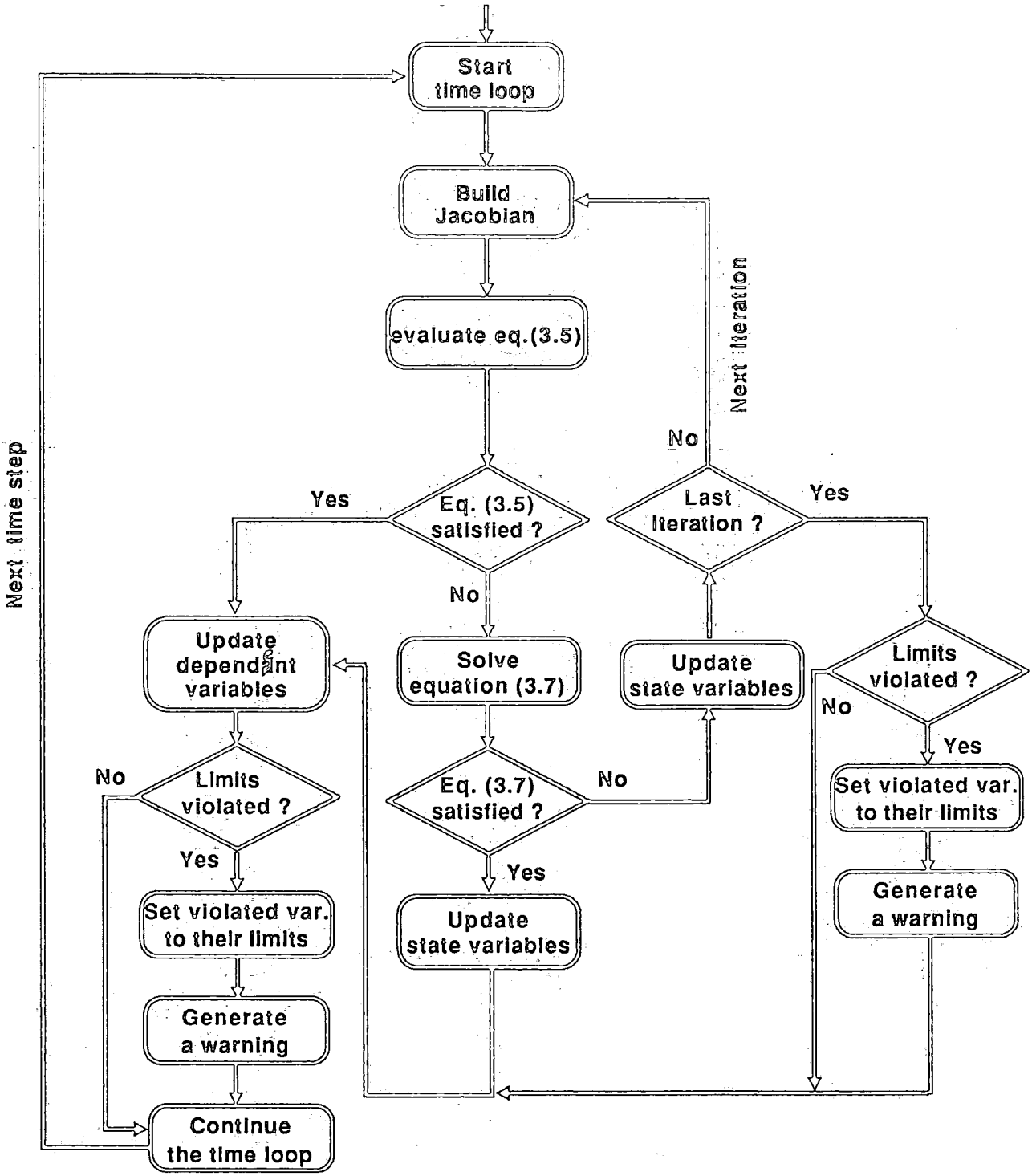


Fig. 6.1 The modified simulator iterative loop

$2 * 2$ sub-matrices over the scalar elements has led to the generalisation of the scheme presented in reference 66 to the whole system instead of the load-flow problem only.

The advantages offered by exploiting the nature of the problem and implementing this $2 * 2$ sub-matrices scheme to the *bifactorisation* technique for solving the set of linear algebraic and dynamic equations are:

(1) Reduction of the amount of indexing information by separating the diagonal elements from the off-diagonal elements and by the use of the $2 * 2$ sub-matrices algorithm.
(2) The separation of the indexing and the updating of the non zero elements since the location of these elements are the same for each matrix structure.
(3) The execution of the reordering of the matrix and its indexing only when the topology changes.
(4) The factorisation of the $2 * 2$ non zero elements and the direct solution is expected to be faster than the factorisation and the solution of scalar elements because of the parallel processing ability of the AP and the reduction of the number of indices (integer operations) by about 75 %. The reduction of indexing is also expected to reduce by 75 % the undesirable frequent random memory access imposed by the way the non zero elements are stored.

The AP is a single instruction multiple data machine which can perform all arithmetic operations synchronously at a cycle time of 167 ns as well as the indexing, loop counting, and data fetching from the memory for the next operations. The proper exploitation of the AP features requires long vectors and regular structure [42,52]. The use of the $2 * 2$ algorithm satisfies the second condition. Unfortunately, satisfying the first condition is not possible with the implementation of sparsity techniques.

Since the sparsity-directed methods require the storage of the non zero elements of the coefficient matrix, this matrix is not stored as an $n * n$ array. Rather it is stored in a compact form and thus enormous storage space is saved.

Therefore, to identify the non zero elements of the reduced matrix and perform the factorisation process in any desired order, indexing information becomes necessary. This process necessitates extra programming and storage. However, compared to the storage space required by a full matrix, this additional effort is worthwhile especially for large networks to which the application of sparsity techniques is essential. The disadvantage of storing the non zero elements randomly is that advantage is not taken from the efficient vector processing of the Array Processor. Although, this is alleviated by storing and processing the non zero elements as $2 * 2$ sub-matrices. This strategy reduces to a certain extent the number of memory access.

In order to fully exploit the advantages offered by this sparsity-directed method implemented for electrical networks, the algorithm has been subdivided into four major steps:

(1) indexing the non zero elements of the Jacobian matrix,
(2) ordering of the coefficient matrix and simulation of the factorisation,
(3) factorisation of the ordered matrix, and
(4) direct solution.

The allocation of the non zero elements to the different arrays is carried out in the main part of this algorithm after performing the indexing.

These steps are illustrated in the flow chart given in figure (3.2).

6.3.3.1 Indexing routine

To facilitate this task and account for the complex process of the variation of the number of non zero terms of the reduced matrices, a flexible indexing is essential. For this purpose two lists (ITAG and ICOL) representing all the necessary information to reconstruct the original matrix are built. These are deduced from the rows and columns of the stored elements and are defined in chapter 3.

The storage of the non zero elements and their indices would have been easier if the factorisation process does not include elimination and generation of non zero terms. This problem can be overcome by using a flexible scheme called a *linked list* represented by vector LNXT (defined in chapter 3) to enable the storage of the non zero elements in any sequence. In this way the deleted and generated elements can be handled efficiently without having to use primitive element shifting process which can be computationally very time consuming. This provides the algorithm with a powerful means to perform the dynamic ordering and the factorisation in any desired order.

Using the number of branches and their sending and receiving ends (rows and columns) evaluated with the construction of the Jacobian matrix, the integer vectors LCOL, ITAG, and LNXT can be constructed as follows:

Consider each branch or fictitious branch:
consider the stored non zero elements in the corresponding column:
find the correct position for the element to be inserted in the coefficient matrix (by comparing its row with the row number of the existing non zero elements in the column),
determine its address in the vector of compacted non zeros (increment the address of the latest inserted element by one),
increment the number of non zeros in the column,
set ITAG to the inserted element row number.
If the inserted element is the first in its column;
set LNXT of that location to the previous LCOL of the corresponding column,
set LCOL of that column to the address of the inserted element.
Else;
set LNXT to the pointer of the preceding element in the column (point to the next element address or zero if the inserted element is the last in its column),
set LNXT of the preceding element in the column to the inserted element address.
Endif.

Increment the index of the compacted vector.
Process next branch.

This function is carried out separately for the upper and lower triangles. The former is processed row-wise and the latter is processed column-wise. One routine is used for both cases with the rows and columns exchanged. In order to facilitate the understanding of this process the use of the modified Jacobian matrix given in figure (3.5) may be helpful.

The inserted non zero elements may or may not be stored next to their natural neighbouring elements in the original $n * n$ coefficient matrix. They are stored at the end of the array as shown in table (6.2). This table illustrates the step by step process of determining the indices and lists of some of the non zero elements to be stored in a packed form. This process is first carried out row-wise to store the upper half of the Jacobian matrix. The assignments of the non zero elements are not evaluated at this stage, they are included in these tables for the sake of clarity only. The final values of the represented indices and lists are given fully in tables (3.1) and (3.2).

It should be noted that the fictitious branches mentioned earlier do not necessarily correspond to physical transmission lines. They are deduced from the structure of the coefficient matrix (Jacobian matrix). In this case the hypothetical sending and receiving ends are the rows and columns or vice-versa.

6.3.3.2 Ordering and simulation routine

The ordering and simulation phase is processed separately since it does not require as many executions as the factorisation and the direct solution, and its execution time is quite significant and cannot be neglected. This is because indexing requires multiple memory references to obtain the element address. Therefore, performing this costly process once for each matrix structure greatly reduces the time necessary to perform a solution especially if this is needed repeatedly.

Location	LCOL	NOZE
2	1	2,3,4*
3	2	2,3,4
4	3	2,3
13	4	2,3
6	8	2
7	9	2
8	10	2,3
14	11	2,3
10	15	2
⋮		
1	31	2,3,4,5
5	38	2
⋮		

Location	ITAG	LNXT	CE	RE
1	1	0,35	$a_{1\ 2}$	$a_{2\ 1}$
2	1	0,5	$a_{1\ 3}$	$a_{3\ 1}$
3	1	0,7	$a_{1\ 4}$	$a_{4\ 1}$
4	1	0,6	$a_{1\ 13}$	$a_{13\ 1}$
5	2	0,37	$a_{2\ 3}$	$a_{3\ 2}$
6	2	0	$a_{2\ 13}$	$a_{13\ 2}$
7	3	0	$a_{3\ 4}$	$a_{4\ 3}$
8	5	0	$a_{5\ 6}$	$a_{6\ 5}$
9	5	0	$a_{5\ 7}$	$a_{7\ 5}$
⋮				
31	2	0,32	$a_{2\ 1}$	$a_{1\ 2}$
32	3	0,33	$a_{3\ 1}$	$a_{1\ 3}$
33	4	0,34	$a_{4\ 1}$	$a_{1\ 4}$
34	13	0	$a_{13\ 1}$	$a_{1\ 13}$
35	3	0,36	$a_{3\ 2}$	$a_{2\ 3}$
36	13	0	$a_{13\ 2}$	$a_{2\ 13}$
37	4	0	$a_{4\ 3}$	$a_{3\ 4}$
38	6	0	$a_{6\ 5}$	$a_{5\ 6}$
⋮				

* The notation x,y,z represents the successive overwriting of x by y and then by z in the storage location.

Table 6.2 Storage of a sparse matrix

This task uses the number of non zero elements in each column to determine the sequence in which the factorisation is the most economical. It also determines the different elements to be eliminated, processed or inserted.

This process is computationally efficient since it approximately minimises the number of fill-ins and the effort required by the factorisation.

The logical steps involved in processing these two functions column-wise can be summarised as follows:

Select a pivotal column p:
 select processing column k,
  consider pivotal row j within column p, with j different to p and processing row i within column k with i different to k.
  If j is less than i;
   if j and k are equal;
    a diagonal element $a_{kk}{}^{p}$ is to be reduced,
    next j.
   Else (j is not included in k);
    fill-ins a_{jk} and a_{kj} are to be created,
    set fill-in addresses to free locations.
    If created elements are first in their columns;
     set $LNXT$ of the fill-ins to $LCOL$ of their columns,
     set $LCOL$ of the columns to the fill-in addresses.
    Else;
     set $LNXT$ of the fill-ins to $LNXT$ of their preceding elements,
     set $LNXT$ of the preceding elements in the columns to the fill-in addresses,
    Endif.
    Set $ITAG$ of the fill-ins to j and k,
    set next free location to $LNXT$ of the preceding free location,
    increment the number of non zero elements,
    next j.
   Endif.
  Elseif j is equal to i;
   reduced elements $a^{p}{}_{ik}$ and $a^{p}{}_{ki}$ are to be processed,
   next i and j.
  Else;

If i and p are equal;

Elements $a_{pk}{}^{p}$ and a_{kp}^{p} to be deleted.

If the element to be deleted is first in the column k;

set $LCOL$ of k to $LNXT$ of the element to be deleted.

Else;

set $LNXT$ of the preceding element to $LNXT$ of the element to be deleted.

Endif.

Set $LNXT$ of the element to be deleted to the free location LF,

set next free location LF to the address of the element to be deleted,

decrement the number of non zero elements in k by one,

store address of the deleted element in vector NEWZ.

Endif.

Next i.

Endif.

If i and j are last in their columns;

next k.

Elseif i or j is last;

set i or j to large.

Endif.

If last k, and i and j are large;

next p.

Endif.

Where

The pivotal column p is the column with the least number of non zero terms.

Pivotal rows j are the rows included in a pivotal column p.

Processed columns k are the columns corresponding to the pivotal rows j.

The processed rows i are the rows included in the processed columns k.

The definitions of the rest of the variables are given earlier in chapter 3.

This process is illustrated in figure (6.2). This representation is related to the network example given in chapter 3. A fill-in is created if there is no direct connection between the adjacent elements of the eliminated node. These are illustrated by the continuous thin lines. The eliminated elements are represented by the dotted lines. Junctions 1-12 are related to the differential equations of the three generators, and junctions 13-19 are related to nodes 1-7. The word junction is used here to differentiate the actual nodes from the fictitious ones which are deduced from the network representation of the matrix given in figure (3.5) which is a simplification of equation (6.1) where the elements grouped into $2 * 2$ sub-matrices are taken as one element.

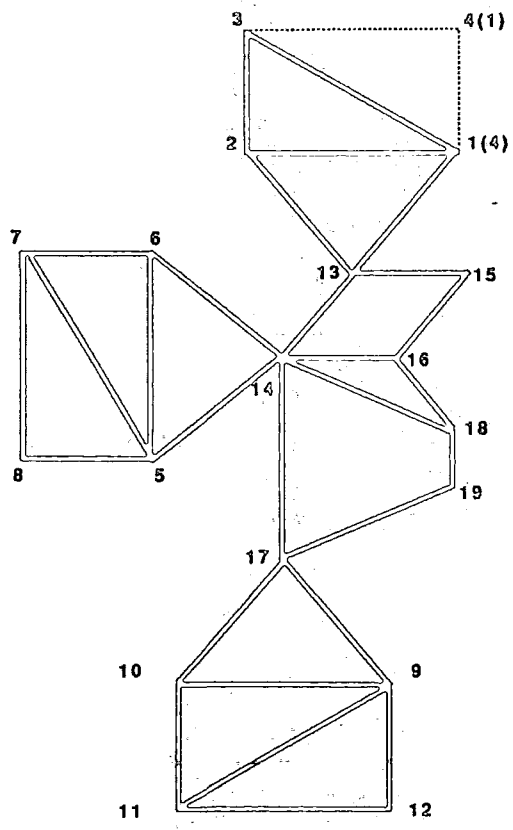
Figures (6.2(a)-6.2(k)) show the reordering and simulation process. Figure (6.3) shows the result of this process in a matrix form. The column numbers illustrate the ordering result. It should be noted that in reality such shifting does not take place. The reordering sequence is stored in an array instead. The crosses represent the $2 * 2$ non zero elements. The circles represent the duplicated terms which have to be deleted after the reordering and simulation is completed. The filled circles show the fill-ins.

The non zero elements of the reduced matrices should all be eliminated at the end of the factorisation. This process is implicit. These elements are simply overwritten by the equivalent factor matrix elements at each elimination step.

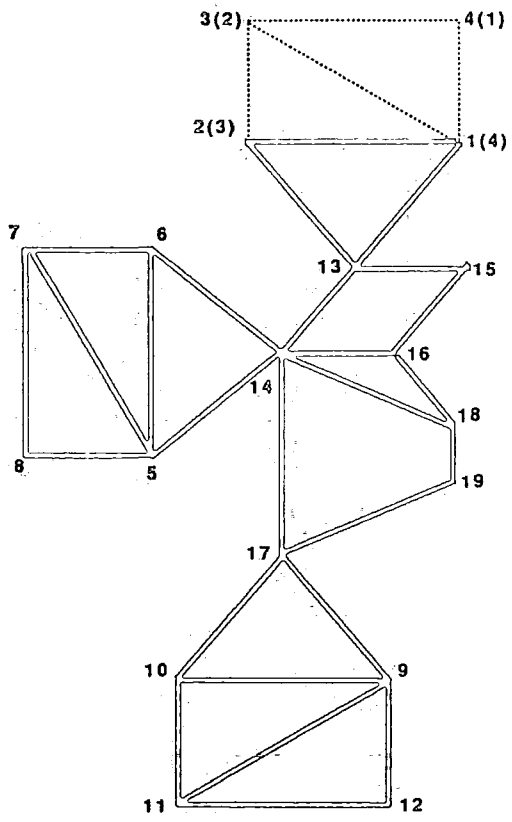
6.3.3.3 Factorisation routine

The factorisation task is greatly alleviated by the simulation described above. The factorised elements can be stored in the predetermined locations without requiring any additional indexing effort. Another advantage of this scheme is the fact that the factor matrix indexing would have to be repeated unnecessarily if it was included with the arithmetical factorisation process which needs repeated processing since the coefficient matrix is constantly changing.

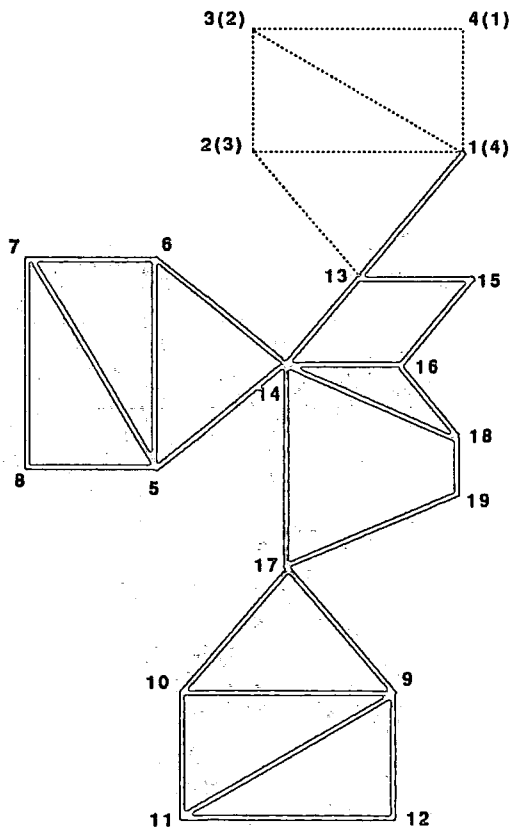
The main function of the factorisation then becomes, the evaluation of the reduced matrix and the factor matrices. By considering the number of factor



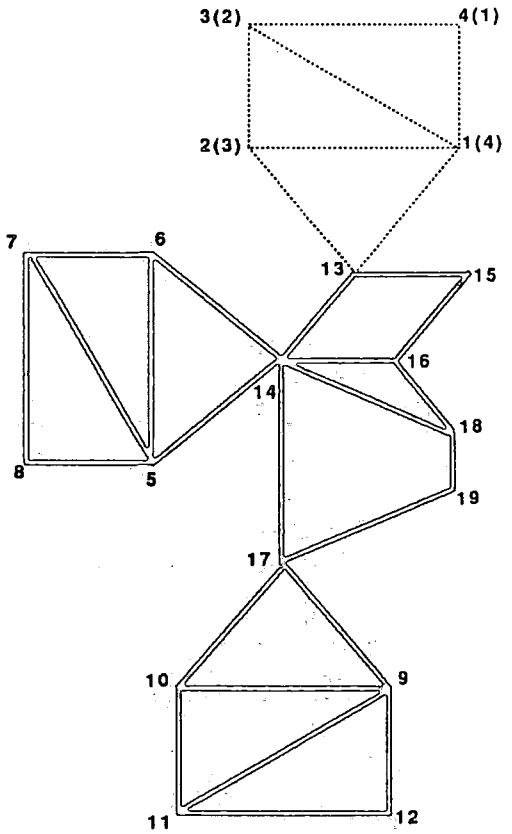
a. Elimination of junction 4



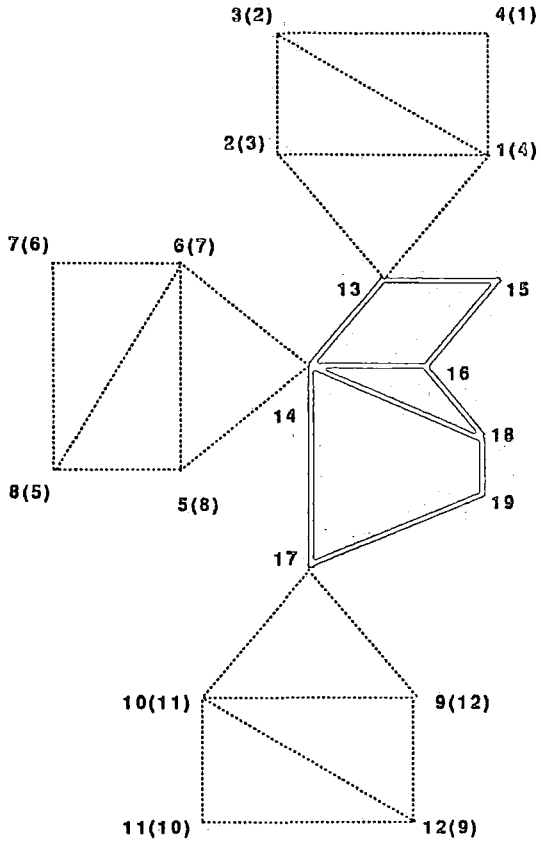
b. Elimination of junction 3



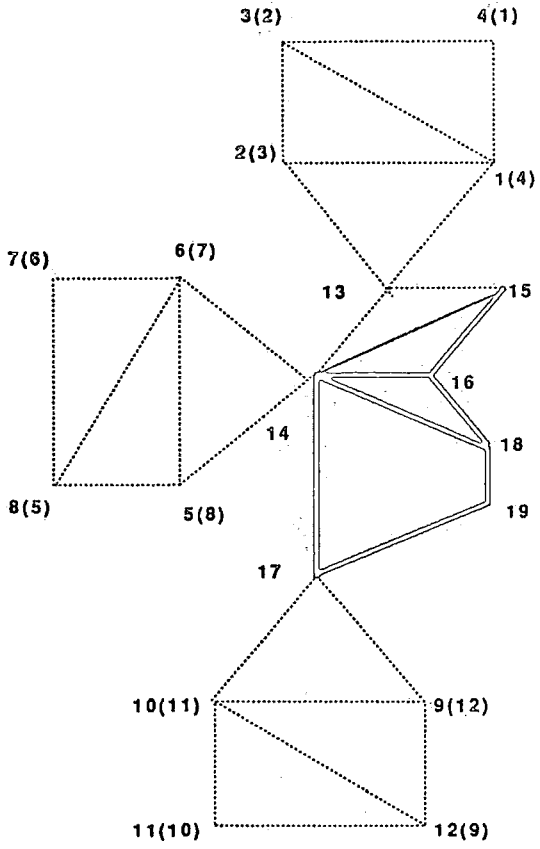
c. Elimination of junction 2



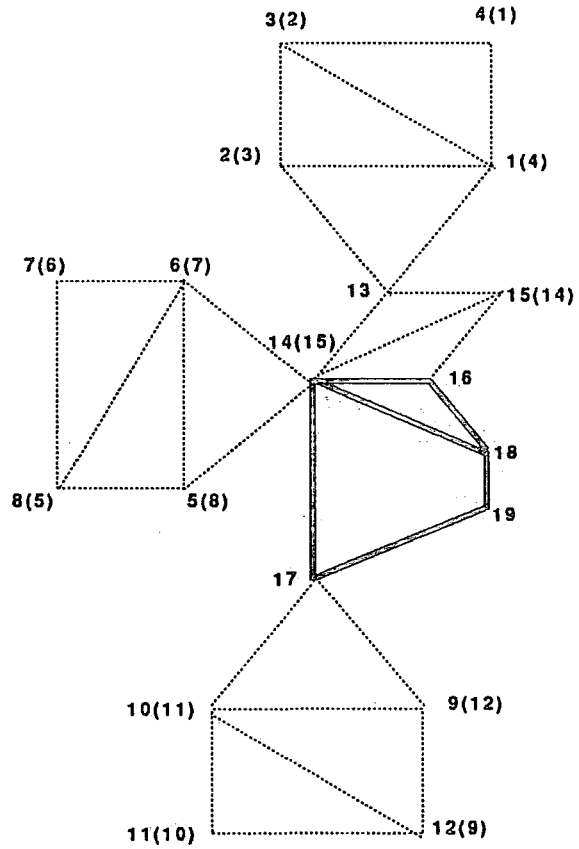
d. Elimination of junction 1



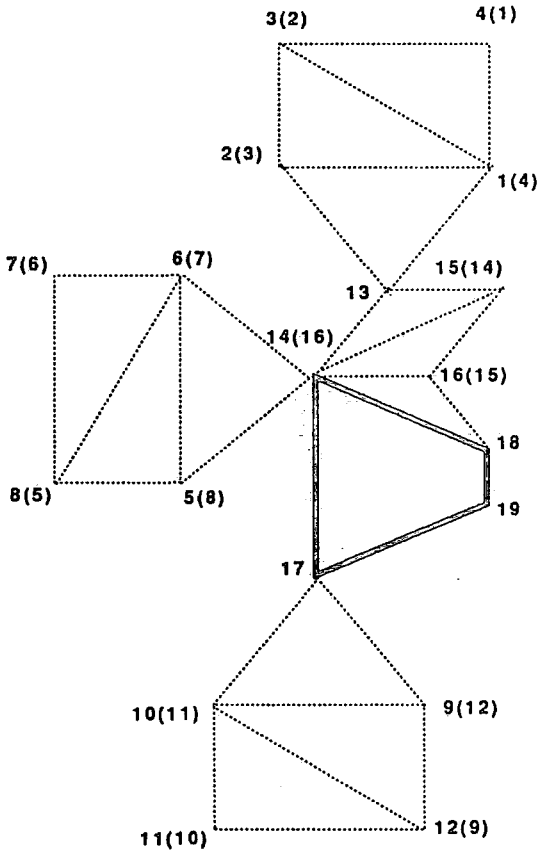
e. Elimination of generators



f. Elimination of junction 13

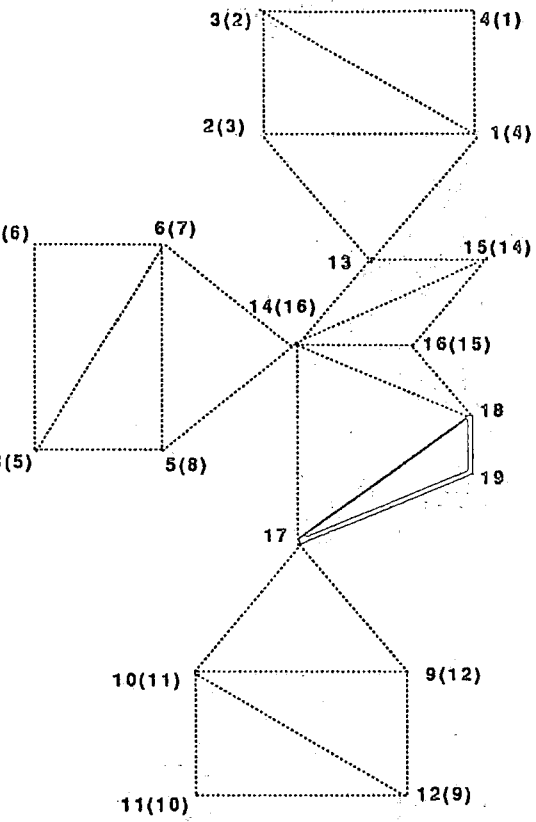


g. Elimination of junction 15

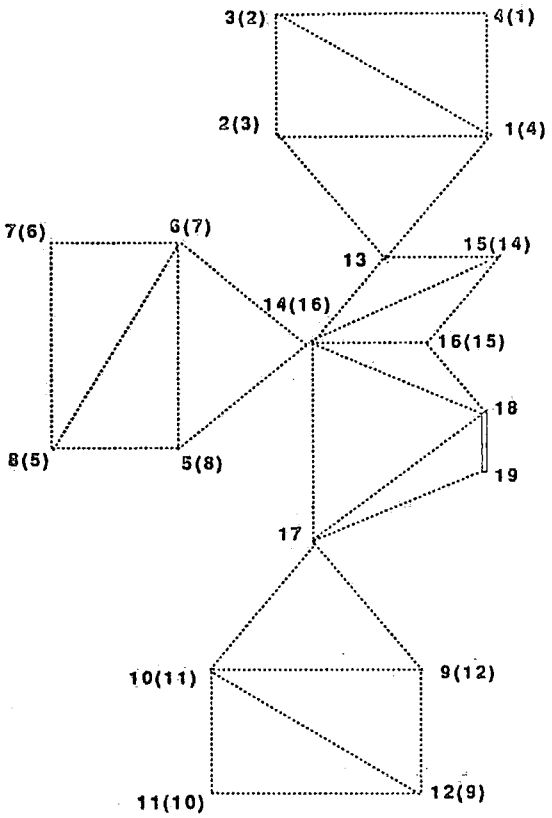


h. Elimination of junction 16

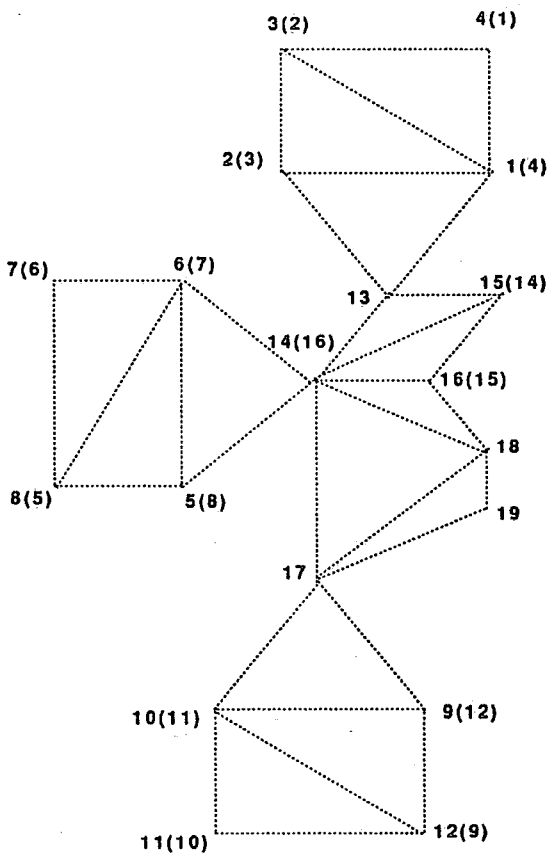
II. Elimination of the generators and nodes 1,3,4



i. Elimination of junction 14



j. Elimination of junction 17



k. Elimination of junction 18

III. Elimination of nodes 2,5,6

Fig. 6.2 Ordering and simulation process

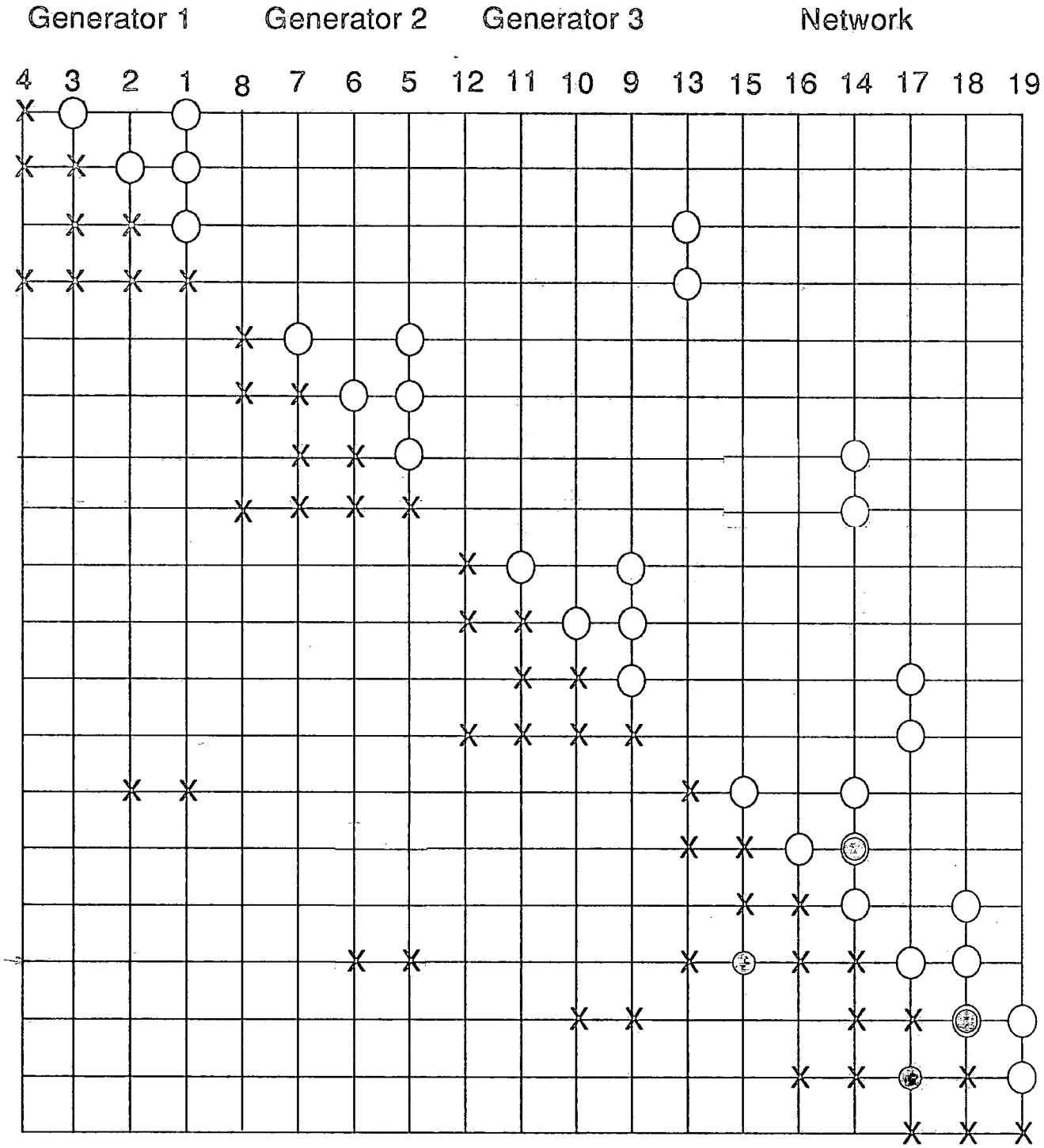


Fig. 6.3 Ordering and simulation of the modified Jacobian matrix related to the example network

matrices involved in the calculation ($2n$) this method may appear to involve more operations than conventional methods. However, most of the elements of these factor matrices need not be processed or stored and the calculated elements need no more operations than for Gauss elimination. For instance, the factor matrices $[C]^p$ (see chapter 3) do not involve any arithmetic operation or wasted storage space since their elements are equal to the negated corresponding elements of the reduced matrix and are stored at the corresponding positions. Deleting the off-diagonal elements from the reduced matrix and setting the diagonal terms to one is implicit. It is achieved by overwriting these quantities with the corresponding elements of the factor matrices. The diagonal elements of the factor matrices $[R]^p$ are also known implicitly. The factor matrices $[D]^p$ involve only one non zero element which is different to unity in the pivoting row.

The factorisation process can be briefly described as follows:

Scan through pivotal column p:
scan through processed column k (varying pivotal and processed rows j and i).
If j is less than i;
if j equals k;
process reduced diagonal element $a_{kk}{}^p$,
next j.
Endif.
Elseif j equals i;
process reduced off-diagonal elements a_{ik}^p and a_{ki}^p,
next j and i.
Else;
next i.
Endif.
If last j and i;
next k.
Elseif last i or j;
set i or j to large.

Endif.
If last k, and i and j are large.
Next pivotal column p.
Endif.

6.3.3.4 Solution routine

The known vector of equation (3.5) has to be multiplied by $2n-1$ factor matrices. Fortunately, these factor matrices are all unit matrices except for one row or one column, and these rows and columns contain only a few non zero elements since the coefficient matrix is very sparse. Also if the zero operations and multiplications by unity are ignored and only the non zero elements are processed, these multiplications become trivial.

The direct solution process can be described as follows:

Consider pivotal column p in an ascending order:
Multiply the known vector of equation (3.5) by the diagonal matrix $[D]^{(p)}$.
Scan through the pivotal column.
Multiply the vector obtained by the factor matrix $[C]^{(p)}$.
Next non zero element in the pivotal column.
Next pivotal column p.
Consider pivotal columns in a descending order:
Scan through pivotal column p.
Multiply the vector obtained by the factor matrix $[R]^{(p)}$.
Next non zero element in the column p.
Next pivotal column p.

6.4 NUMERICAL STABILITY AND ACCURACY

Some networks are difficult to solve numerically. Among these are the ill-conditioned cases in which the difference between the smallest and the largest impedances is very large. In this case the solution is very sensitive to even the computational round off errors. This problem can be alleviated by using

a double precision scheme, and choosing the largest elements as pivots. The Array Processor words are 38 bits long. This gives an effective precision and adequately solves this problem without any additional effort.

An ill-conditioned problem may arise from a combination of line series capacitance and reactance which can result in small diagonal elements. However, with the use of the $2 * 2$ sub-matrices this type of problem is unlikely to arise. In addition, the power system Jacobian matrices have usually predominant diagonal elements. However, if a cancellation between the capacitive and inductive impedances takes place and no resistive elements are modelled, the matrix may become singular. Although, the resonance which is the cause of instability problem is usually avoided in practice. In this case a limit is set for the diagonal elements. If this boundary is violated, they are forced to take the limit value. This involves some inaccuracy of calculation, but since it avoids numerical instability of the solution this compromise is acceptable and is more advantageous than a pivoting search which always considers the stability factor whether the system is well- or ill-conditioned.

6.5 COMPARISON OF ZOLLENKOPF AND HARWELL METHODS

The pivoting adopted in the *Harwell* subroutines considers a compromise between numerical stability and the number of fill-ins. This is included with the factorisation process and carried out at each iteration since the stability scheme depends on the non zero element values.

Since the power systems equations usually involve large diagonal elements compromising between accuracy and number of fill-ins becomes unnecessary. Therefore, the only task of the ordering scheme would be minimising the number of generated elements. This does not depend on the values of non zero elements of the Jacobian matrix, but depends on the number and location of these terms in each column or row. Consequently, this process should be carried out separately from the factorisation which depends on the values of the Jacobian matrix and needs to be evaluated at each iteration. Reference

23 has also dismissed the pivoting to maintain numerical stability and stressed minimising the number of fill-ins.

The *bifactorisation* technique takes advantage of these characteristics, and saves enormous computational time because the bulk of the calculation is spent in performing the matrix reordering.

The use of the $2*2$ sub-matrices reduces the number of integer operations expended in counters and address allocation. This is one of the requirements of the AP [42,52,110].

Another inefficiency of the LU decomposition method used in the *Harwell* routine is that the matrix compacting and indexing are carried out together. No advantage is taken from the fact that the indices remain the same as long as the matrix structure is not changed.

The number of operations necessary for the determination of the unknown vector x is about the same for both methods since they are based on Gauss elimination. However, the *bifactorisation* method is superior to the LU method because it requires about 25 % of the original indexing information if all $2 * 2$ sub-matrices are full.

It can be concluded that *Harwell* library is a commercial general-purpose package and therefore it considers the stability of the system which may be important for certain ill-conditioned problems. The *bifactorisation* method can be considered as an efficient technique to solve sets of power system linear equations which are usually well-conditioned or any system which exhibits similar characteristics.

CHAPTER 7

DISTRIBUTED SIMULATION

7.1 INTRODUCTION

The distributed nature of power systems encourages the use of decomposition methods in power system analysis and control. For large integrated systems using local computers linked to a master processor the implementation of system decomposition is essential for efficient and fast operation and control. The motivation behind the use of these techniques is the continuous expansion of power systems and the limited capabilities of the available computers in terms of both speed and storage.

Kron was one of the pioneers in this field [51,102]. He developed the diakoptics method. This method consists of solving each of the sub-problems independently and combining the partial results to obtain the overall solution of the system. This technique has quickly been adopted in many fields of power system control and operation, and has been applied in practice to most of the power system control and analysis tasks such as load-flow, economic dispatch, state estimation etc. [51,53,102]. The development of relatively cheap microprocessors has provided researchers with an economical and efficient means to be used for multiprocessing.

In order to apply a decentralised technique to a given problem, the system should be decomposable as well as the algorithm used. Some algorithms can be very efficient if applied to a single processor, but, this does not necessarily imply that they would be efficient for decomposed schemes. Fortunately, the Newton-Raphson method is efficient in both cases although more iterations may be needed in the case of distributed solutions especially in the case of the

solution of non-linear systems [28,102,106]. However, this may not be important if the computational time is decreased significantly. The decentralised Newton-Raphson technique is widely used in the decoupled load flow studies [35,53,91,107]. The *bifactorisation* technique is also applicable to decomposed systems since it is concerned with the solution of a given set of linear equations which may arise from a system or a subsystem model. Reference 14 presented a decomposed LU triangular sparsity technique which includes both area sub-matrices and tie line elements. The LU triangular algorithm used in the decomposed simulator presented in reference 92 solves the areas only. The overall solution is obtained by combining the area results with the effects of the tie lines using the *Householder* method.

Parallelism is subject to saturation constraints [28,80] i.e. if more than a specific number of processors is used to solve a problem the marginal speed-up diminishes. This can reach a point where the overall speed decreases. This limitation may be attributed to the prevalence of the decomposition penalties and overheads over the actual processing time. In a discussion of the work of Happ et al. [53] it has been strongly emphasised that the use of optimal methods for tearing networks is necessary for efficient and economical results [67]. Automatic methods should also be used to select high impedance connection branches to aid Newton-Raphson convergence. However, the authors of this paper [53] disagreed with the last suggestion and claimed that no restrictions on the tie line parameters are required.

Ideally a decomposed solution is supposed to be speeded up by a factor equal to the number of processors. Unfortunately, this is not the case because the solution is subject to decomposition penalties. The main penalties are caused by:

(1) the additional code necessary to coordinate and synchronise the area programs and compute the overall solution from the individual results,
(2) the impossibility of finding an ideal network split which would take exactly the same execution time in each area processor and not involve

any idle time during which the machines solving small areas are in a waiting mode,

(3) the overheads required by data communication between processors, and

(4) the synchronisation of the different tasks also involves a penalty since this requires the use of waiting routines.

An efficient decomposed method is one which minimises these supplementary times and improves the performance ratio which can be expressed as $(\frac{single\ processor\ time}{decomposed\ time})$. The overheads involved in the calculation of the overall solution can be significantly reduced if the number of tie lines are minimised. The uneven split of the network problem can be alleviated by using optimal methods to subdivide the network as evenly as possible [67]. The data transfer problem depends mainly on the type of algorithm used (tightly coupled or loosely coupled) and on the amount of data to be transferred. The overheads involved in the tightly coupled algorithms are usually trivial since the data is acquired through shared memories. The loosely coupled overheads are more significant since hardware links between the different machines are used. This requires supplementary computer coding to perform the data transfer. In this case the minimisation of this time requires the minimisation of the amount of data to be transferred, the use of fast communication links such as Ethernet as well as the reduction of the number of program interruptions. As far as the waiting routines are concerned, the selection of an optimum waiting time is essential.

This work on parallel processing has been performed by decomposition into hypothetical parallel processors which are simulated on a uni-processor (PE 3230 minicomputer). This is supposed to provide a good estimate to see whether it is economical to use parallel processing or use a powerful single machine. Only one computing code is used for all areas since the different processors are assumed to be identical. This greatly reduces the programming effort. The decomposed simulator presented here exploits the benefits of both parallelism as well as the powerful methodology which uses the sparsity of matrices (the *bifactorisation* technique).

The decomposed simulator is based on a principle similar to that presented by Hatcher et al. [54]. In this reference various algorithms are used to solve linear equations. Among these the Newton-Raphson method and Gauss-seidel technique. The former has been selected because it has a better rate of convergence and is already in use in the centralised version on which the decentralised simulator is based.

7.2 NETWORK TEARING

To illustrate network tearing an example of splitting a system into two subsystems linked via a tie line is given in figure (7.1). The removed intersection is represented by current sources connected to each area. The intersection network is represented by a voltage source linked to the tie line admittance. Details of network tearing using matrix algebra is well presented in reference 10.

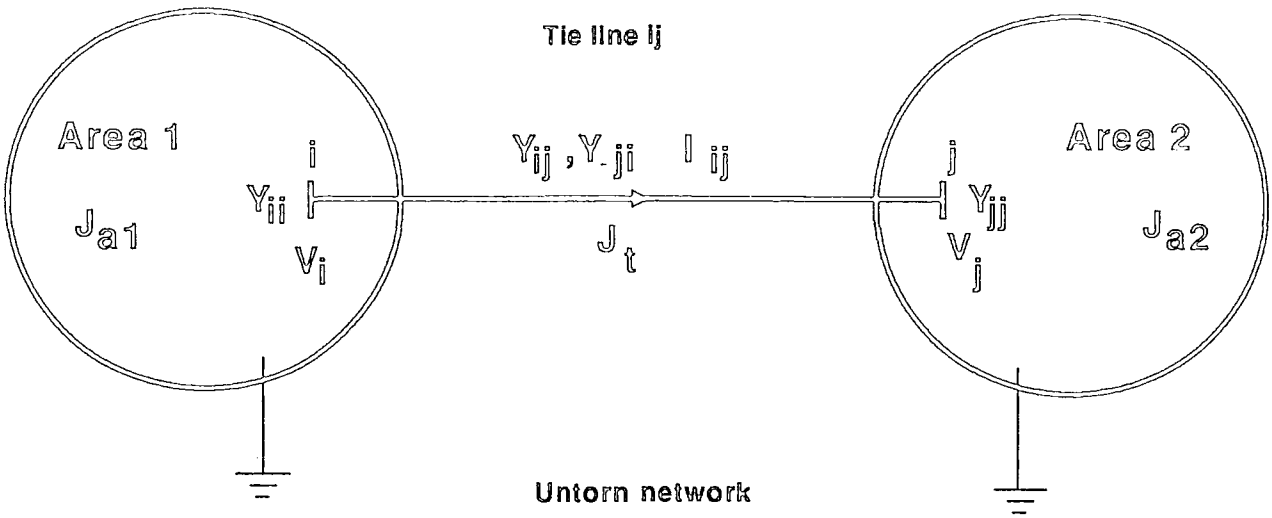
The area static and dynamic models are described in chapter 4 and their corresponding Jacobian matrices has the same structure as equation (4.104). The only exception is that the tie node currents are evaluated separately in the main processor and transferred to the areas where these injected currents are added to the relevant tie nodes. In this respect, by network analysis of the example of figure (7.1), the tie node algebraic equations derived from Kirchhoff's first law can be expressed as follows

$$\sum_{k=1}^{n} \bar{Y}_{ik}(\bar{V}_i - \bar{V}_k) + jS_{ci}\bar{V}_i = \bar{I}_i + \bar{i}_i \qquad (7.1)$$

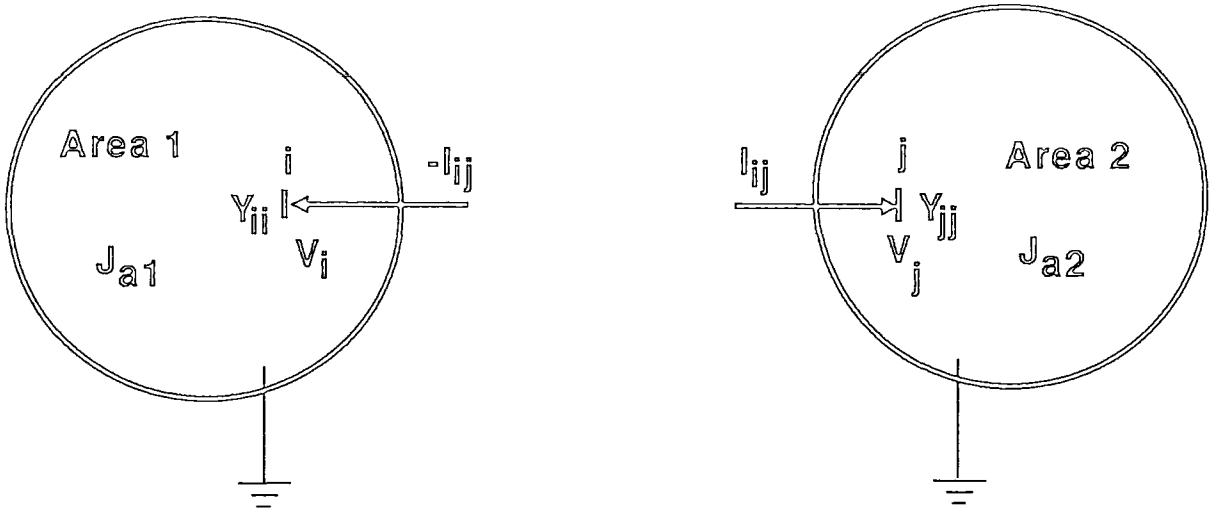
$$\sum_{l=1}^{m} \bar{Y}_{jl}(\bar{V}_j - \bar{V}_l) + jS_{cj}\bar{V}_j = \bar{I}_j + \bar{i}_j \qquad (7.2)$$

Where

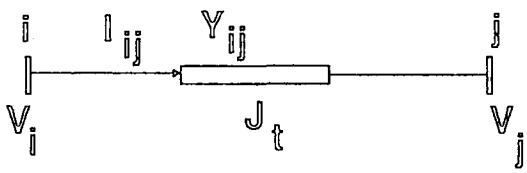
$\bar{Y}_{ik}$ and $\bar{Y}_{jl}$ are the admittances of the tie nodes included in area 1 and area 2 respectively excluding the tie lines.
S_{ci} and S_{cj} are the tie line susceptances at node i and j respectively.
$\bar{V}_i$ and $\bar{V}_j$ are the voltages at tie nodes i and j respectively.



Untorn network



Subnetworks



Intersection network

Fig. 7.1 Equivalent torn electrical networks

$\bar{V}_k$ and $\bar{V}_l$ are the voltages of the nodes linked to tie nodes i and j respectively.

$\bar{I}_i$ and $\bar{I}_j$ are the injected currents at tie node i and j respectively excluding the tie lines.

$\bar{i}_i$ and $\bar{i}_j$ are the tie lines injected currents at tie nodes i and j respectively.

n and m are the number of nodes linked to tie node i and j respectively.

Currents $\bar{i}_i$ and $\bar{i}_j$ are equal in magnitude. They are expressed by the following equations related to the intersection sub-network

$$\bar{I}_{ij} = (\bar{V}_i - \bar{V}_j)\bar{Y}_{ij}$$

$\bar{i}_i = -\bar{I}_{ij}$, and $\bar{i}_j = \bar{I}_{ij}$

Differentiating the complex components of currents $\bar{i}_i$ and $\bar{i}_j$ with respect to the complex components of voltages $\bar{V}_i$ and $\bar{V}_j$ will lead to a sub-Jacobian matrix, similar to equation (4.98) corresponding to line models with the exclusion of the tie lines charging. This is included in the area tie node models given in equations (7.1) and (7.2).

The overall Jacobian matrix can be obtained by sequentially positioning the different area sub-matrices and linking them through the intersection sub-matrix. The Jacobian matrix can therefore be subdivided into area sub-Jacobians and the intersection sub-Jacobian as follows

$$[J] = [J_a] + [J_t] \tag{7.3}$$

For symmetrical matrices $[J_t]$ can be expressed as follows [10,102,116]

$$[J_t] = [M][\bar{Y}]_{ij}[M]^t$$

Where

[M] is an $n * 2m$ connection matrix consisting of 1's and -1's for the sending and receiving ends of the torn branches respectively. m is the

number of tie lines in the system and $n = n_d + 2n_N$ is the dimension of the system, with n_d the number of differential equations of all active generators and n_N the number of nodes. If the $2 * 2$ sub-matrices algorithm is used n would be halved.

$[\bar{Y}]_{ij}$ is the tie lines admittance.

t denotes the matrix transpose.

Since the line sub-Jacobians are not symmetrical, by matrix analysis and inspection of the Jacobian matrix and its structure, reference 92 suggests the following way of analysing the intersection network and the connecting matrix $[M]$.

$$[J_t] = [M][T]^{-1}[M^{-1}]^t \tag{7.4}$$

The elements of matrix $[T]$ depend on the tie line parameters but, they are different to their admittance matrices. The sub-Jacobian matrix $[J_t]$ elements include the tie line admittances $\bar{Y}_{ij}$. The elements of matrix [T] are chosen so as to satisfy the above equality and reduce the dimension of the tie lines model.

The $2m * 2m$ inverse matrix $[T]^{-1}$ for m tie lines includes only diagonal $2 * 2$ elements and is given by:

$$[T]^{-1} = \begin{bmatrix} 2B_1 & B_1 - G_1 & & & & & & \\ G_1 - B_1 & G_1 & & & & & & \\ & & \ddots & & & 0 & & \\ & & & 2B_k & B_k - G_k & & & \\ & & & G_k - B_k & G_k & & & \\ & & 0 & & & \ddots & & \\ & & & & & & 2B_m & B_m - G_m \\ & & & & & & G_m - B_m & G_m \end{bmatrix} \tag{7.5}$$

The dimension of the connecting matrix [M] $(n * 2m)$ is chosen in such a way that the sub-Jacobian $[J_t]$ would have the same dimension as the diagonal sub-Jacobian $[J_a]$. For an asymmetrical system the matrix $[M]$ is given by

$$[M] = \begin{array}{c} \\ \\ tie \\ node\ i \\ \\ tie \\ node\ j \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} tie\ 1 & \dots & tie\ k & \dots & tie\ m \end{array} \\ \begin{pmatrix} \vdots & \vdots & [0] & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & 1 \quad 1 & \vdots & \vdots \\ \vdots & \vdots & 0 \quad 1 & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & -1 - 1 & \vdots & \vdots \\ \vdots & \vdots & 0 - 1 & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & [0] & \vdots & \vdots \end{pmatrix} \end{array} \tag{7.6}$$

The position of the non zero elements related to a tie line k linking tie nodes i and j located in area b and c respectively is deduced from this matrix and the transpose of its inverse. Area $\overset{1}{b}$ (node i) is supposed to be the sending area and area $\overset{2}{c}$ (node j) is assumed to be the receiving area.

In order to determine the state variables, the inverse of the overall Jacobian matrix is calculated. Since this matrix is decomposed into two terms the *Householder* lemma is used. After manipulation this can be formulated as follows

$$[J]^{-1} = [[J_a] + [J_t]]^{-1}$$

$$[J]^{-1} = [J_a]^{-1} - [J_a]^{-1}[M]\{[T] + [M^{-1}]^t[J_a]^{-1}[M]\}^{-1}[M^{-1}]^t[J_a]^{-1} \tag{7.7}$$

The overall Jacobian matrix $[J]$ of the torn system has a different structure to the Jacobian matrix of the untorn system. In the single processor solution all the generator dynamic sub-Jacobians are positioned in sequence and are followed by the network elements. In decomposed systems the Jacobian matrix of each area is represented by a similar structure to the overall solution. These appear sequentially in ascending order. The area matrices are in the form of diagonal blocks linked through tie line elements.

7.3 ALGORITHM ORGANISATION

Two slightly different algorithm implementations are presented. The first implementation differs from that of reference 92 in the technique used for solving the sparse sets of linear equations. This algorithm is completely independent of the OCEPS package. The second algorithm interacts with the existing software facilities such as the load-flow package, the loader, etc. This strategy enables the simulator to be considered as an integral system with the Energy Management Software. And with some further modifications this package could readily be used on line. These two algorithms are shown in the two flow charts given in figures (7.2) and (7.3).

The two algorithms are based on the same principle. Therefore no improvements in speed or accuracy are expected. The only advantage from using this scheme is to exploit the potential of the decomposed simulator in interfacing with the OCEPS software. A new supporting program which divides the physical data among the areas and the main program has been developed. The amount of the initial data included in the data communication routine and sent from the new supporting program is larger than that used previously. However, since this is carried out only once for each simulator run, this approach is acceptable.

Compared to the single processor simulator the present decentralised scheme is expected to have an accuracy of the same order to within the round off errors, since they are both based on the same principle. However, the number of iterations may be different to that of the uni-processor algorithms, since to confirm convergence all the linear equations and state variables are checked against tolerances each time unless the linear functions are within predetermined limits. Therefore, even a small variation of one single quantity may lead to an extra iteration.

Another inefficiency of the present decomposed algorithm is that it forces each processor to iterate until all processors have converged. This is because

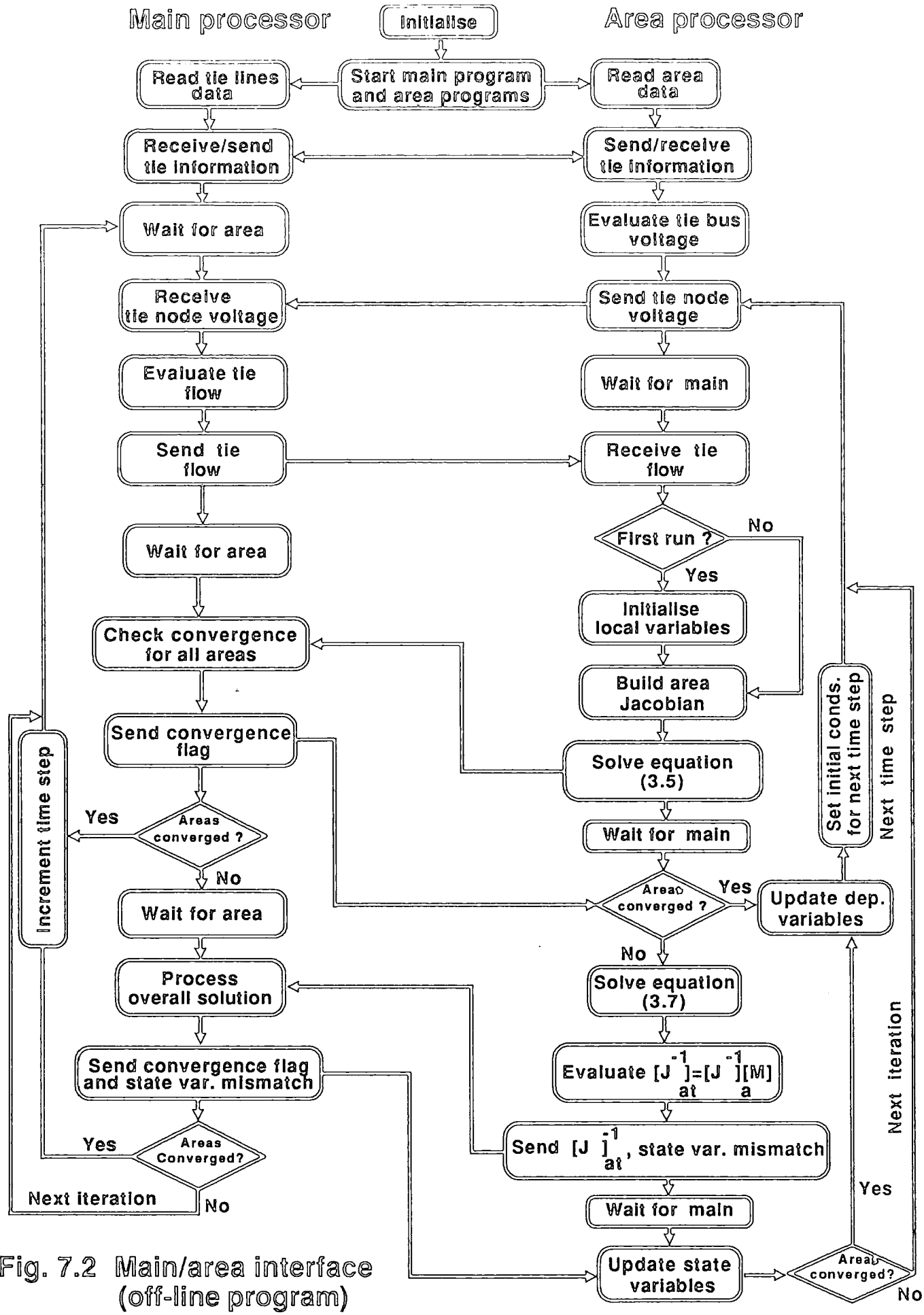
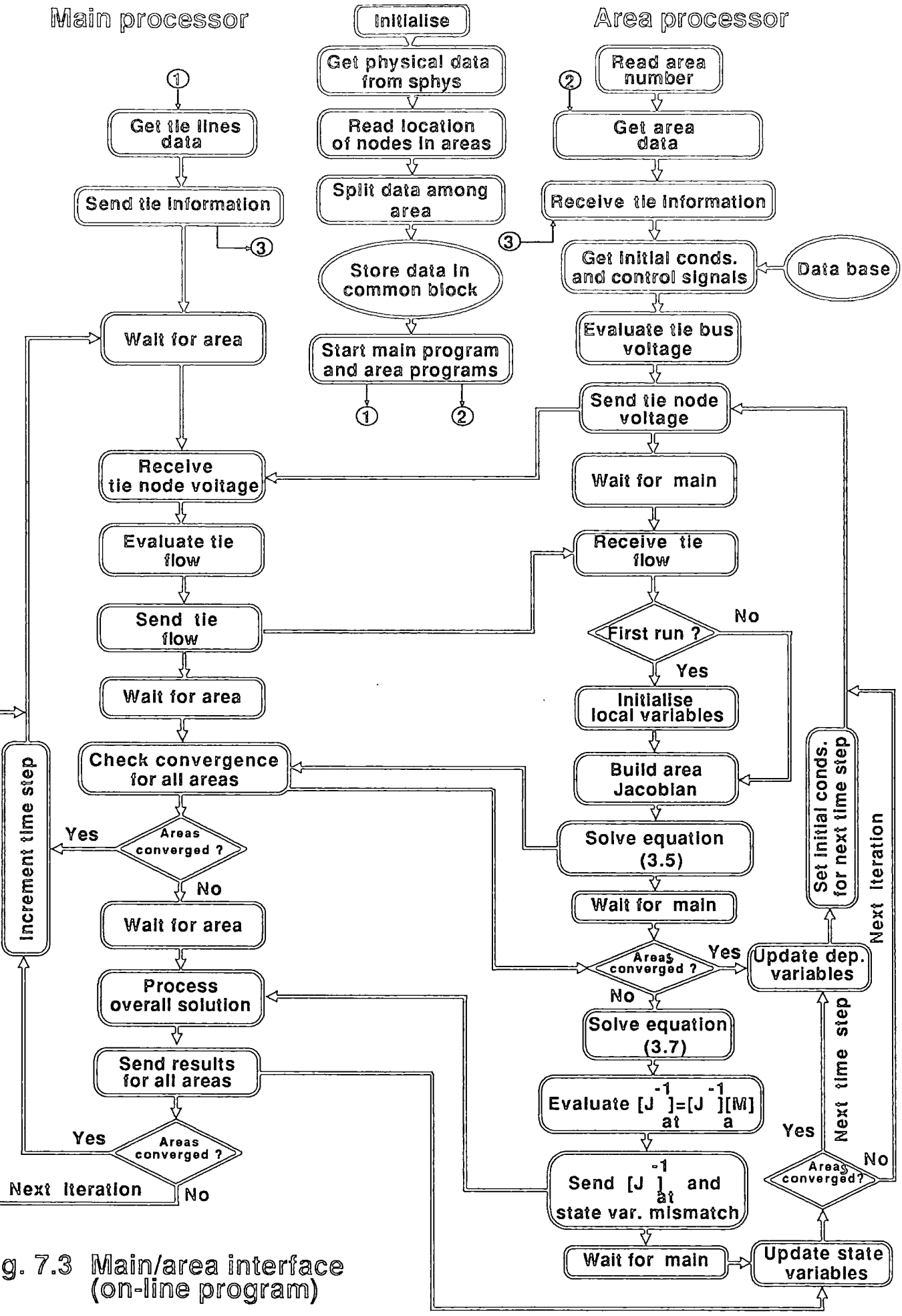


Fig. 7.2 Main/area interface (off-line program)



g. 7.3 Main/area interface (on-line program)

the time step incrementing is initiated by the main processor upon convergence of all processors.

7.4 DATA INITIALISATION

Previously, each area reads its dynamic and static data from an independent data file. The initialisation of the control signals, and the bus voltages and power flows was performed separately by initially running a control task and a load flow package. In order to develop an integrated decomposed simulator which can take advantage of the existing support packages, the physical data available within the OCEPS suite has been used. Extra computing code is used to subdivide the data of the overall system between the different areas according to their topology. The tie line data is communicated to the main computer where the coordination between areas is performed. The only additional data needed is the allocation of the nodes to their different areas. This task is run once for each simulation process. The data is accumulated in a common block which can be accessed by both the main and the communication programs. Also the initialisation of the bus voltage and the power flow etc. is done in the same way as the decentralised and parallel simulators. These packages evaluate the necessary initial variables which are stored in specific common blocks before the simulator starts running. Each area receives the appropriate variables to determine the local static and dynamic initial conditions as described in chapter 5. This process is illustrated in figure (7.3).

7.5 DATA COMMUNICATION AND SYNCHRONISATION

The adopted approach requires communication between the areas and the main computer only. No inter-area communication is involved since the areas are considered as totally independent entities. The only common variables are the tie line currents and power flows. These are evaluated in the main program as functions of the parameters of these branches and the tie node voltages (received from the areas). Then the relevant variables are sent to the appropriate area at the required time through a common block.

One of the disadvantages of this decomposed scheme is that it does not fully exploit parallelism (the areas and the main programs are solved sequentially). The area programs are solved in parallel, but, while the coordinating program is carried out all the areas are idle waiting for the main processor response. A waiting state is also imposed on this device when the area tasks are processed. It is hoped that the idle time can be used in performing some of the tasks which are not yet included in the decomposed simulator such as telemetry and measurements

The communication process between the main processor and two areas after the initialisation is illustrated in the diagram given in figure (7.4).

A number of interrupts are introduced in the area programs. The first one is to send the tie node voltages and receive the tie line power flows. The second is to send the convergence flags from the Newton-Raphson function calculation to allow a check on the convergence of all the areas. The third is for sending the state variable mismatch and the Jacobian elements related to the tie nodes and receive the corresponding overall solution and the convergence flags. This process is shown together with the area and the coordinating schemes in figures (7.2) and figure (7.3).

The amount of data to be communicated between the two levels of the decomposition hierarchy is strongly dependent on the number of tie lines, the size of the simulated network and the number of interruptions included in the algorithm. Therefore, the reduction of these cut branches and halts is highly recommended.

The synchronisation between the coordinating and the area processors is ensured through sending and receiving special flags which enable the task to start a process or to wait for the information. These flags are communicated together with details on the progress of the calculation. Therefore, the data communication procedure plays a double role of transferring the data and controlling the calculation flow and sequence of the areas and the main processor.

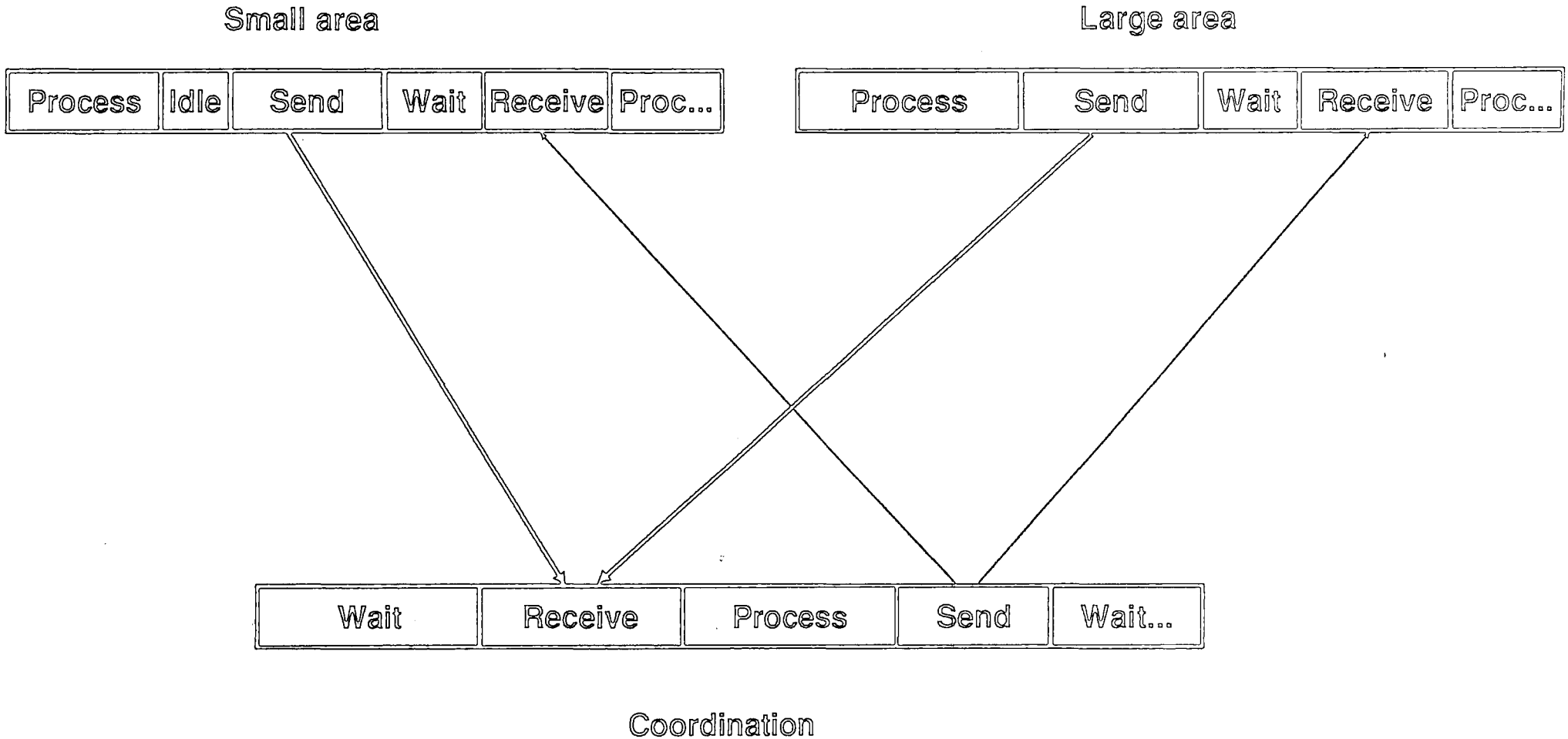


Fig. 7.4 Data communication and synchronisation of processors

7.6 AREA SIMULATION

The global area Jacobian matrix $[J_a]$ is readily divisible between n areas. The number of areas depends on the number of available processors and is limited by the saturation constraints. This matrix can be rewritten as follows

$$[J_a] = \begin{bmatrix} [J_{a1}] & & & \\ & \ddots & 0 & \\ & & [J_{ai}] & \\ & 0 & & \ddots \\ & & & [J_{an}] \end{bmatrix} \tag{7.8}$$

Since this matrix has a block diagonal structure its inverse is equal to the inverse of the sub-matrices. Therefore, applying the Newton-Raphson algorithm (equation 3.7) to evaluate the area state variables would result in the following expression for a 2 area system

$$\begin{bmatrix} [\Delta S_1] \\ [\Delta S_2] \end{bmatrix}_k = -\begin{bmatrix} [J_{a1}]^{-1} & 0 \\ 0 & [J_{a2}]^{-1} \end{bmatrix}_k \times \begin{bmatrix} [F_1] \\ [F_2] \end{bmatrix}_k \tag{7.9}$$

This shows that the individual area models are identical to the original matrix (equation (3.7)) with $[\Delta S_i]$ equivalent to $[[\Delta Y], [\Delta X]]_i^t$ and $[F_i]$ equivalent to $[h([Y]^t, [X]^t)], [g([Y]^t, [X]^t)]_i^t$. The structure of each of the area sub-matrices $[J_{ai}]$ is given by equation (4.104) where the dynamic equations of the area are followed by its network equations.

Each of these sub-problems can be solved separately in the same way as the untorn network solution without including any extra matrix manipulations. Therefore, the solution of the above equations can be obtained by using the modified Zollenkopf *bifactorisation* algorithm described in the previous chapter.

The bus voltage is calculated in the area processors as a state variable. The tie nodes voltages are selected and sent to the main computer where the tie line currents and power flows are evaluated. In order to simulate the network models properly using Kirchhoff's law, these currents should be transferred to the area program and added to the tie node injected currents.

7.7 COORDINATING ALGORITHM

The execution time of the main program is significantly dependent on the number of tie lines as stated earlier. Therefore, the minimisation of these cut branches is one of the most efficient ways of reducing the main program processing time.

This algorithm is meant to process the intersection subsystem which includes the tie lines linking the different areas. This approach allows the areas to be treated as small sub-networks independent from each other. The communication between these area processors via tie line variables is indirect since it is fully accommodated by the coordinating program.

The coordinating processor has three tasks

(1) it calculates the overall solution of the network using the partial solutions of the areas and the intersection sub-networks,
(2) it distributes this response among the areas,
(3) and it synchronises the area and the main processors by waiting or forcing area programs to wait.

7.7.1 Determination of the overall solution

Equation (7.7) gives the inverse of the Jacobian matrix [J] in terms of the inverse of the area sub-Jacobians, the tie line matrix [T] and the connecting matrix [M].

The state variables of the overall system at iteration $k+1$ can thus be determined as follows

$$[J]_k^{-1}[F]_k = [J_a]_k^{-1}[F]_k - [J_a]_k^{-1}[M]_k[T']_k^{-1}[M^{-1}]^t{}_k[J_a]_k^{-1}[F]_k \qquad (7.10)$$

Or

$$[\Delta S]_k = [\Delta S_a]_k - [J_{at}]_k^{-1}[T']_k^{-1}[M^{-1}]^t{}_k[\Delta S_a]_k \qquad (7.11)$$

Where

$[\Delta S]$ is the overall network state variable corrections corresponding to $[\Delta Y]$ and $[\Delta X]$ given in chapter 3,

$[\Delta S_a]$ are the n area state variable corrections,

$[J_{at}]^{-1} = [J_a]^{-1}[M]$ consists of the area non zero elements related to the tie nodes,

$[T'] = [T] + [M^{-1}]^t [J_{at}]^{-1}$

The adopted algorithm imposes the communication of the area state variables $[\Delta S_{ai}]$ and $[J_{ati}]^{-1}$ to the coordinating program to process the overall solution. This procedure is undesirable because the overheads necessary for this transfer may reach unacceptable values for large networks. Thus restricting the size of the systems to be solved.

The matrices included in the main program are very sparse. In this case only the non zero terms are used in the calculation. These are identified by their addresses. The multiplication of matrix $[J_{at}]^{-1}$ by the transpose of the inverse of the connecting matrix [M] is reduced to a simple selection of the non zero elements related to the tie nodes, and matrix $[T']$ is reduced to a summation of this resulting matrix with matrix $[T]$.

In order to distinguish the different area vectors, $[\Delta S_{ai}]$ is stored with area number indices as $2*n$ arrays. These are transferred to their corresponding areas whenever an overall solution is performed.

7.8 HARDWARE SUPPORT

As stated earlier this package has been run on a dedicated single Perkin Elmer 3230 minicomputer. This 32 bit machine is provided with a multi-task shared memory operating system which permits the execution of up to 255 tasks at a time [86]. Actually, the instructions are executed sequentially during short time-slices with the appearance that each of the tasks is solved exclusively, especially if the time slices are frequent and the number of tasks is small. This operating system also allows inter-task communication and control. Therefore,

the machine is actually transferring attention from one task to another until all the areas are processed or an interruption is encountered. This strategy is not desirable for timing the different tasks since too much overhead is involved. Hence, the computer is forced to process one area at a time until a halt command is issued. The areas are processed according to their ascending order. The data is communicated to the main task when all areas are in an interrupt state i.e. the same calculation is carried out for each of them. This process is sketched for three areas in figure (7.5). In this figure it is assumed that area 1 is the largest area and area 2 is the smallest.

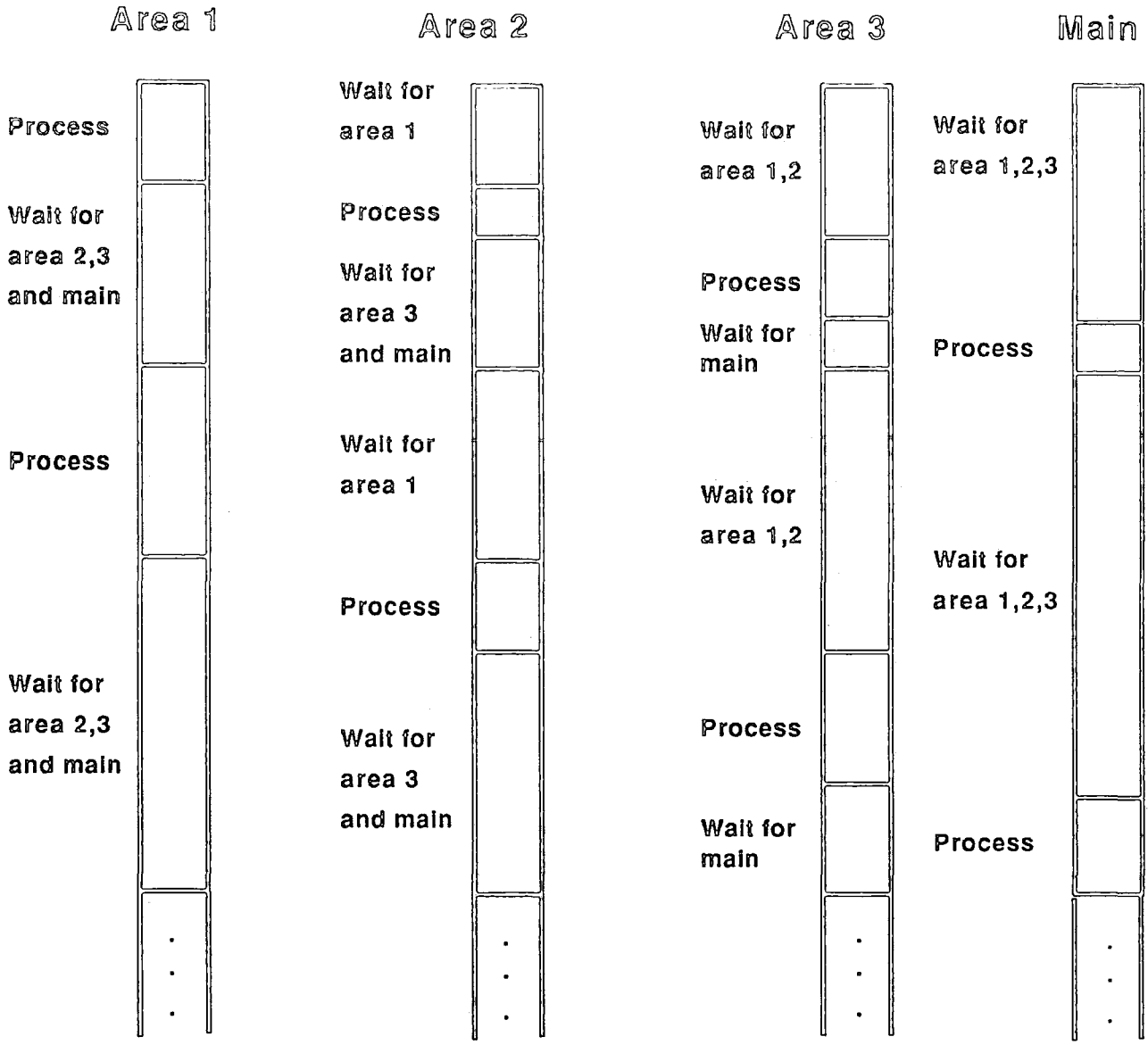


Fig. 7.5 Decomposed process for timing

CHAPTER 8

RESULTS AND COMPARISON

8.1 INTRODUCTION

Usually power systems are capable of surviving most contingencies. Occasionally following a major disturbance heavy stress is imposed on the power system leading to large variations in frequency, voltage, and heavy loading. This may result in cascading which can lead to the separation of the system into islands and the loss of consumer load. Cascading is a sequence of failures or outage of important elements of the system. The initiating event may be caused by false operation of the equipment such as the relays, or natural such as short circuits caused by storms, lightening etc., or through human intervention during tests, and maintenance or even a false diagnosis of the fault.

This chapter is intended to present the long-term simulator efficiency and robustness under such circumstances. Many tests have been carried out to highlight the behaviour of the simulated network by plotting out selected variables relevant to each of these tests.

The OCEPS standard version is used as a benchmark against which the performance of the present algorithm is compared and validated. A realistic validation necessitates the use of actual recorded data. For this purpose a study of the 1981 south of England incident has been carried out [68]. The results obtained were satisfactory. Therefore, the analysis of the present results and their comparison with the earlier ones obtained from the standard package provides a realistic means of validating the present simulator [68,93].

Most of the trend plots which are presented are the result of simulating the modified IEEE 30 node test network on the Array Processor using a time step of 1 second and tolerances of 0.001 (p.u.), for the state variable mismatch, and 0.005 (p.u.), for evaluating the linear functions. To test the accuracy of the results obtained, smaller time steps and tolerances have also been used. To ensure the numerical stability of the Newton-Raphson algorithm under severe events a time step of no more than a quarter of a second is necessary, during critical periods. Some results are obtained by running the parallel simulator as a stand-alone package. Others are obtained by running the simulator with the control package. The latter are called controlled responses (controlled simulator) only to differentiate these two sets of results.

For speed tests the centralised, parallel, and decomposed simulators have been implemented on the IEEE 30 and 118 node systems using a time step of 1 second. To highlight the performance of the developed algorithm for smaller time steps the parallel simulator has also been applied to the 141 bus southwest CEGB area with a time step of a quarter of a second. Figures (8.1)-(8.4) show the 30 and the simplified 118 node systems split into different areas.

8.2 EFFECT OF LOAD VARIATION ON GENERATING UNITS

To illustrate the reaction of the generating units to sudden load variation, load 4 has been disconnected. Since this load is one of the largest loads in the system its impact on the different components of the network would be significant and noticeable.

The scenario starts at midnight where the load is decreasing towards its daily minimum. The effect of load decrease can be seen in the smooth reduction of the generator power outputs (figure 8.5) and the slight increase of the frequency (figure 8.7) to match the new load level and preserve the balance generation/load. Generator controllers must follow closely the variation of the load although the operating constraints of certain elements such as boilers slow down their response. The frequency variation is first detected by the governor which acts on the turbine inlet valve. The decrease in this valve position

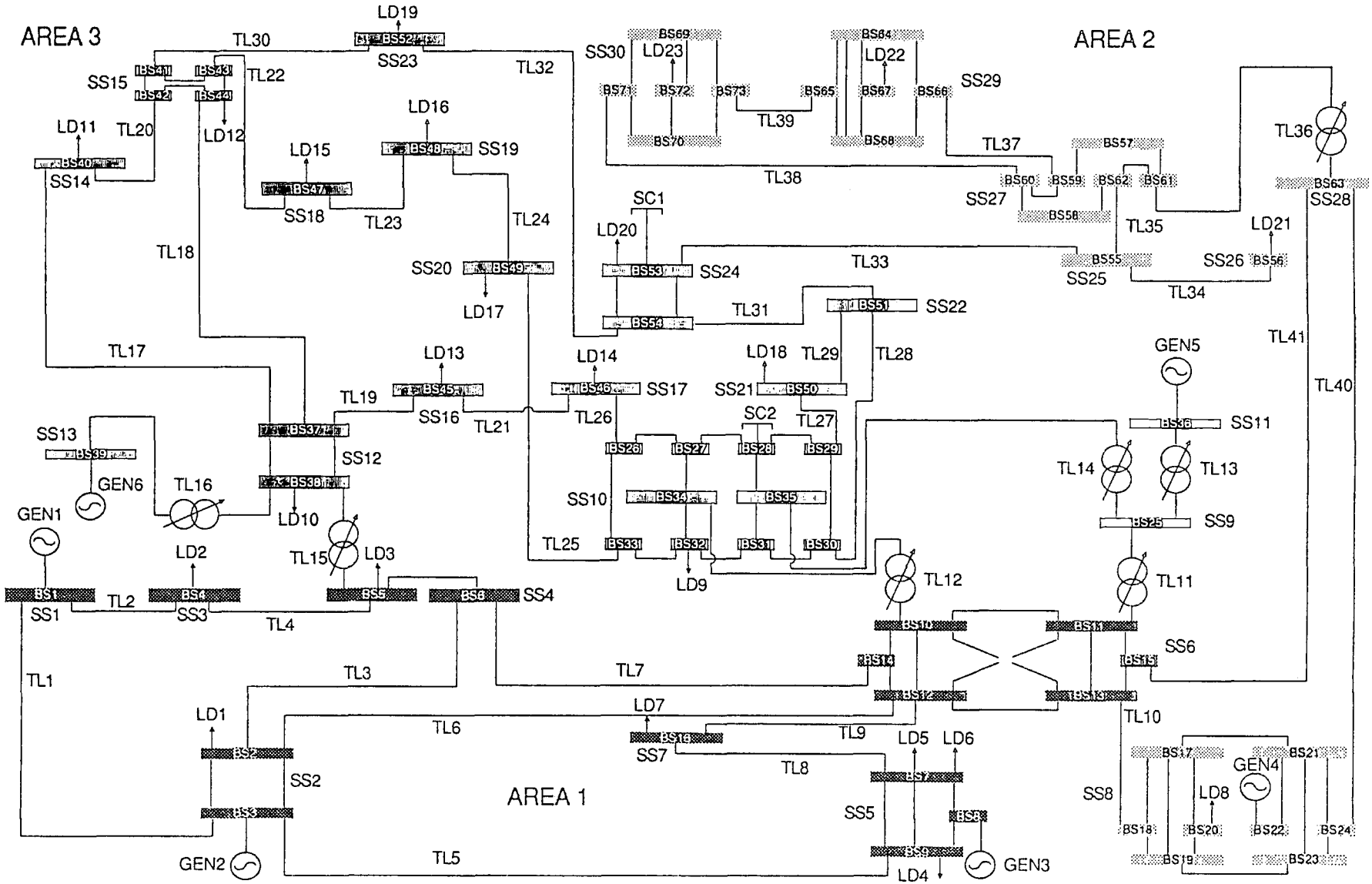


Fig. 8.1 30 node system decomposed into 3 areas

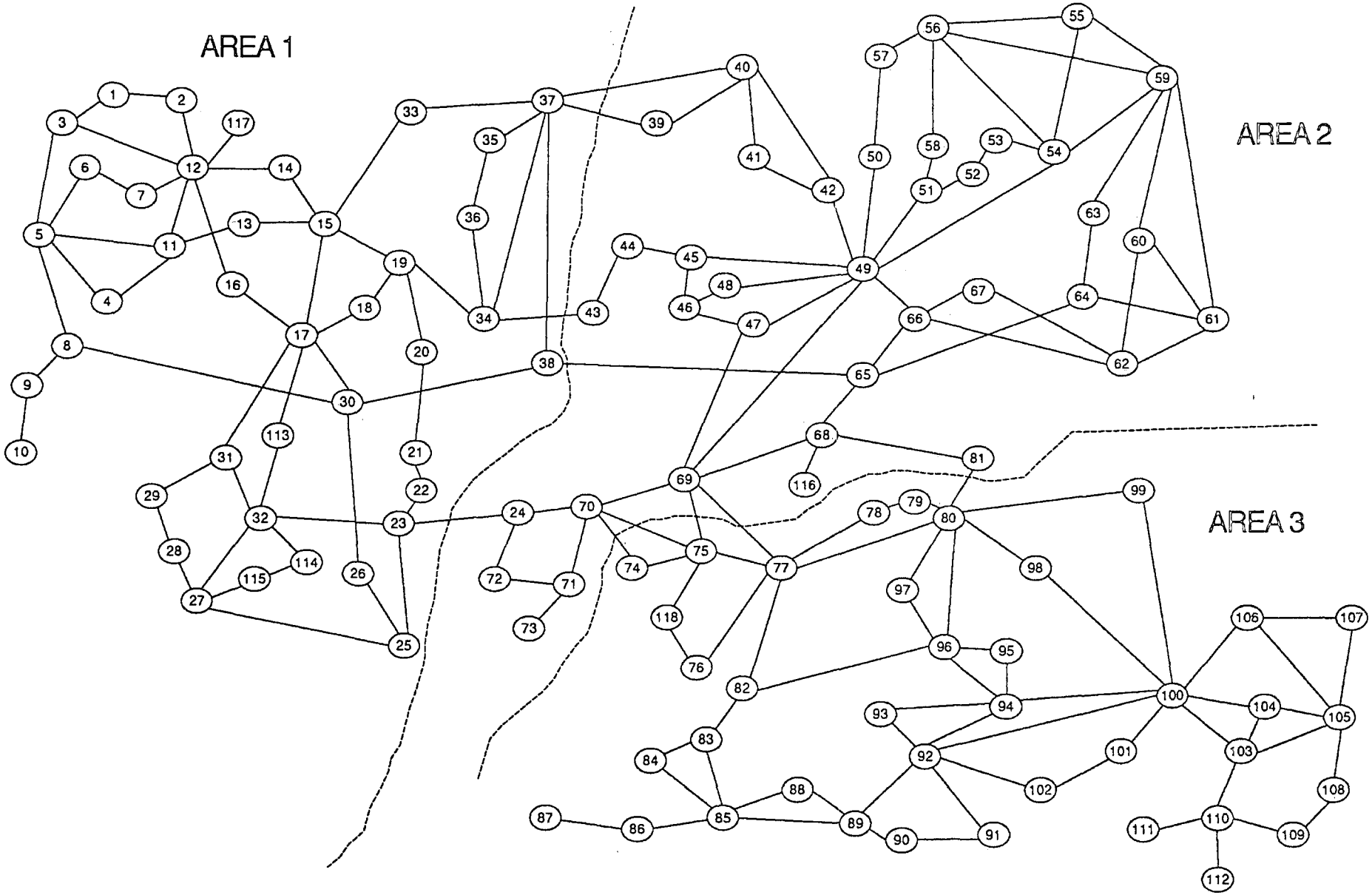


Fig. 8.2 118 node test system decomposed into 3 areas (first case)

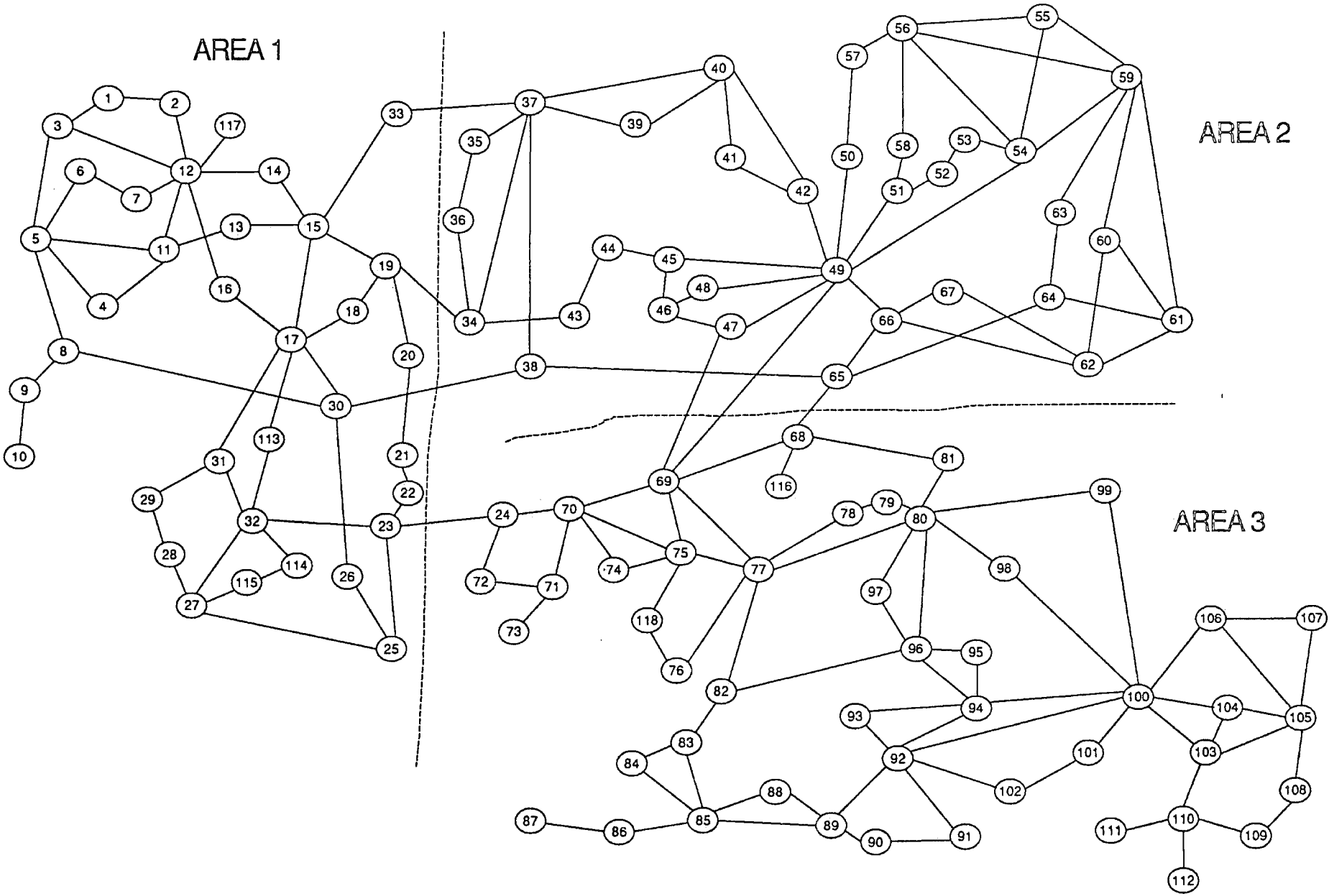


Fig. 8.3 118 node test system decomposed into 3 areas (second case)

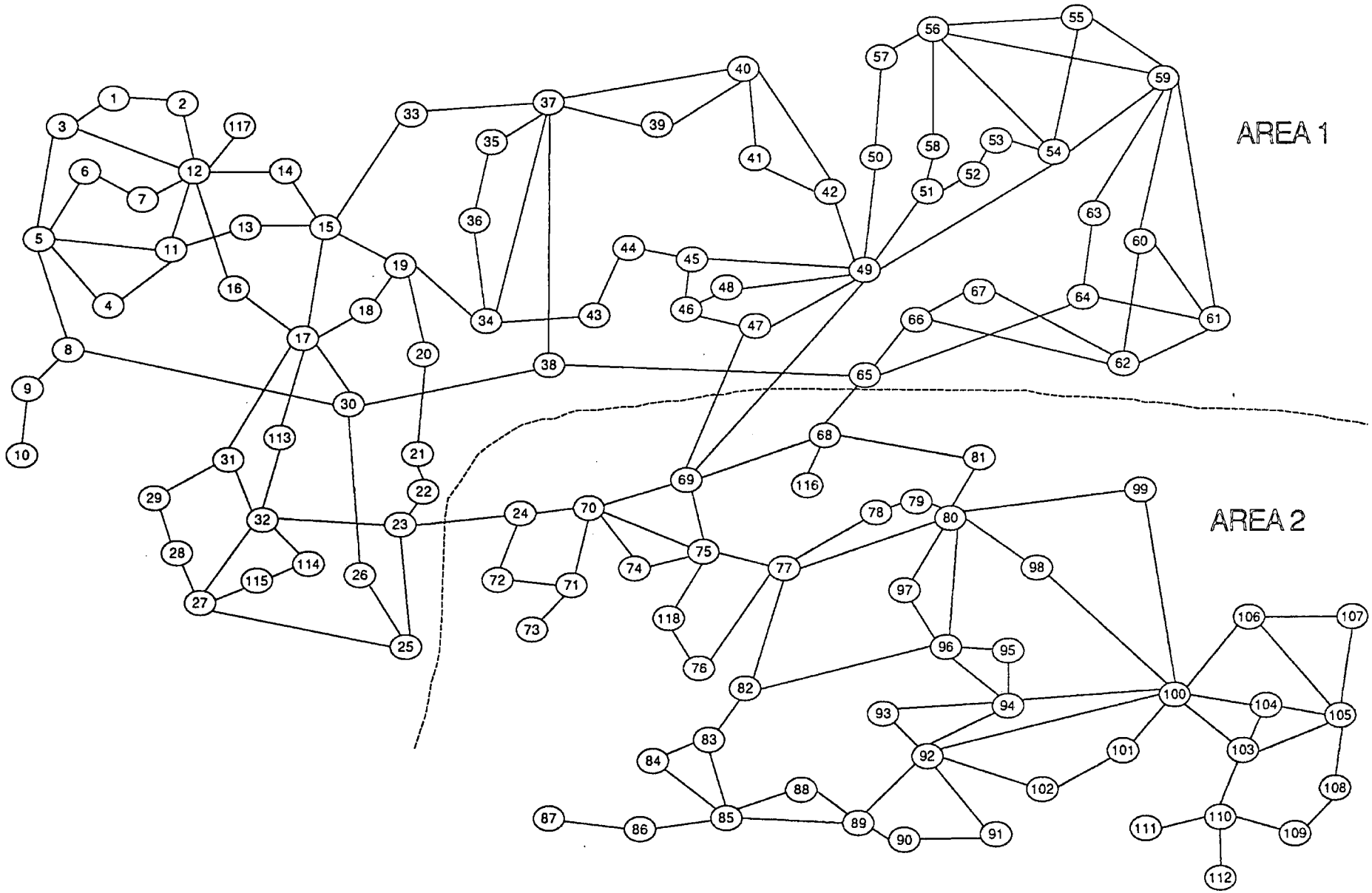


Fig. 8.4 118 node test system decomposed into 2 areas

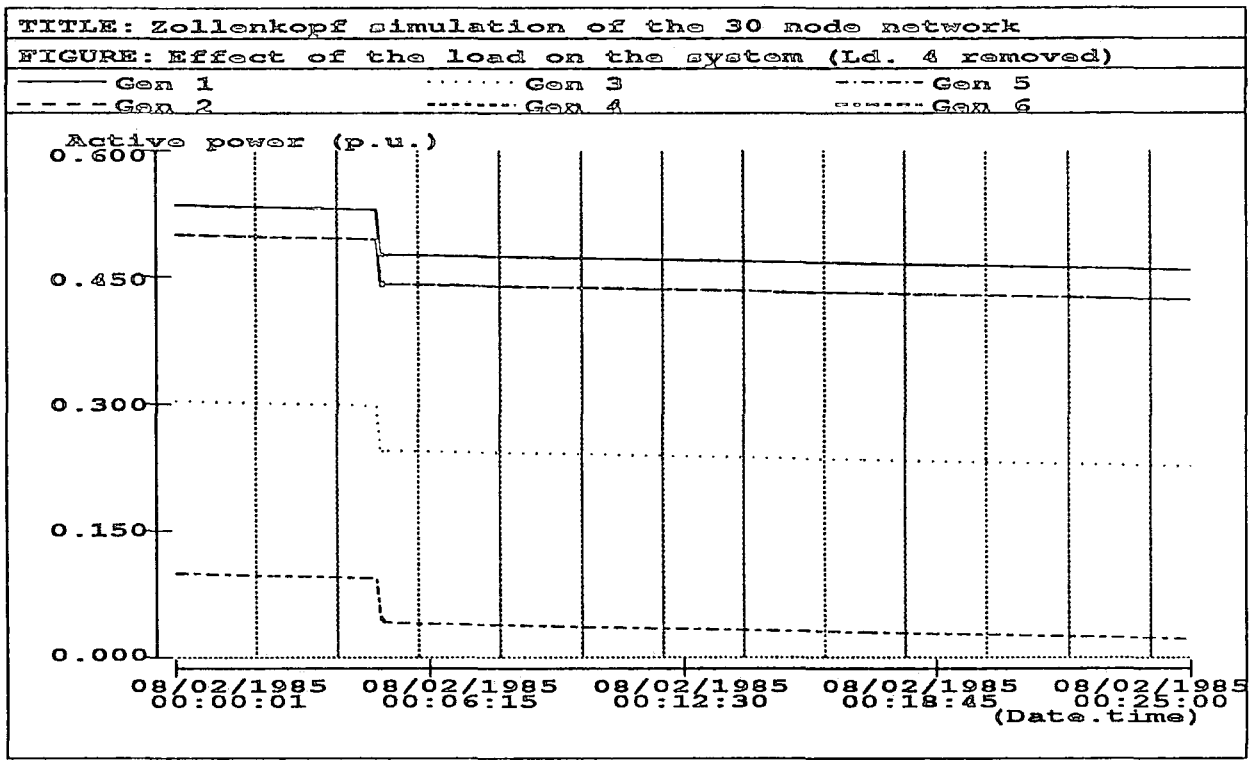


Figure 8.5

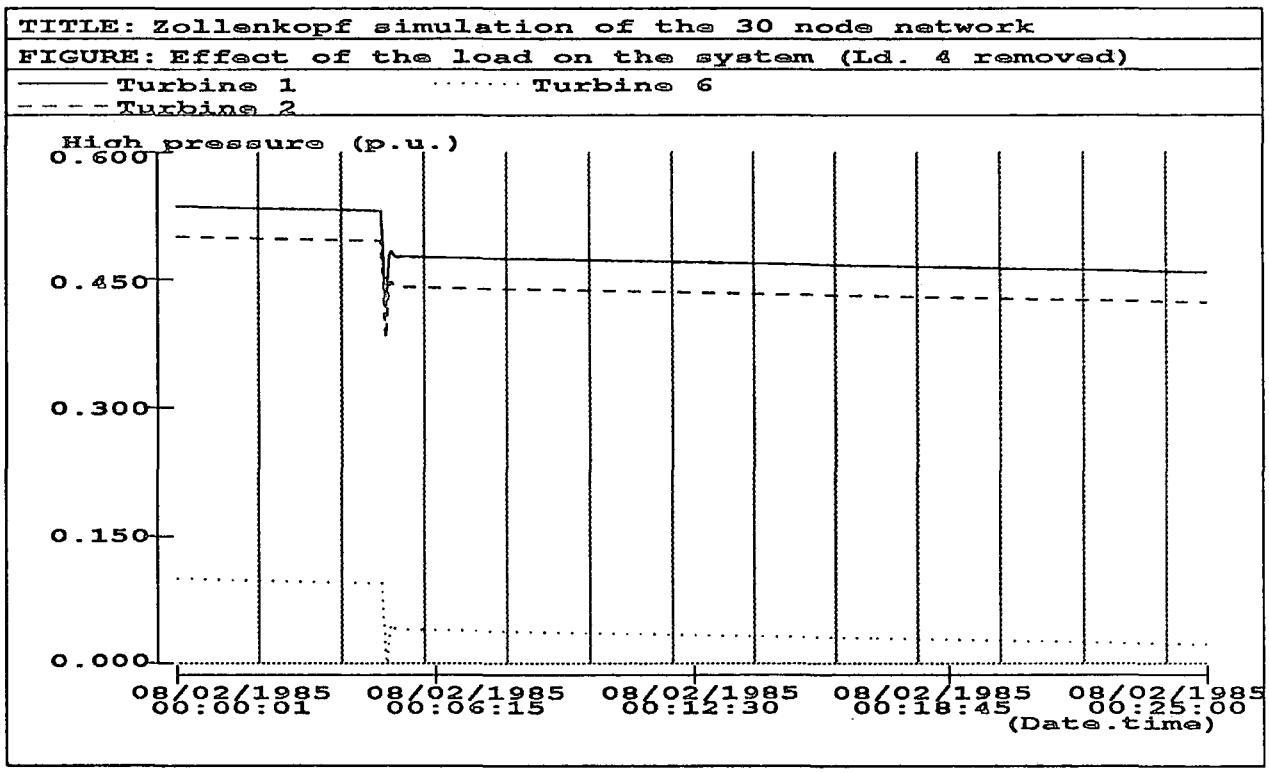


Figure 8.6

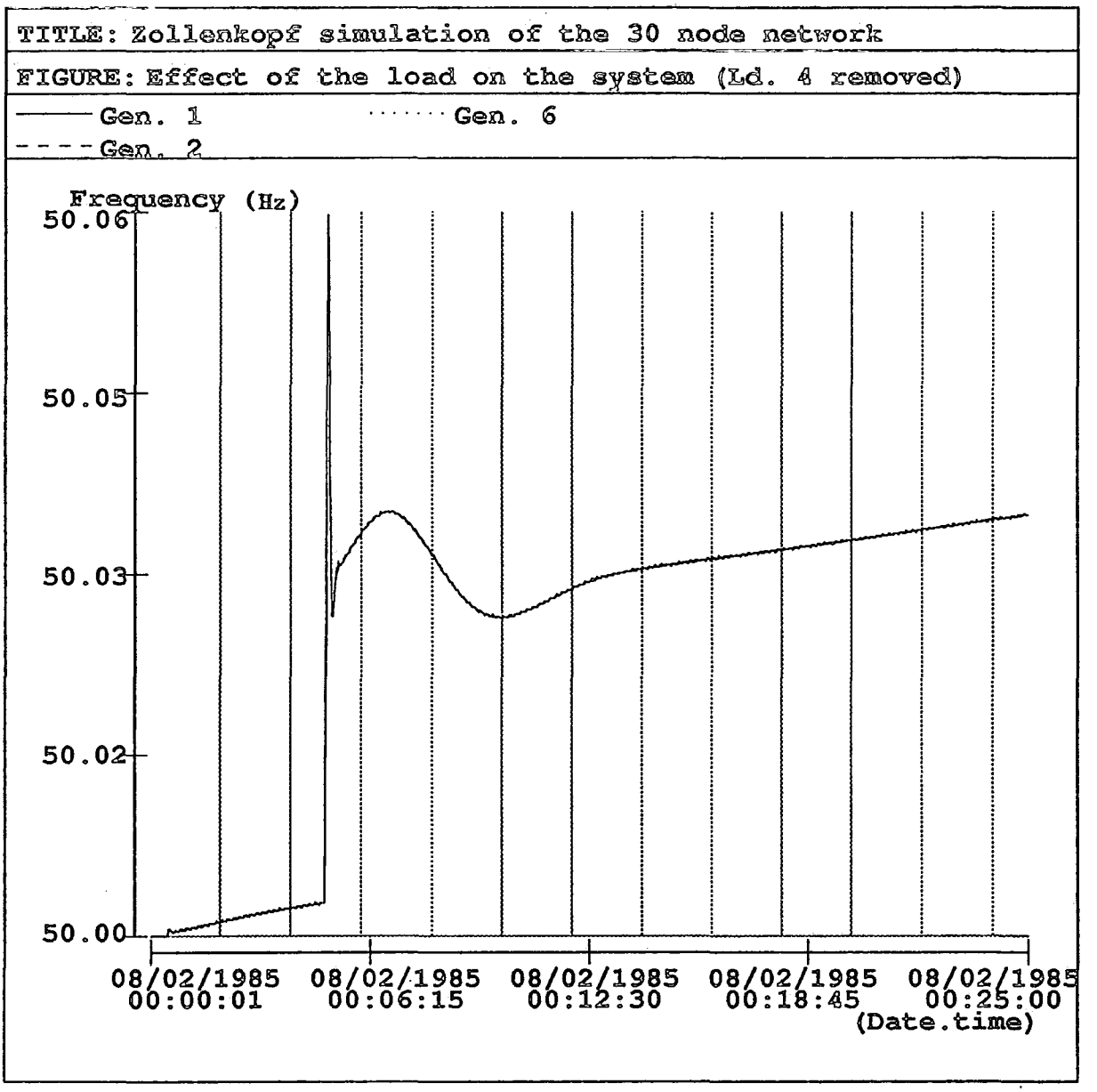

TITLE: Zollenkopf simulation of the 30 node network
FIGURE: Effect of the load on the system (Ld. 4 removed)
Gen. 1
Gen. 6
Gen. 2
Frequency (Hz)
50.06
50.05
50.03
50.02
50.00
08/02/1985 00:00:01
08/02/1985 00:06:15
08/02/1985 00:12:30
08/02/1985 00:18:45
08/02/1985 00:25:00
(Date.time)

Figure 8.7

leads to a reduction of the pressure and steam flow in the turbine to reduce the mechanical power to the desired value. The boiler controllers sense this variation together with the variation of the generator speed and power set point and act accordingly on the different boiler variables. In this case the amount of steam flow and pressure is decreased, and the drum pressure is initially increased. This is because the inlet valve position is adjusted before the boiler quantities, and all the spare steam flow, which is stopped by this valve, is stocked in the drum. Figures (8.6), (8.8), and (8.9) show the turbine high pressure, the boiler steam flow and drum pressure variations.

The frequency shows a sharp increase following the disconnection of load 4. The turbine high pressure shows an instantaneous response because its time constant is of about 0.3 seconds, and the adopted simulator time step (1 second) would not allow for such lags to appear on the graphs. The reduction of this pressure is followed by a reduction in the generator speed, therefore its frequency is decreased after the sudden overshoot. Following this reduction the high pressure in the turbine is increased to provide enough mechanical power for the slowing down generator. These oscillations are damped out after few minutes for most of the variables and the balance is restored within the system.

The active power (figure 8.5) is decreased by the same amount for all generators. This is because their governor gains are set to the same level. Therefore the variation of the load is initially equally shared between all the active units within the system. Since the simulator is running alone without any external control from the EMS not only the initial sudden load change is shared equally between the different units, but the continuous system load variation is also shared equally between the generators.

Compared to the generator and prime mover response, the boiler quantities are varying slowly following their long time constants.

8.2.1 Implementation of EMS with the simulator

If the simulator is controlled by Energy Management Software, the dispatch tends to distribute the variation of the load over the system economically

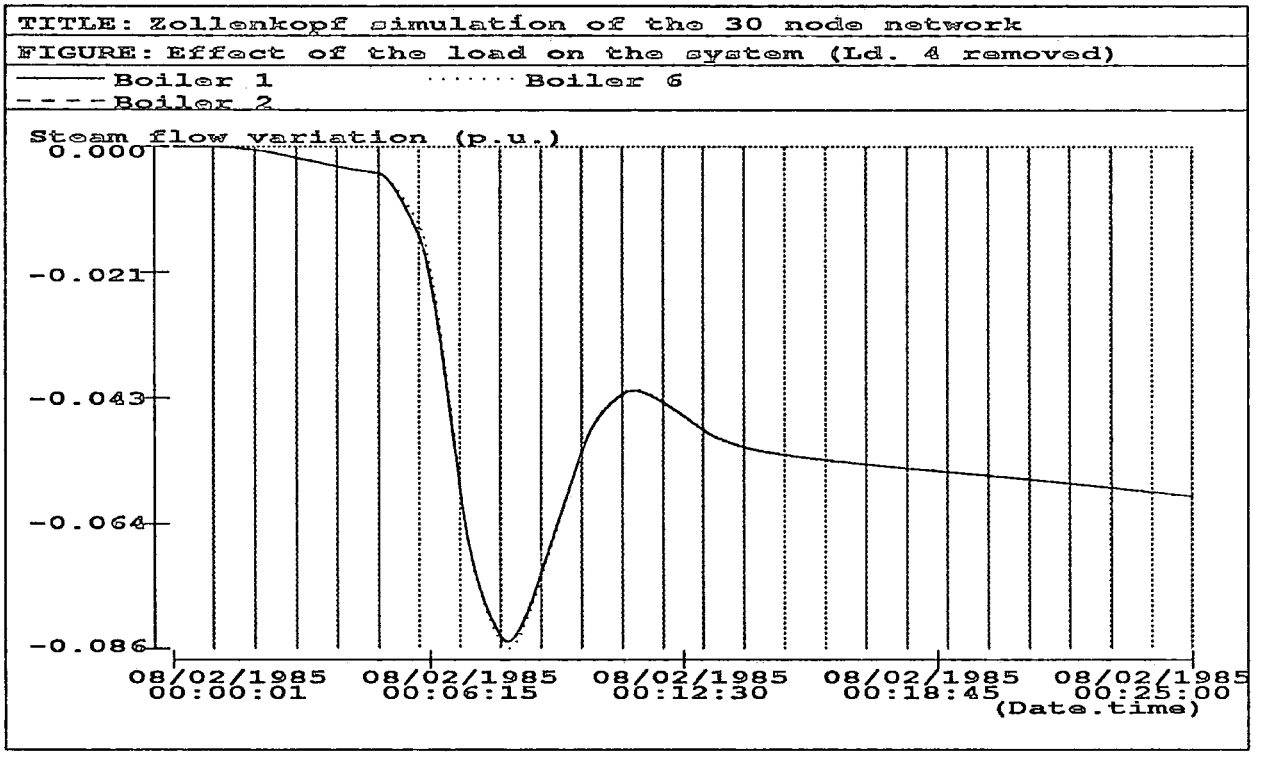


Figure 8.8

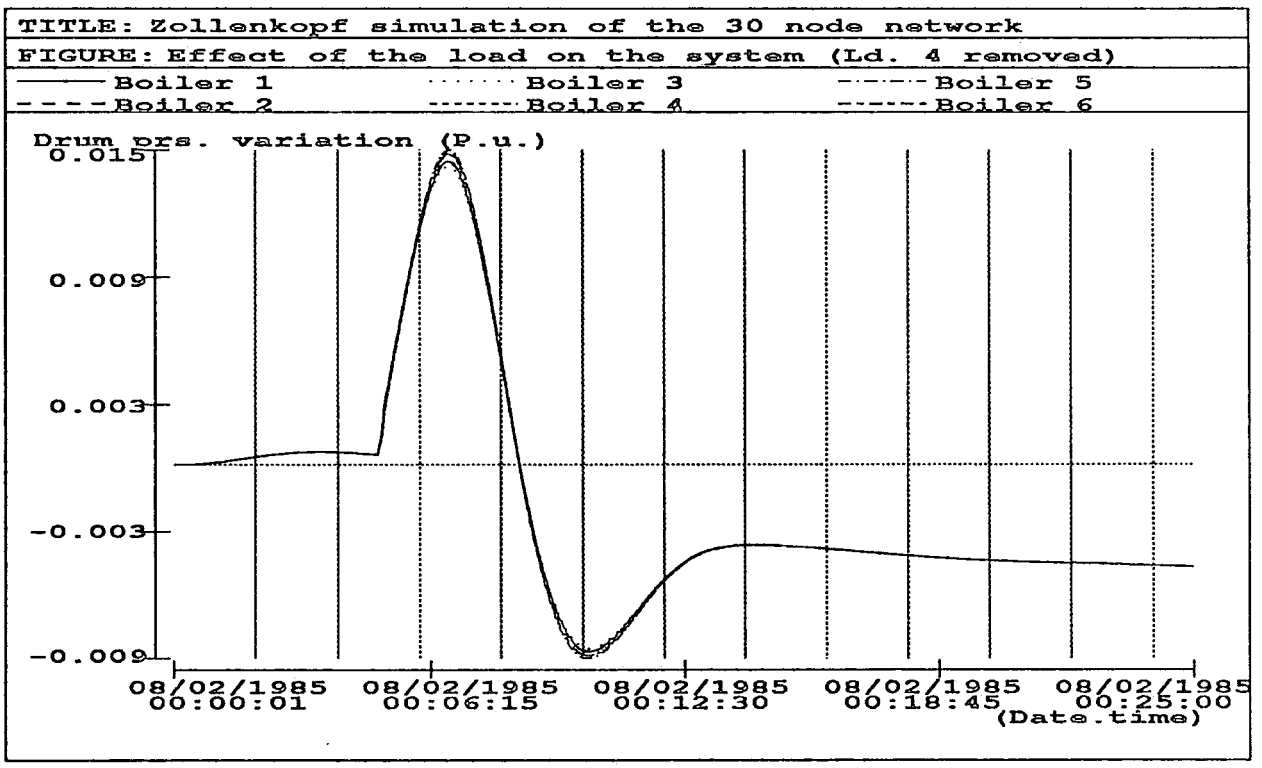


Figure 8.9

(figure 8.10). Generators 6 and 3 are operating close to their minimum outputs following the requirements of unit commitment to shut these units down. This is illustrated by the fact that the power set points for these two units are maintained constant at their minimum all through the period, prior to their actual disconnection (figure 8.12). This plot also shows that the other four plants are closely following the variation of their power set points. The decision to shut-down generators 3 and 6 is because the system load is light at this time of the day. The target time for this control action is set to 00:29:51 (hours:minutes:seconds) after midnight.

To illustrate in detail the ramping down of the generators to be disconnected and the ramping up of the remaining generators to support this power variation the segment of time around the shut-down target in figure (8.10) has been reproduced in an expanded form in figure (8.11). Before the closure of units 6 and 3 their output was quickly ramped down to the minimum. When the action of shutting down generators 6 and 3 took place the active generators ramped up quickly to produce the amount of power which had been supported by the disconnected plants.

The effect of this incident on the generating units is opposite to the load disconnection effect. Figure (8.10) shows an initial augmentation of the power of all the active generators to compensate for the disconnected generation. This has resulted in an increase of the turbine high pressure, a decrease of the frequency, and an increase of the boiler steam flow variation, as shown in figures (8.13), (8.14), and (8.16) respectively. The initial change in the turbine pressures and steam flow is supported by the boiler storage capacity. This explains the decrease in the drum pressure shown in figure (8.15).

There is also a difference between the locally controlled frequency (figure (8.7)) and that controlled by LFC (figure (8.14)). This is manifest in the difference between the steady state error and the oscillation of the locally controlled frequency after the first overshoot. The externally controlled frequency tends to settle down at 50 Hz whereas the locally controlled one continues to increase following the reduction of the system load. This is due to the

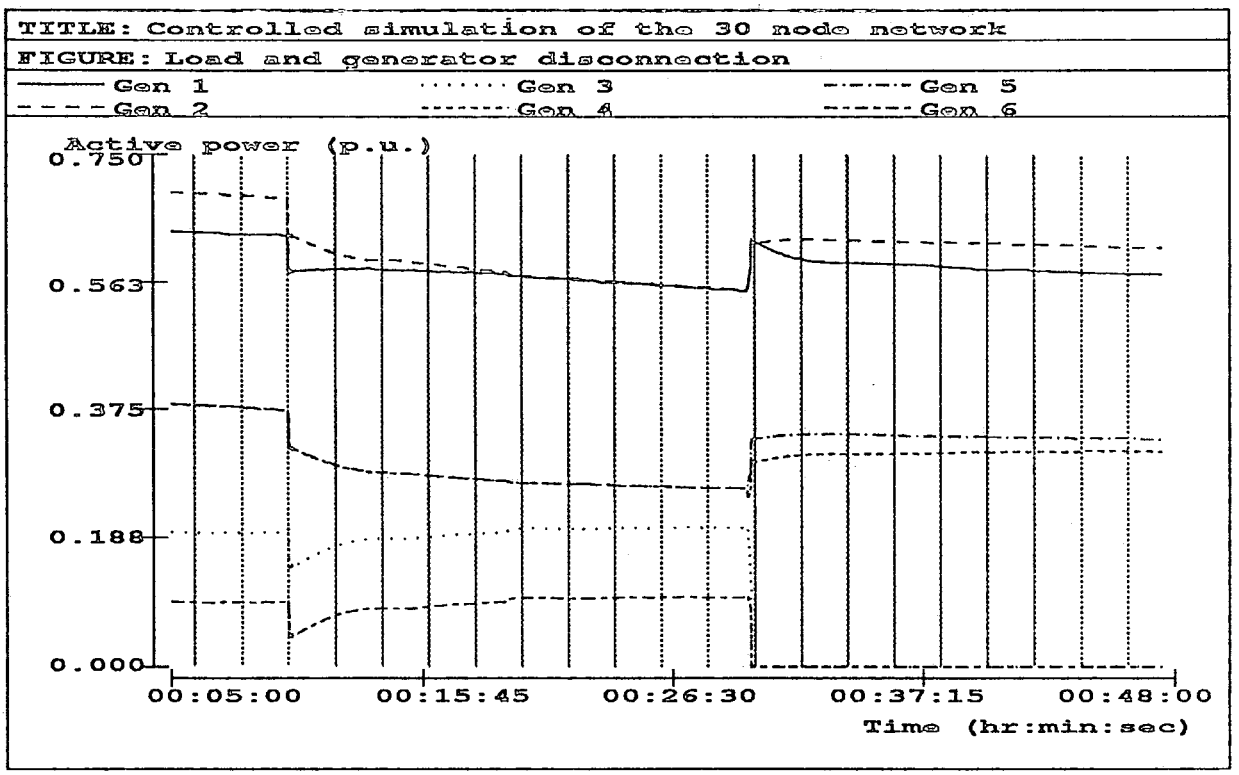


Figure 8.10

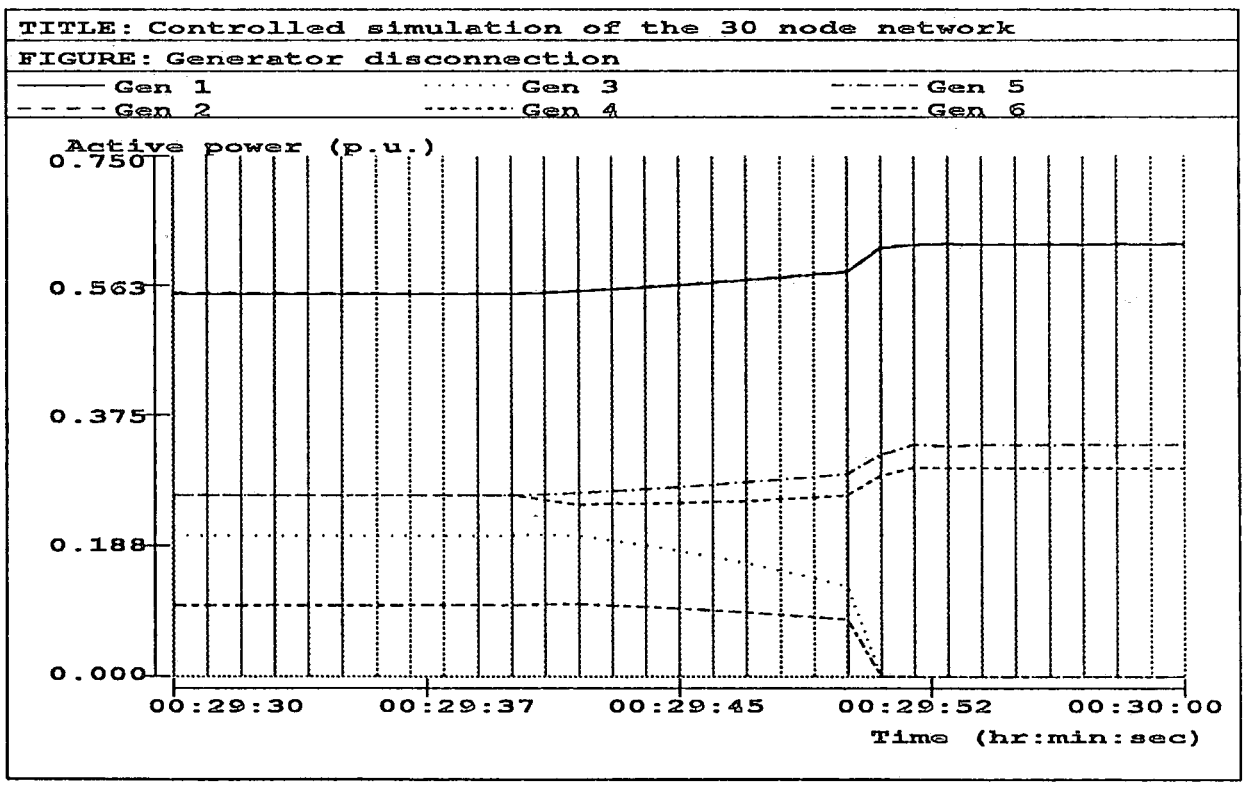


Figure 8.11

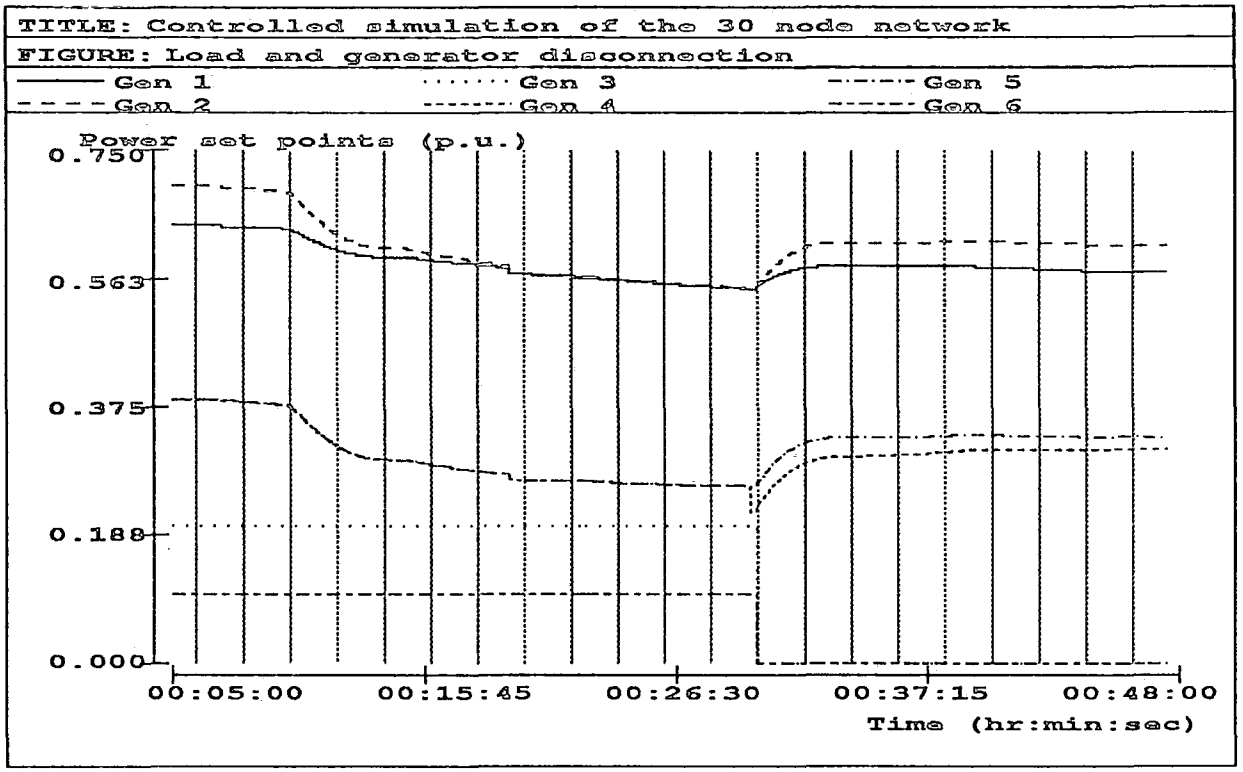


Figure 8.12

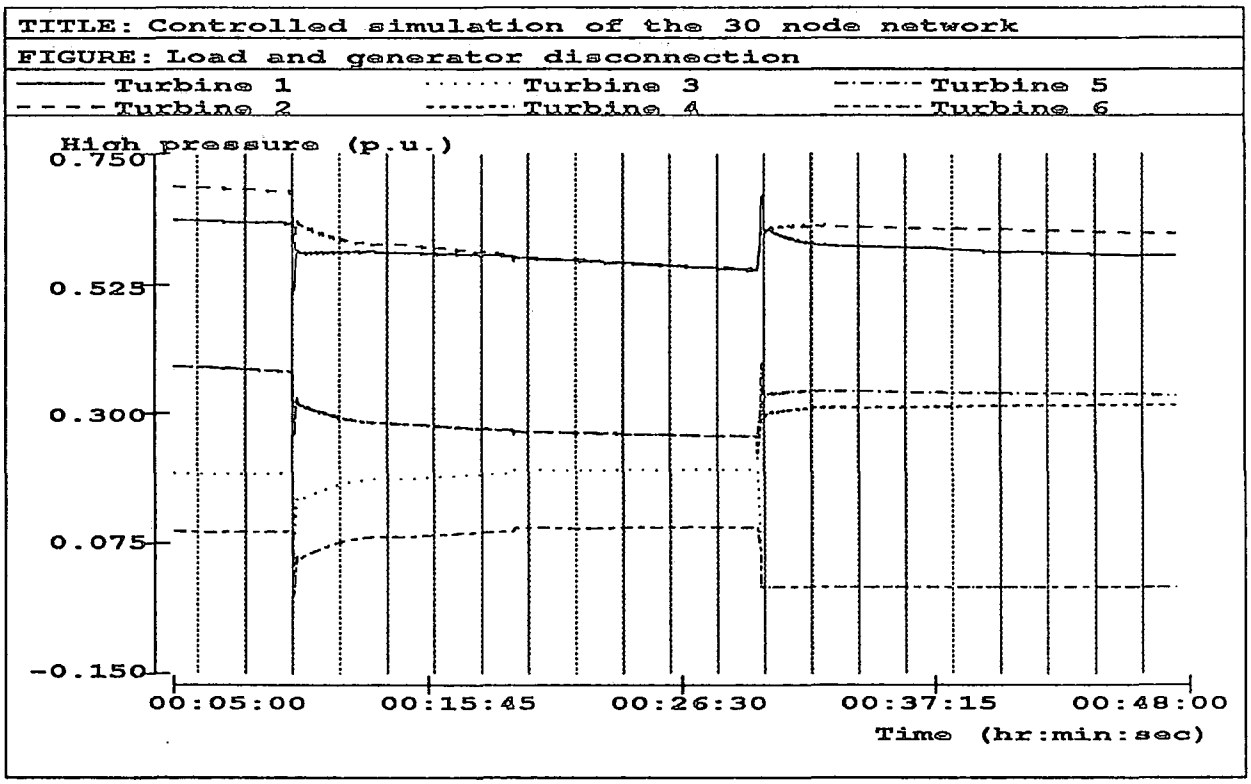


Figure 8.13

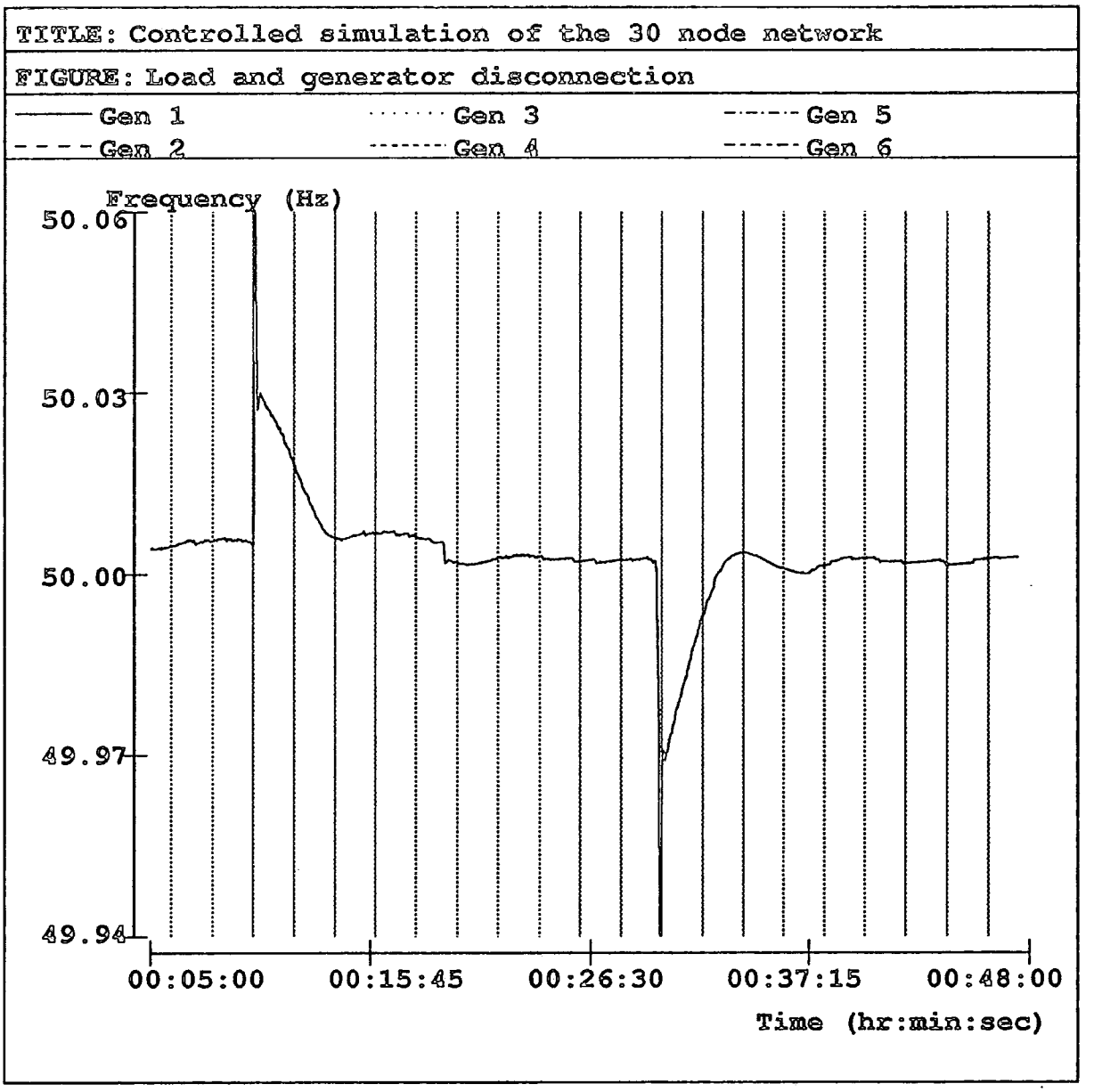

TITLE: Controlled simulation of the 30 node network
FIGURE: Load and generator disconnection
Gen 1
Gen 2
Gen 3
Gen 4
Gen 5
Gen 6
Frequency (Hz)
50.06
50.03
50.00
49.97
49.94
00:05:00
00:15:45
00:26:30
00:37:15
00:48:00
Time (hr:min:sec)

Figure 8.14

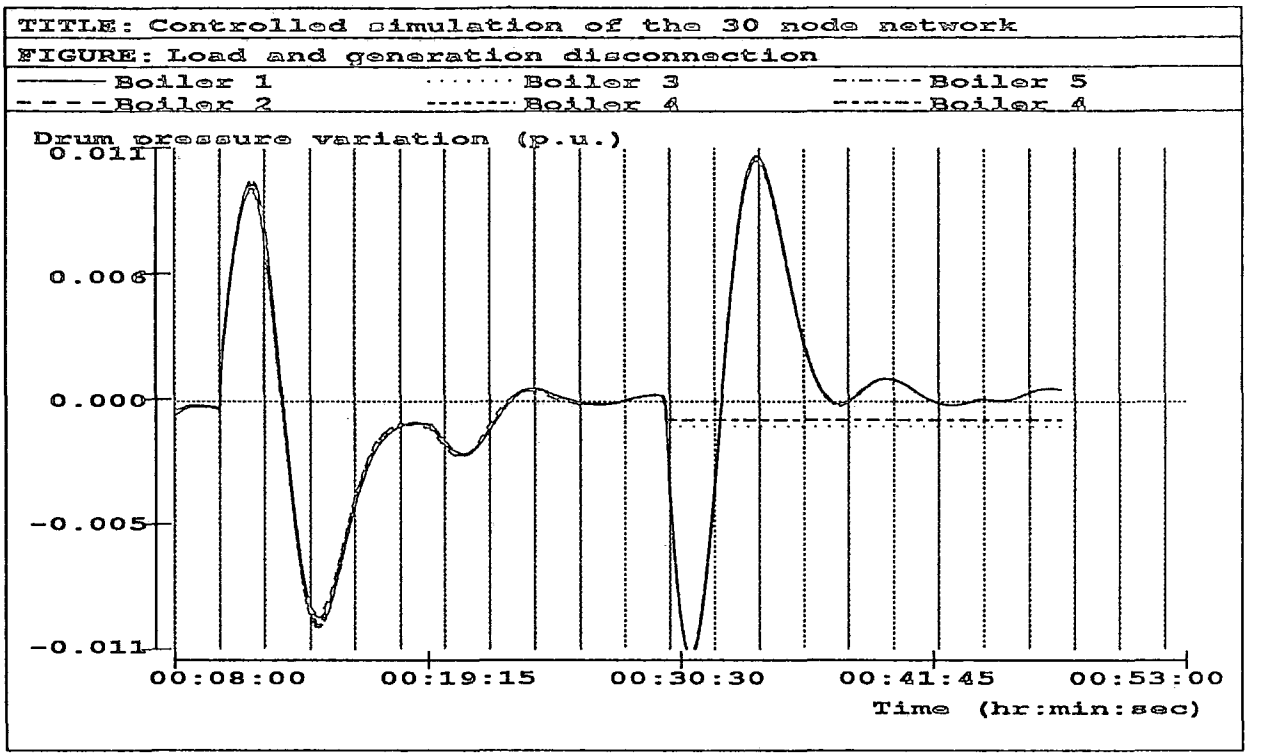


Figure 8.15

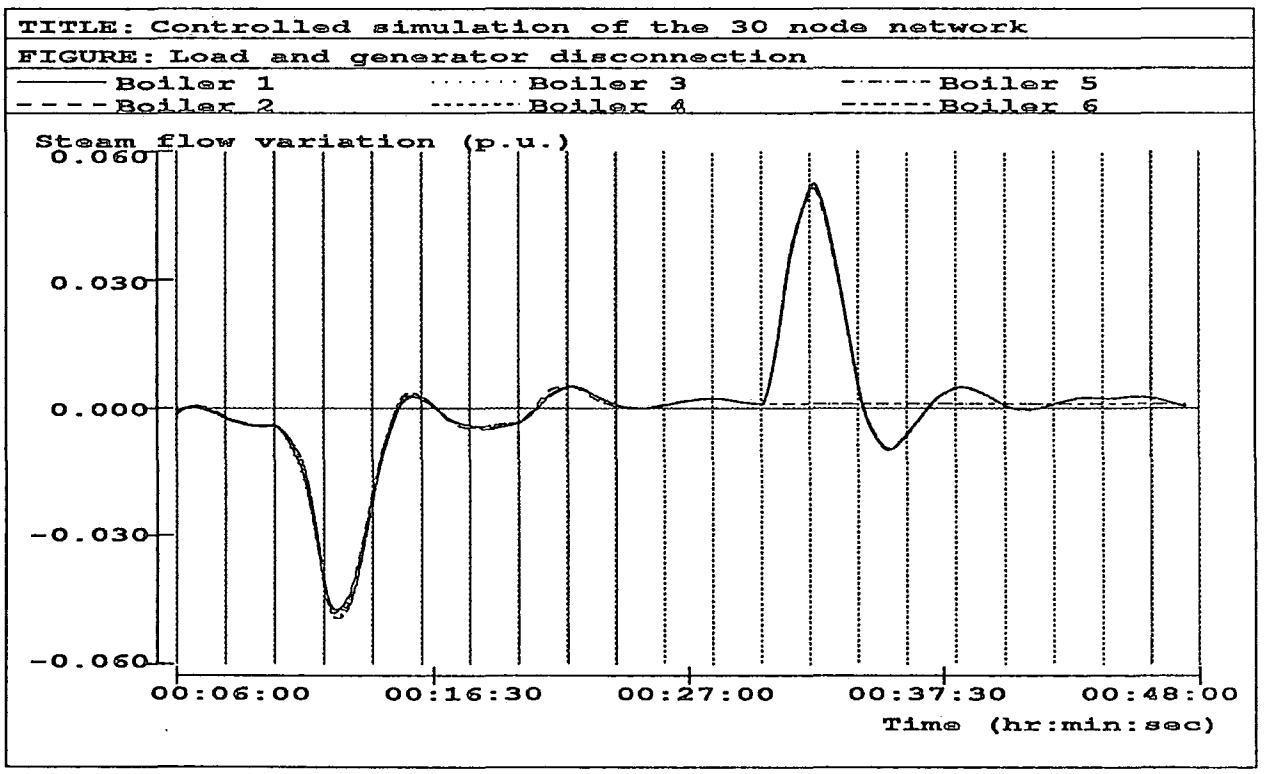


Figure 8.16

introduction of integral action in the load frequency control [7], which is not considered in the governor model. This also has an effect on the variation of the steam flow and drum pressure where the steady state errors are different.

Figures (8.15-8.16) show that some of the variables related to the disconnected generators such as the variation of the drum pressure, and steam flow are not set to zero, but kept constant at their latest value prior to disconnection. The generator frequency is also kept active following the variation of the system frequency. This simulates the bringing up of the generator frequency to the synchronising level to facilitate the re-synchronisation of the off-line plants in the case of a sudden loss of generation or augmentation of the load.

Figure (8.17) shows the active power of load 4 (the simulated incident). This load is consuming about 31 MW (about a seventh of the total load) as shown in the following event log. This also shows that the load is disconnected from the system at the required time (00:10:00), and the global load before and after removing generators 3 and 6 has continued varying normally. This load is fully supported by the remaining active units.

Time	Events
8/2/1985.00:00:10	total system load 237.847MW updated
	.
8/2/1985.00:09:50	total system load 231.376MW updated
8/2/1985.00:10:00	total system load 231.265MW updated
8/2/1985.00:10:00	* $open: load breaker 4
8/2/1985.00:10:10	total system load 200.24MW updated
8/2/1985.00:10:20	total system load 200.143MW updated
	.
8/2/1985.00:29:40	total system load 188.932MW updated
8/2/1985.00:29:50	total system load 188.835MW updated
8/2/1985.00:29:51	gen.3 is shut-down as requested by unit comm. at 00:29:51
8/2/1985.00:29:51	gen.6 is shut-down as requested by unit comm. at 00:29:51
8/2/1985.00:30:00	total system load 188.738MW updated

Table 8.1 First test events and load variation

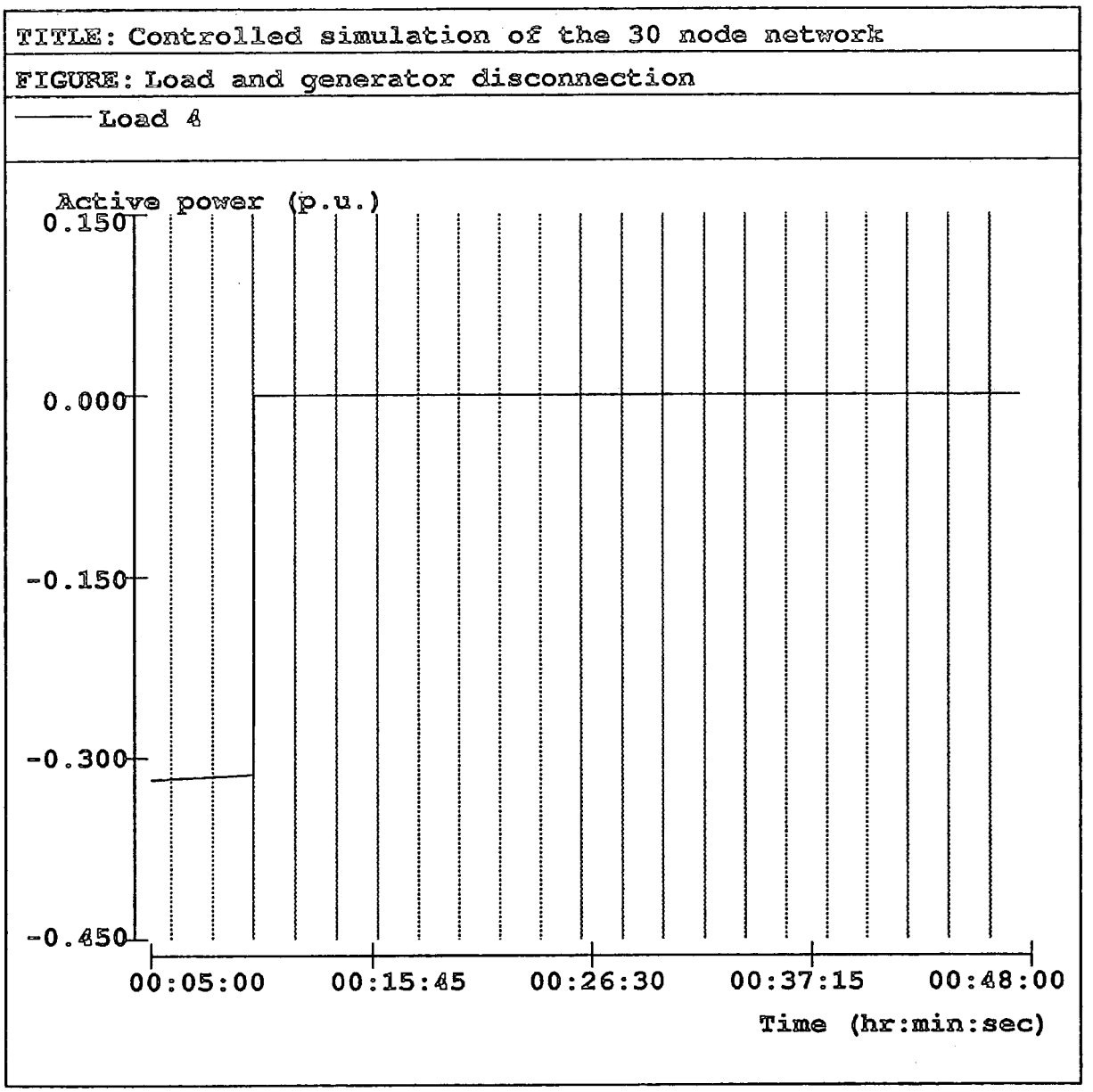

TITLE: Controlled simulation of the 30 node network
FIGURE: Load and generator disconnection
Load 4
Active power (p.u.)
0.150
0.000
-0.150
-0.300
-0.450
00:05:00
00:15:45
00:26:30
00:37:15
00:48:00
Time (hr:min:sec)

Figure 8.17

8.3 ISLANDING AND SYNCHRONISATION

This test has been carried out to illustrate the robustness and efficiency of the simulator in handling severe events such as islanding. This simulator is capable of preserving the stability of the system over the test period which lasted about 45 minutes. The scenario implemented consists of the following sequence of events.

```
$time 7:25:0
      $open sending end of line 32
      $open sending end of line 24
      $open sending end of line 19
*
$wait 0:1:30
      $open sending end of line 15 ;* network split into two islands
*
$wait 0:4:0
      $open sending end of line 10
      $open sending end of line 41
*
$wait 0:1:30
      $open sending end of line 33 ;* network split into three islands
*
*restore the split system
*
$wait 0:4:0
      $sync sending end of line 32 ;* re-synchronisation of island 1 and 2
*
$wait 0:1:30
      $close line 24,19, and 15
*
$wait 0:5:0
      $sync sending end of line 33 ;* re-synchronisation of remaining islands
*
```

```
$wait 0:1:30
      $close line 41 and 10
*
$wait 0:2:0
      $close load 19 ;* disconnected by protection
*
$wait 0:2:30
      $close load 16 ;* disconnected by protection
*
$wait 0:1:30
      $close load 23 ;* disconnected by protection
*
$exit
```

The starting time for this test is selected as 7:25:00 since at this time of the morning the system is heavily loaded. An integration time step of a quarter of a second is necessary to ensure the stability of the Newton-Raphson algorithm. Although this is normally necessary only when a large variation of state variables occurs. This usually happens under very severe events such as the first time steps following the islanding of a heavily loaded system.

The scenario started with an unexpected split of the system into two islands at 7:26:30. The resulting subsystems were unevenly loaded. Most of the generation was located in the first island whereas island 2 includes only one small generator (50 MW) (generator 6). Five and a half minutes later island 1 was split into two islands. The layout of these independent islands can be deduced from figure (8.1). The loading conditions of the two newly formed islands were similar to the ones produced by the first incident. Consequently, the frequency of subsystems 2 and 3 has decreased sharply (figure 8.18) leading to load shedding. Load 19 was disconnected by the under-frequency protection 3 seconds after the first islanding. Three seconds later load 16 has also been tripped since the system was still overloaded. The removal of load 23 was sufficient to restore the balance in island 3. The behaviour of these loads is shown in figure (8.20). The disconnection of the load has caused a transient

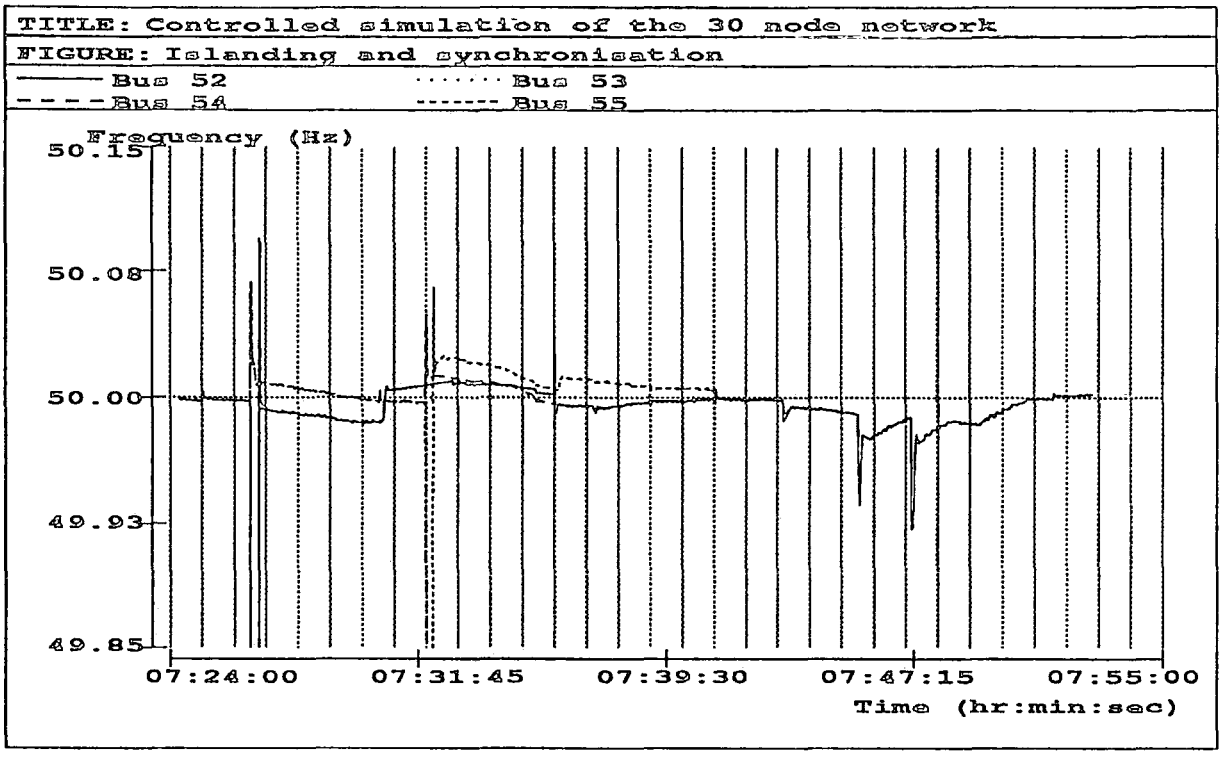


Figure 8.18

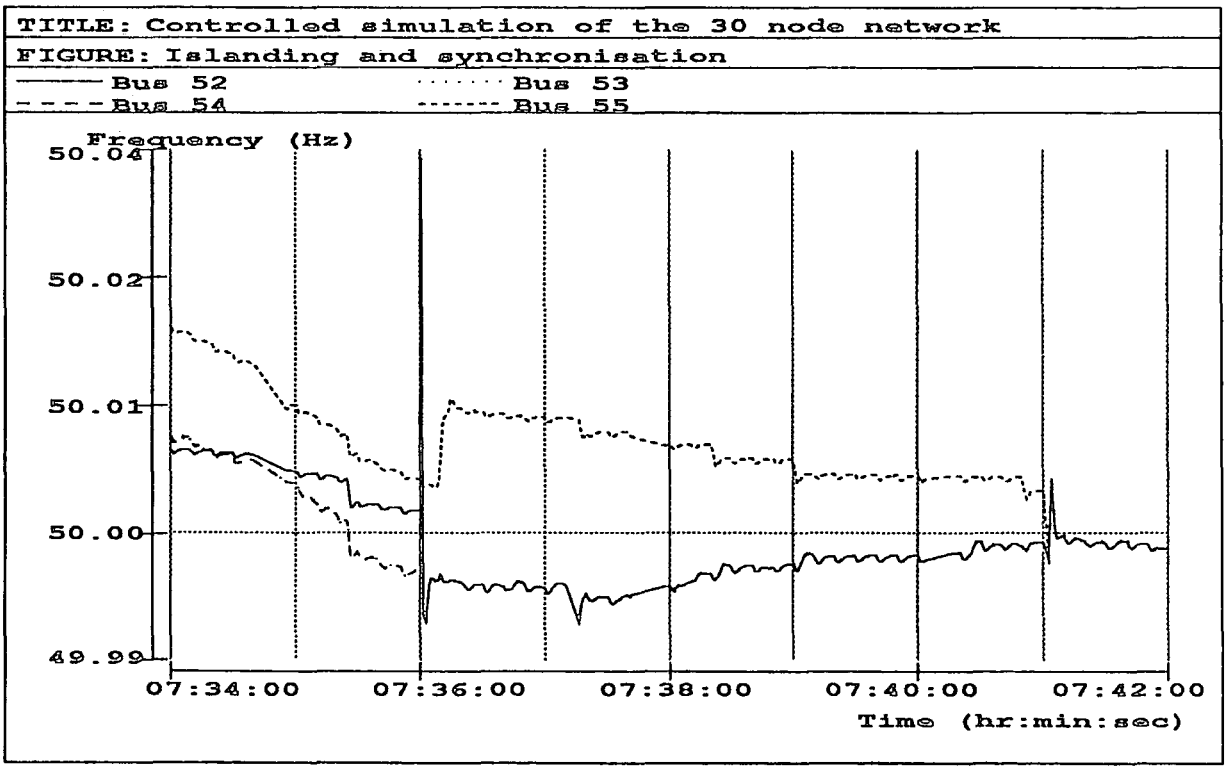


Figure 8.19

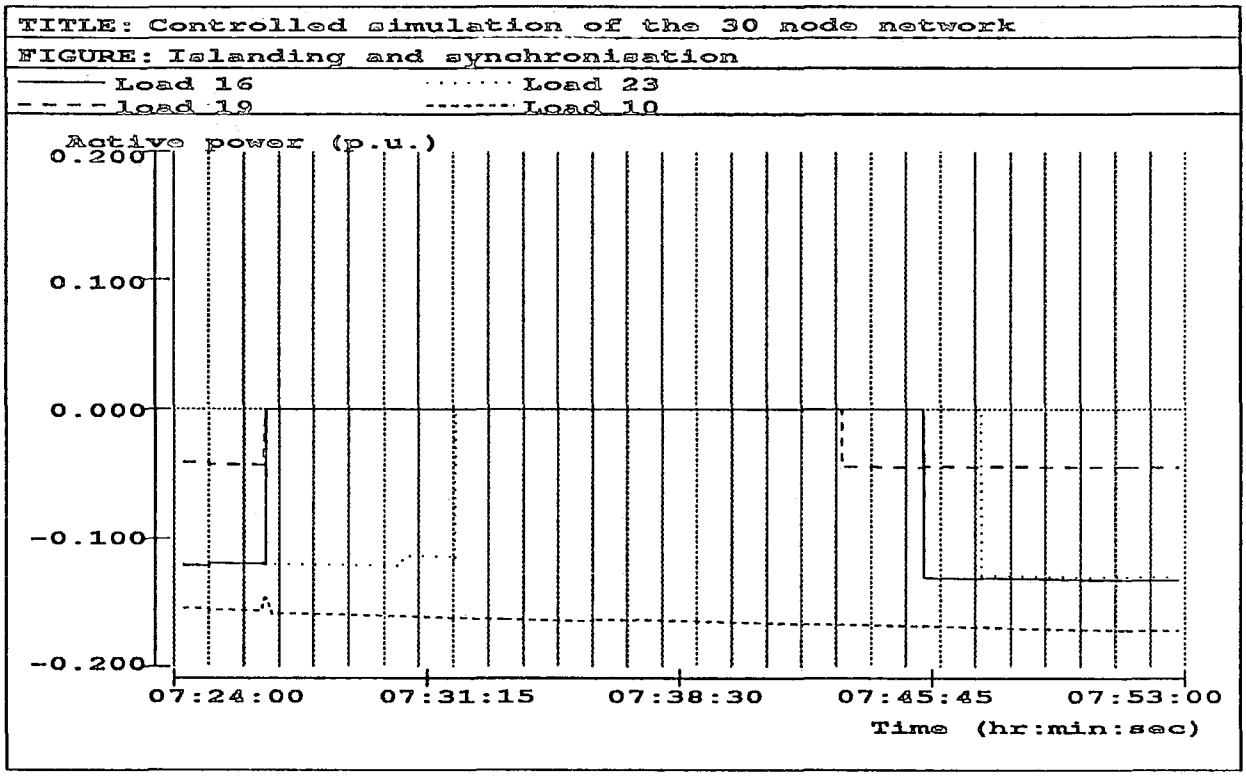


Figure 8.20

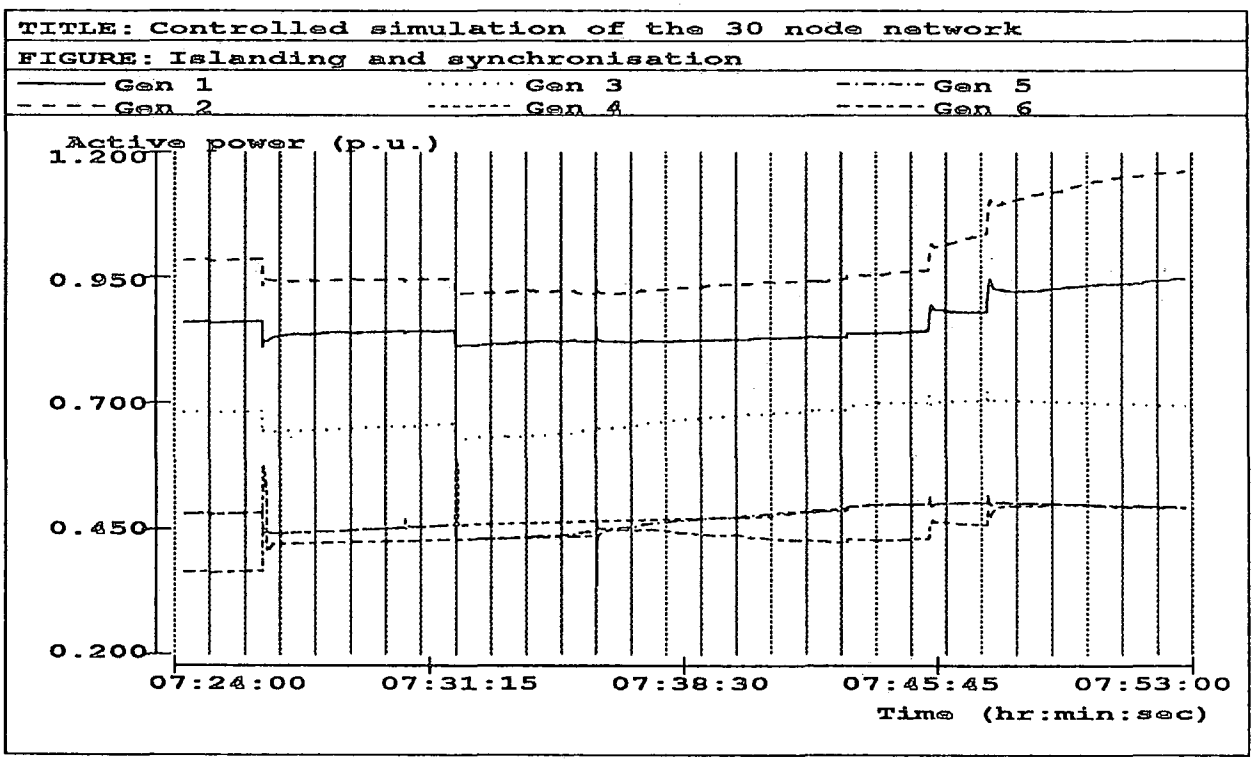


Figure 8.21

increase of the system frequency which has settled down quickly. The first island was under-loaded. To preserve stability in this case the generator outputs (generators 1,2,3,5) were decreased (figure 8.21). This has led to an increase of the island frequency (figure 8.18).

As the three islands were stabilised, island 1 and 2 have been successfully re-synchronised. Five minutes later the remaining islands were also re-connected and the normal state of the network has been restored. Since the system has completely recovered from this severe disturbance the loads which were removed from the system by under-frequency protection tripping were re-energised again without any problem.

8.3.1 Variation of the generation

Figure (8.21) shows the behaviour of the various generator active powers. The full capacity of generators 3 and 4 was exploited. Generator 6 was operating close to its maximum limit. This explains why this isolated generator could support only a small amount of the load increase caused by the system split. Therefore, to establish the balance within this island some load has been shed. Generator 4 behaved similarly after being isolated within an independent island. The three independent sub-networks continued operating normally until island 1 and 3 have been re-synchronised and re-connected at 7:36:00. This was followed by a slight increase in the active power of generators 3, 5, and 6. The re-synchronisation of the remaining islands was also carried out without any serious secondary effect. The output of generators 3, 4, and 5 were slowly ramped up towards their maximum as dictated by the economic dispatch. This was balanced by a decrease of the output of generator 6. The connection of load 19 at 7:43:00 was supported by all the units following governor action. When load 16 and 23 were due to be re-connected generators 3, 4 and 5 were already at their maximum capacity. In this case the added load was supported by generators 1, 2, and 6 only. The latter has thus reached its maximum leaving generators 1 and 2 as the only regulating units in the system.

8.3.2 Variation of reactive power and voltage

To interpret the bus voltage behaviour, reactive power plots, the power

flow through the lines and the changes of the total load are needed. This is because all these quantities are related.

Figures (8.22) and (8.23) show the behaviour of the reactive power of generators 1,2,3 and generators 4,5,6 respectively. Figures (8.24-8.28) show the voltage magnitude at busbars 5, 38, 48, 49 and 52, 54, 53, 55, the load-flow through the disconnected lines listed in the scenario file, and the total active load and generation respectively.

Contrary to the distribution of active power among the generators, the size of the different units as regards reactive power is decreasing in the descending order of the generator number i.e. generator 6 is the biggest reactive contributor in the system and unit 1 is the smallest.

Opening lines 19, 24, and 32 has resulted in a deficit of reactive power in island 1 and a surplus in the second island. This is shown by the increase of this quantity in the first island (generators 1,2,3,4,and 5) and its decrease in the second subsystem (generator 6). The lightly loaded island has seen a rise of its voltage (busbars 52 and 38) except for bus 48. The voltage at busbar 48 has decreased because it is remote from all the generators and the power flow supplying its loads has found an alternative route with larger losses. The heavily loaded subsystem has shown a decrease of the voltage at busbars 54, 53, 55, and 49. Voltages at buses 54 and 53 are equal since these are located at the same substation. This voltage drop was the most significant among the represented voltages because the node where busbars 53 and 54 are located is remote from all generators and is linked to the disconnected line. As a result the AVR acted on the generators of island 1 to increase their output. The drop in the voltages has also resulted in a drop of the load in this sub-network (figure 8.20). The reduction of these loads has resulted in a reduction of total load followed by a decrease in total generation as shown in figure (8.28).

The load shedding following the complete split of the system has led to a decrease of the system load and generation. This was followed by the reduction in the reactive power of generator 6. Reactive generation in island

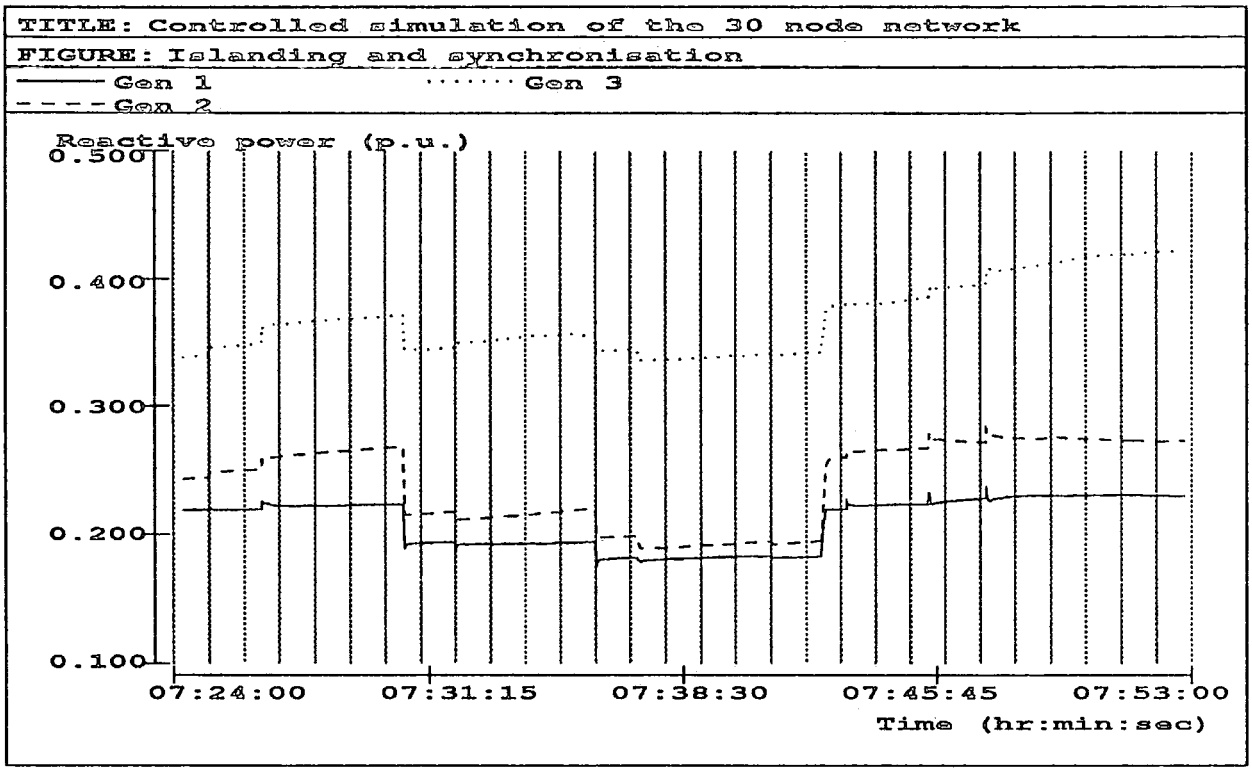


Figure 8.22

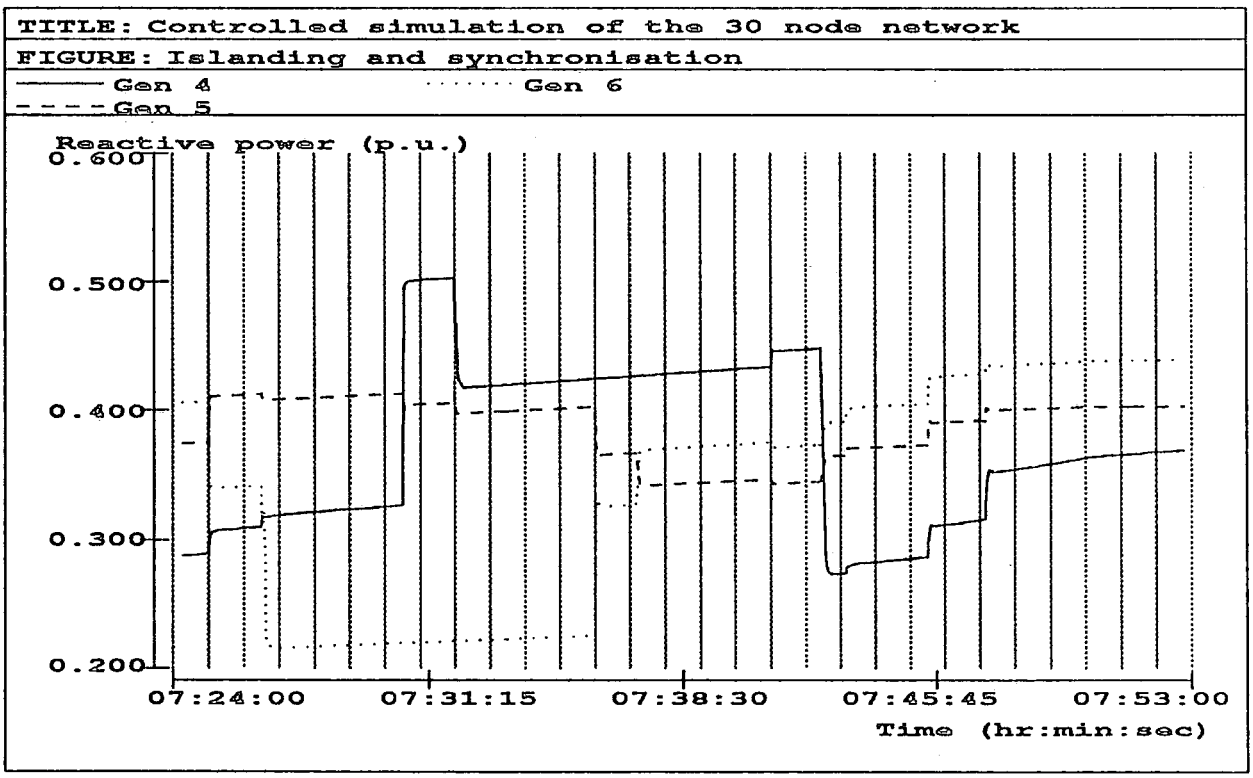


Figure 8.23

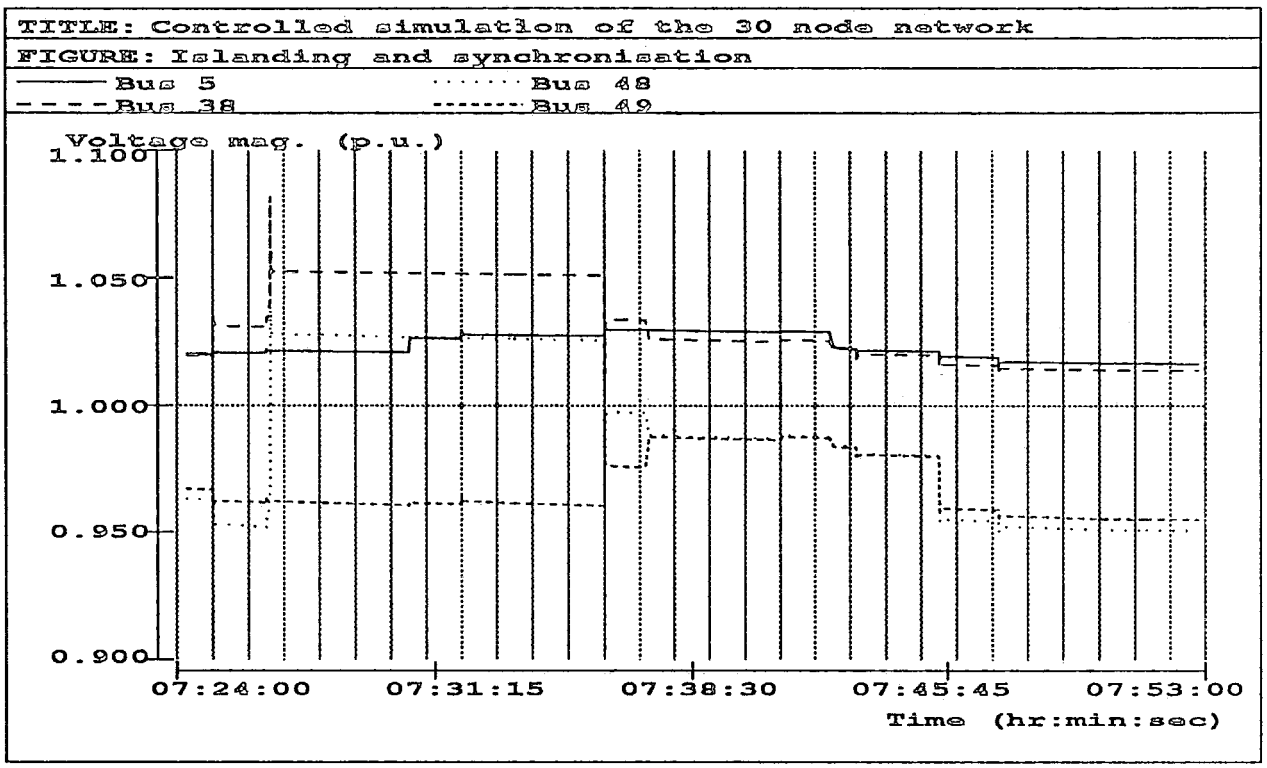


Figure 8. 24

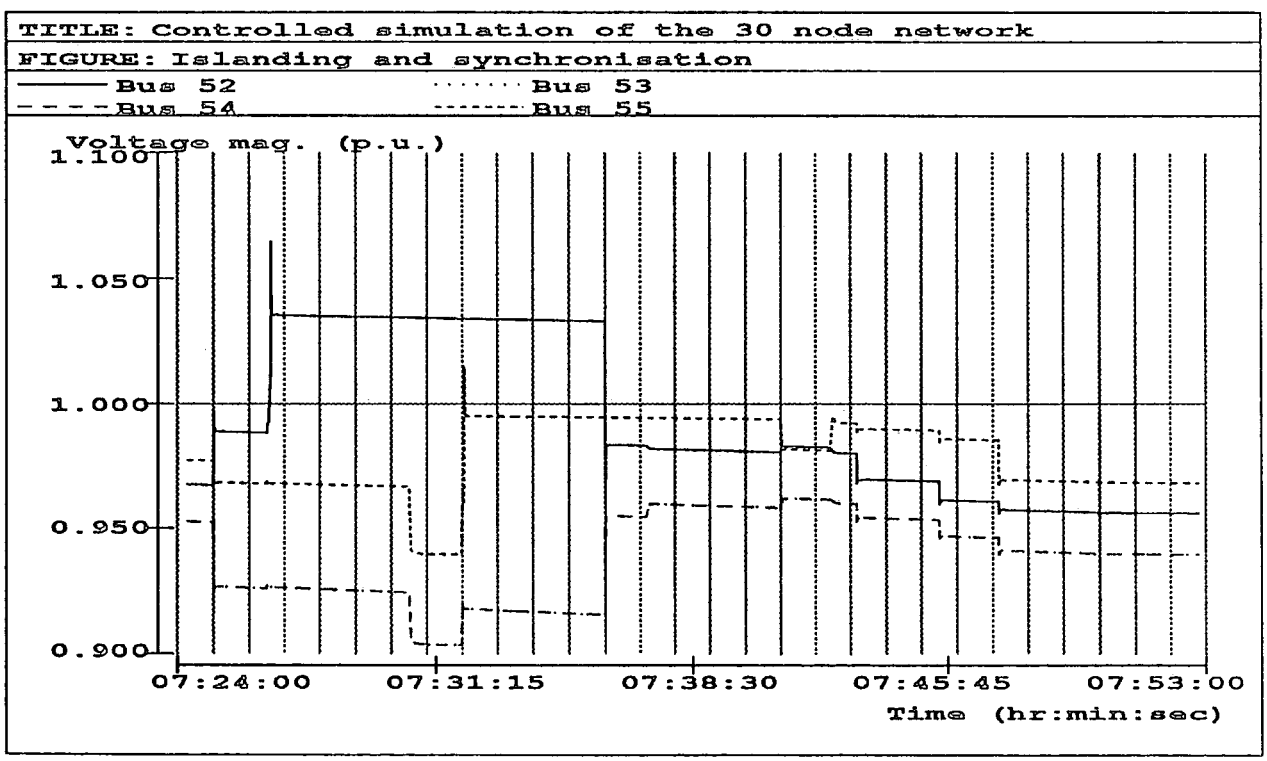


Figure 8.25

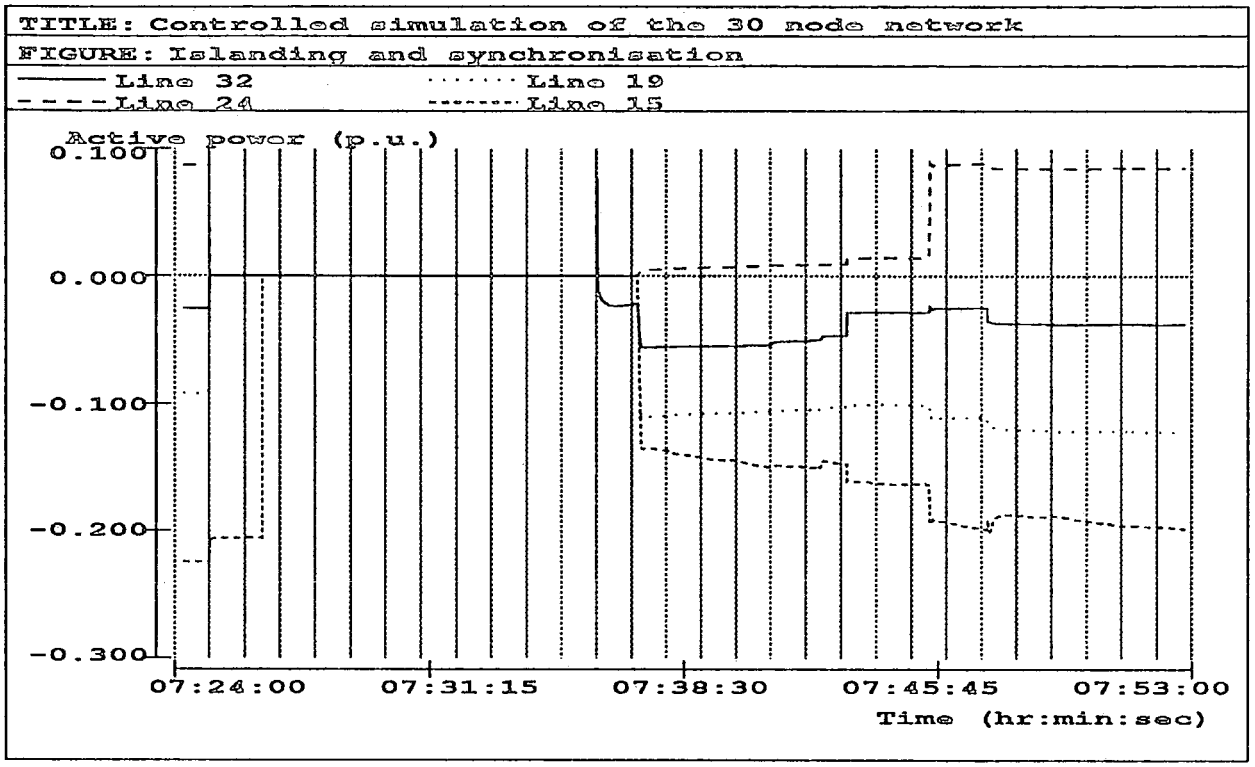


Figure 8.26

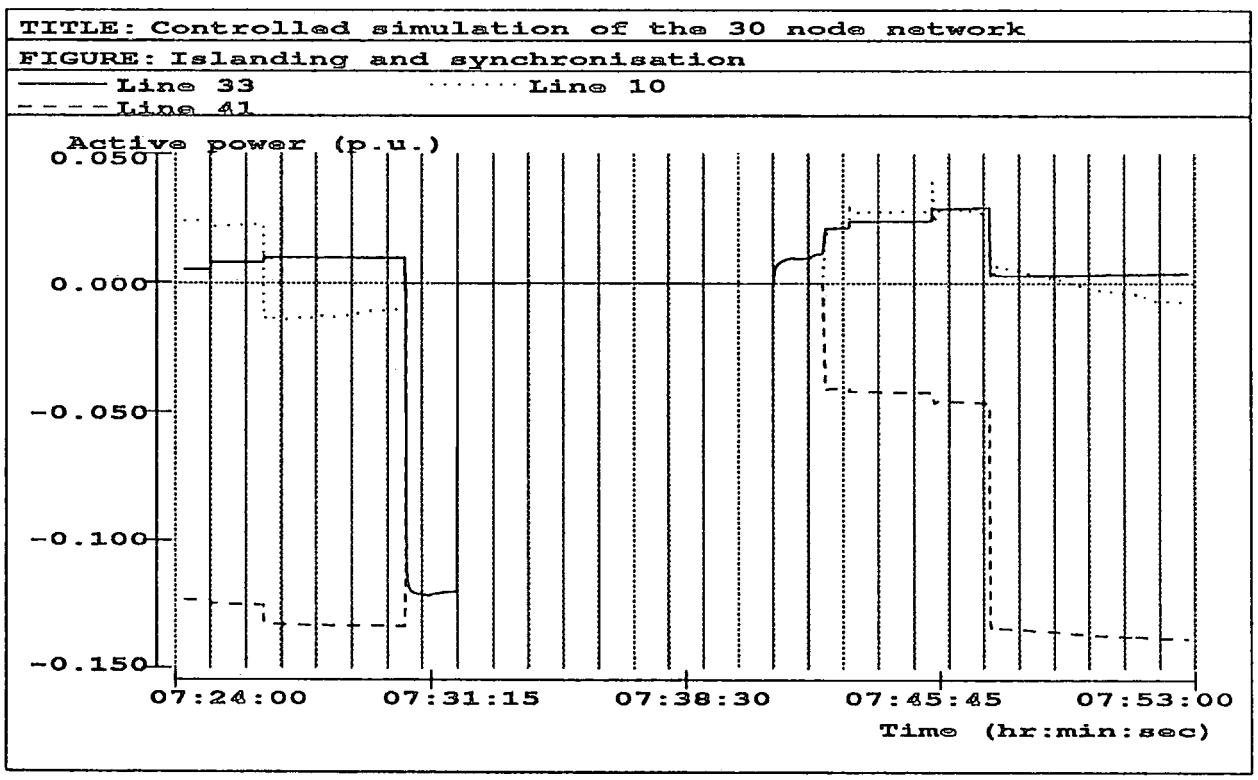


Figure 8.27

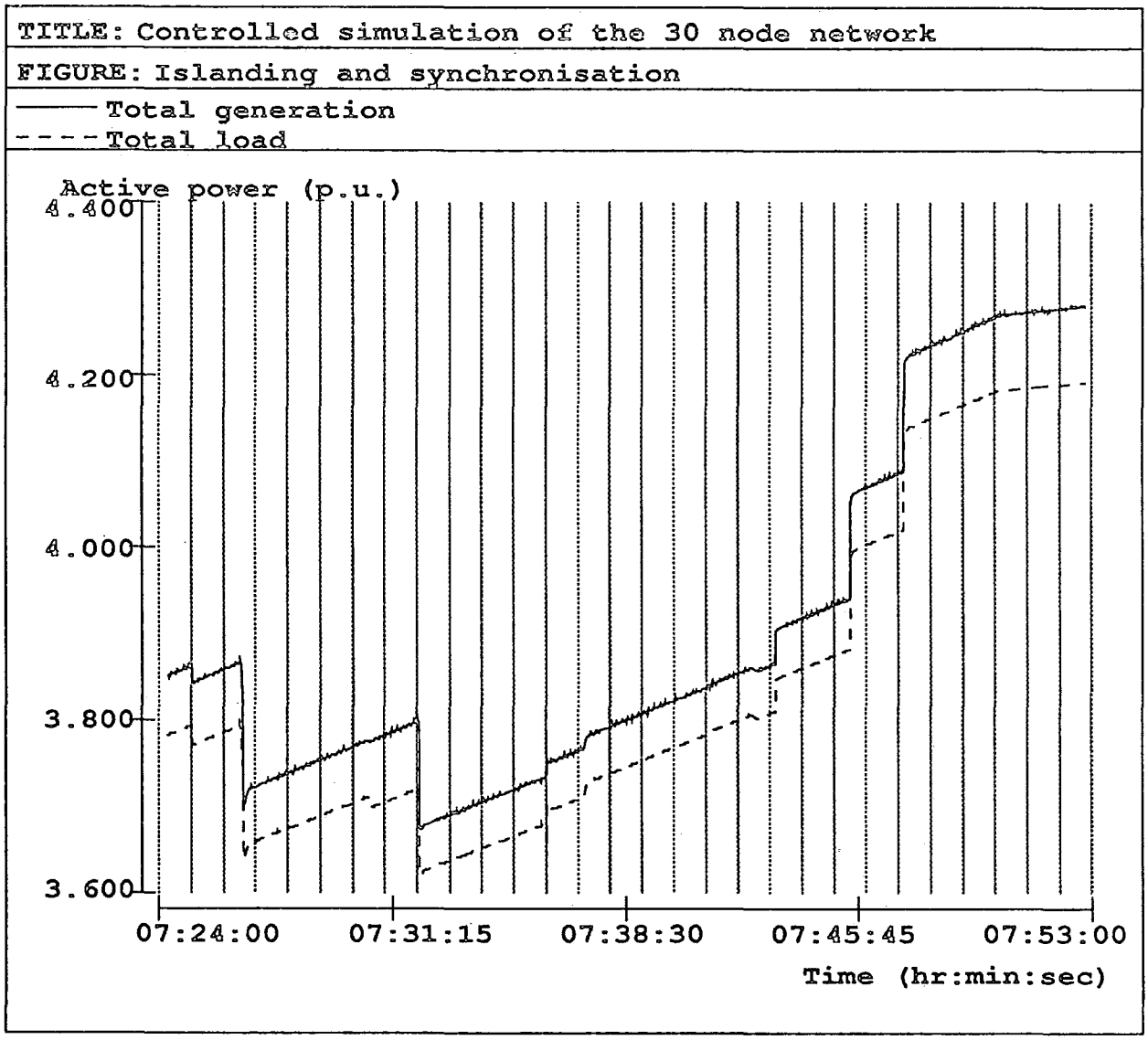

TITLE: Controlled simulation of the 30 node network
FIGURE: Islanding and synchronisation
Total generation
Total load
Active power (p.u.)
4.400
4.200
4.000
3.800
3.600
07:24:00
07:31:15
07:38:30
07:45:45
07:53:00
Time (hr:min:sec)

Figure 8.28

1 has increased since it is no longer importing any reactive power from the second subsystem. This event is more significant for the busbars of island 2 since the disconnected loads are located within this area. Its effect on the busbars of island 1 is almost unnoticeable since the system load decrease and generation reactive power increase have opposite effect on the voltage.

The disconnection of lines 10 and 41 at 7:30:30 has enormously increased the power flow of line 33 (weak tie) linking busbar 55 to busbar 53 (figure 8.27). This has increased the line losses in the system as is clearly shown in figure (8.28) where the difference between the total load and total generation has increased. Consequently the voltage at the two ends of this line has dropped busbars 55, 53, 54). The importation of reactive power to island 1 together with these line losses resulted in an increase of the reactive power of generator 4. The increase of the voltage at busbars 5 and 49 is due to the total load drop. Because of this voltage variation, the transfer of energy from island 3 to island 1 and the reduction of the system load, the reactive power produced locally in subsystem 1 is decreased.

In this case the load shedding following the disconnection of line 33 (formation of a third island) also resulted in a decrease of the total load. This has led to a decrease in the generated reactive power in the 2 islands and a rise of the voltage, although the elements of island 3 (generator 4 and bus 55) are affected more.

Restoring the system after these incidents has an opposite effect on the power generation, load and voltage. These variables tend to return to their initial values prior to the disturbance. However, because of the system load variation, these variables do not have exactly the same values as in their initial steady state.

Load re-connection increased the total load and the generation of the system as well as the reactive power of all the generating units. It has also resulted in a voltage reduction throughout the system.

8.3.3 Synchronisation of independent islands

To be able to re-synchronise a split system successfully these conditions have to be satisfied: the difference between the frequencies and the voltages at the two ends of the line to be re-connected should be within a predetermined tolerance. The selected tolerances for this test are 0.08 Hz, 0.8 rd, and 0.15 (p.u.), however smaller values can be used if necessary.

Using the zoom in graphics capability the interval of time during which the synchronisation took place has been selected from figure (8.18) and magnified in figure (8.19). This figure shows clearly the frequencies of the two ends (busbars 52 and 54) of line 32 locked together at 7:36:00. Since some load has been removed from the system an overshoot of the new common frequency occurred. A similar thing can be said about the synchronisation of the remaining islands which occurred at 7:41:00. The frequencies at bus 53 and 54 are equal since both these two busbars are within island 1.

Figure (8.25) shows that the voltages at both ends (busbars 52, 54 and busbars 55, 53) of the synchronised lines (32 and 33) are within the required tolerance for their synchronisation.

The re-synchronisation was successful in the first attempt in both cases because the required conditions were satisfied immediately after the command for re-synchronisation was issued from the scenario given above. This is shown in figure (8.19) and the selected events shown in table (8.2).

8.4 ACCURACY OF THE SYSTEM

To check the accuracy of a given solution, all that can be done is to compare this response with supposedly more accurate results. The more accurate results are the responses obtained using smaller time steps.

Time	Events
8/2/1985.07:24:00	total system load 385.609MW updated
	.
8/2/1985.07:24:50	total system load 387.071MW updated
8/2/1985.07:25:00	* $open: line send breaker 32
8/2/1985.07:25:00	* $open: line send breaker 24
8/2/1985.07:25:00	* $open: line send breaker 19
8/2/1985.07:25:00	* $wait: wait for 00000.00:01:30
8/2/1985.07:25:00	network top. now 1 island and 30 nodes
8/2/1985.07:25:00	total system load 387.364MW updated
	.
8/2/1985.07:26:20	total system load 389.903MW updated
8/2/1985.07:26:30	* $open: line send breaker 15
8/2/1985.07:26:30	* $wait: wait for 00000.00:04:00
8/2/1985.07:26:30	network top. Now 2 islands and 30 nodes
8/2/1985.07:26:30	total system load 389.995MW updated
8/2/1985.07:26:33	load 19 disconnected by under-frequency protection
8/2/1985.07:26:36	load 16 disconnected by under-frequency protection
8/2/1985.07:26:40	total system load 372.798MW updated
	.
8/2/1985.07:30:30	total system load 379.222MW updated
8/2/1985.07:30:30	* $open: line send breaker 10
8/2/1985.07:30:30	* $open: line send breaker 41
8/2/1985.07:30:30	* $wait: wait for 00000.00:01:30
8/2/1985.07:30:40	total system load 379.501MW updated
	.
8/2/1985.07:31:50	total system load 381.456MW updated
8/2/1985.07:32:00	* $open: line send breaker 33
8/2/1985.07:32:00	* $wait: wait for 00000.00:04:00

Table continued

8/2/1985.07:32:00	network top. Now 3 islands and 30 nodes
8/2/1985.07:32:00	total system load 381.735MW updated
8/2/1985.07:32:03	load 23 disconnected by under-frequency protection
8/2/1985.07:32:10	total system load 367.056MW updated
	.
8/2/1985.07:35:50	total system load 372.96MW updated
8/2/1985.07:36:00	* $sync: line send breaker 32 timeout 00:28:00
	.
8/2/1985.07:36:00	* note: synchronisation initiated
8/2/1985.07:36:00	* $wait: wait for 00000.00:05:00
8/2/1985.07:36:00	* islands 3 and 1 synchronised
8/2/1985.07:36:00	total system load 373.228MW updated
8/2/1985.07:36:00	network top. Now 2 islands and 30 nodes
8/2/1985.07:36:10	total system load 373.496MW updated
	.
8/2/1985.07:37:10	total system load 375.107MW updated
8/2/1985.07:37:13	* $close: line send breaker 19
8/2/1985.07:37:13	* $close: line send breaker 24
8/2/1985.07:37:13	* $close: line send breaker 15
8/2/1985.07:37:20	total system load 375.375MW updated
	.
8/2/1985.07:40:50	total system load 381.011MW updated
8/2/1985.07:41:00	* $sync: line send breaker 33 timeout.00:28:00
8/2/1985.07:41:00	* note: synchronisation initiated
8/2/1985.07:41:00	* $wait: wait for 00000.00:02:00
8/2/1985.07:41:00	* islands 1 and 2 synchronised
8/2/1985.07:41:00	network top. Now 1 island and 30 nodes
8/2/1985.07:41:00	total system load 381.279MW updated

table continued

8/2/1985.07:42:20	total system load 383.426MW updated
8/2/1985.07:42:28	$close: line send breaker 41
8/2/1985.07:42:28	$close: line send breaker 10
8/2/1985.07:42:30	total system load 383.694MW updated
	.
8/2/1985.07:43:00	total system load 384.5MW updated
8/2/1985.07:43:00	* $close: load breaker 19
8/2/1985.07:43:00	* $wait: wait for 0000.00:02:30
8/2/1985.07:43:10	total system load 389.501MW updated
	.
8/2/1985.07:45:20	total system load 393.033MW updated
8/2/1985.07:45:30	* $close: load breaker 16
8/2/1985.07:45:30	* $wait: wait for 0000.00:01:30
8/2/1985.07:45:30	total system load 407.496MW updated
	.
8/2/1985.07:47:00	total system load 410.029MW updated
8/2/1985.07:47:00	* $close: load breaker 23
8/2/1985.07:47:10	total system load 426.253MW updated

Table 8.2 Event log of the islanding and synchronisation test

A number of tests have been carried out using the same scenario (disconnecting load 4) with different time steps and tolerances. The first test is run using the usual time step of 1 second and tolerances of 0.001 (p.u.) and 0.005 (p.u.). The second one used a time step and a tolerance a tenth of the previous test variables. Three variables have been selected for this comparison. A rapidly changing variable (frequency), a slowly changing variable (drum pressure variation), and an intermediate variable (turbine high pressure). These variables are related to generator 1.

In order to compare the simulation results obtained from varying the tolerance and the time steps, the three categories of variables mentioned above are grouped into different figures. Figure (8.29) shows the rapidly changing variable, figure (8.33) shows the intermediate variable and figure (8.35) shows the slow variable.

In figure (8.29) the two plots obtained by varying the time step when the parallel simulator was run as a stand-alone package are compared to the frequency obtained by running the simulator with the control system. Figure (8.30) is an expanded form of figure (8.29). In this expansion the behaviour of the frequency of generator 1 during the first seconds following the disturbance is highlighted. This shows clearly that the frequency calculated at a time step of 1 second is not as accurate as the one calculated at a time step of 0.1 second during this period. However, in both cases the oscillations are damped out at the same time (about 10 seconds after the event) at close steady state values. Since the time range of interest is of a long-term nature, and neglecting these oscillations does not affect the dynamic stability of the system and does not require too many iterations to converge, the results obtained using a time step of 1 second are acceptable. Also the fact that the implementation of the load frequency control reduces the steady state errors, as shown in figure (8.29), suggests the advantage of using larger time steps so that the simulator would run in real-time. If the system shows a numerical instability then a smaller time step can be used.

The approximation due to linearising the response of the differential equations from one step to the next when using a 1 second time step can be clearly seen in the expanded plots of figure (8.30).

The reduction of the Newton-Raphson tolerance to a tenth of its original value has not shown any difference in the case of a 1 second time step (figure 8.31) and an insignificant one in the second case (figure 8.32). Therefore, it can be concluded that the selected tolerances provide quite accurate solutions at a minimum cost in terms of number of iterations and solutions of the set of linear equations.

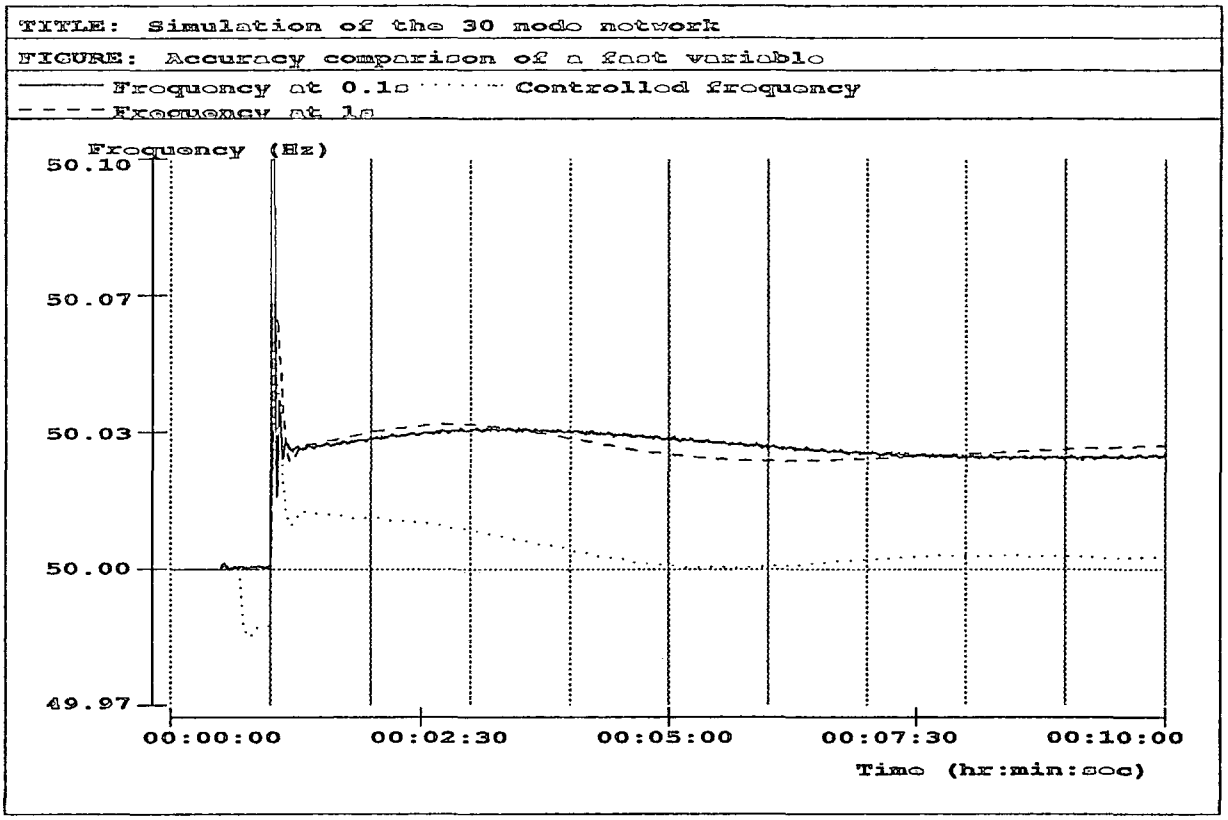


Figure 8.29

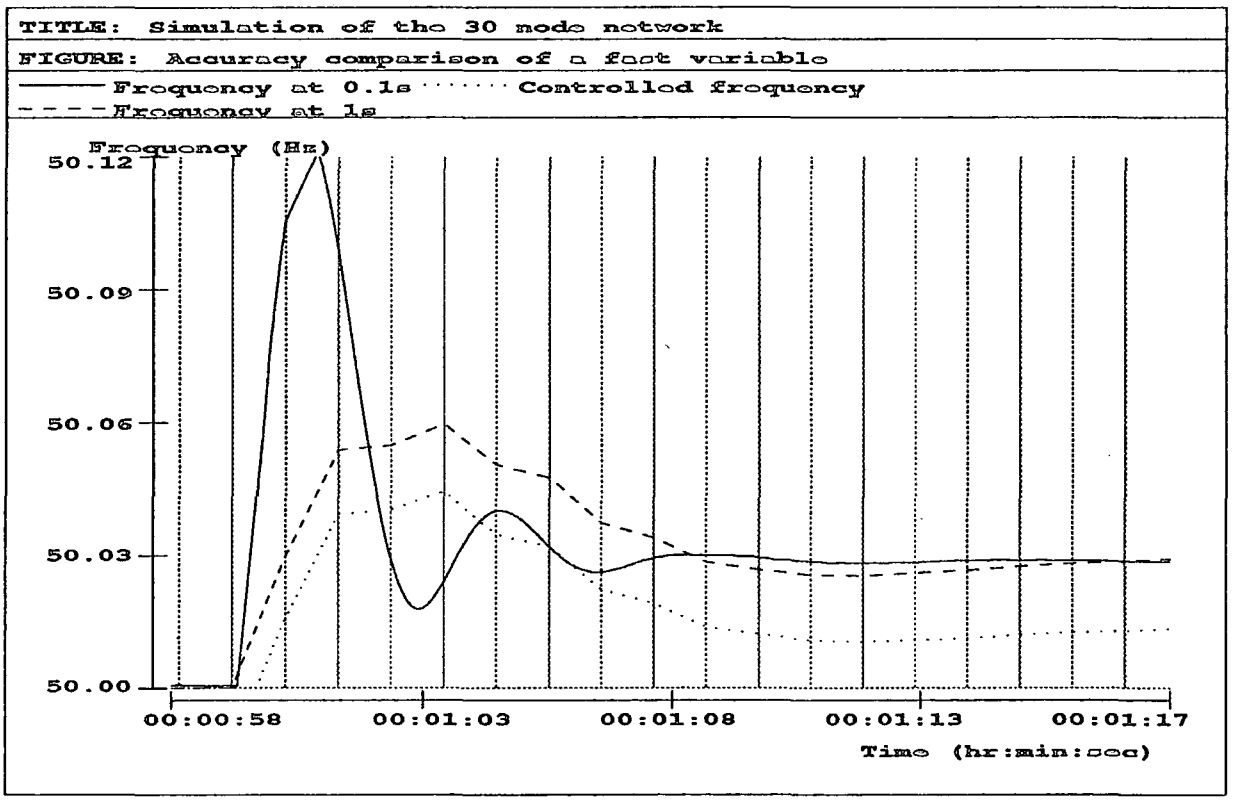


Figure 8.30

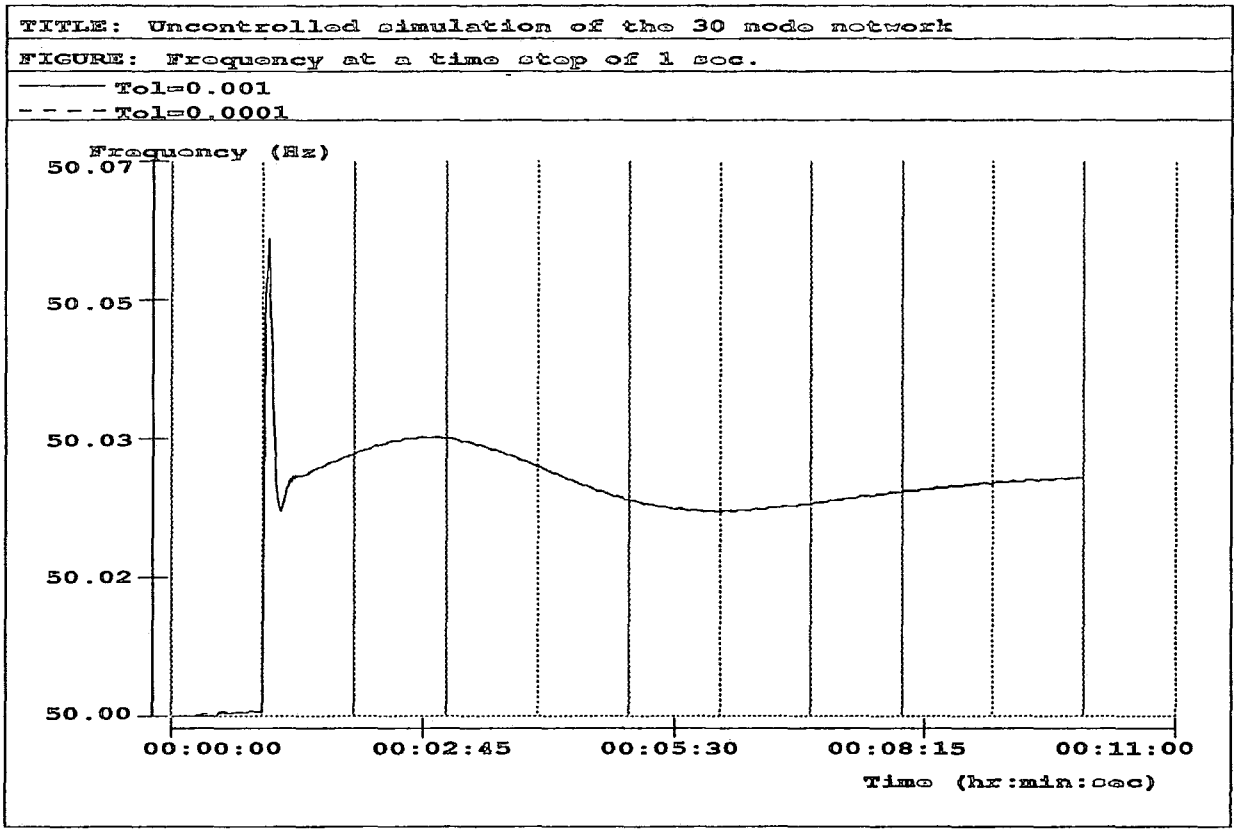


Figure 8.31

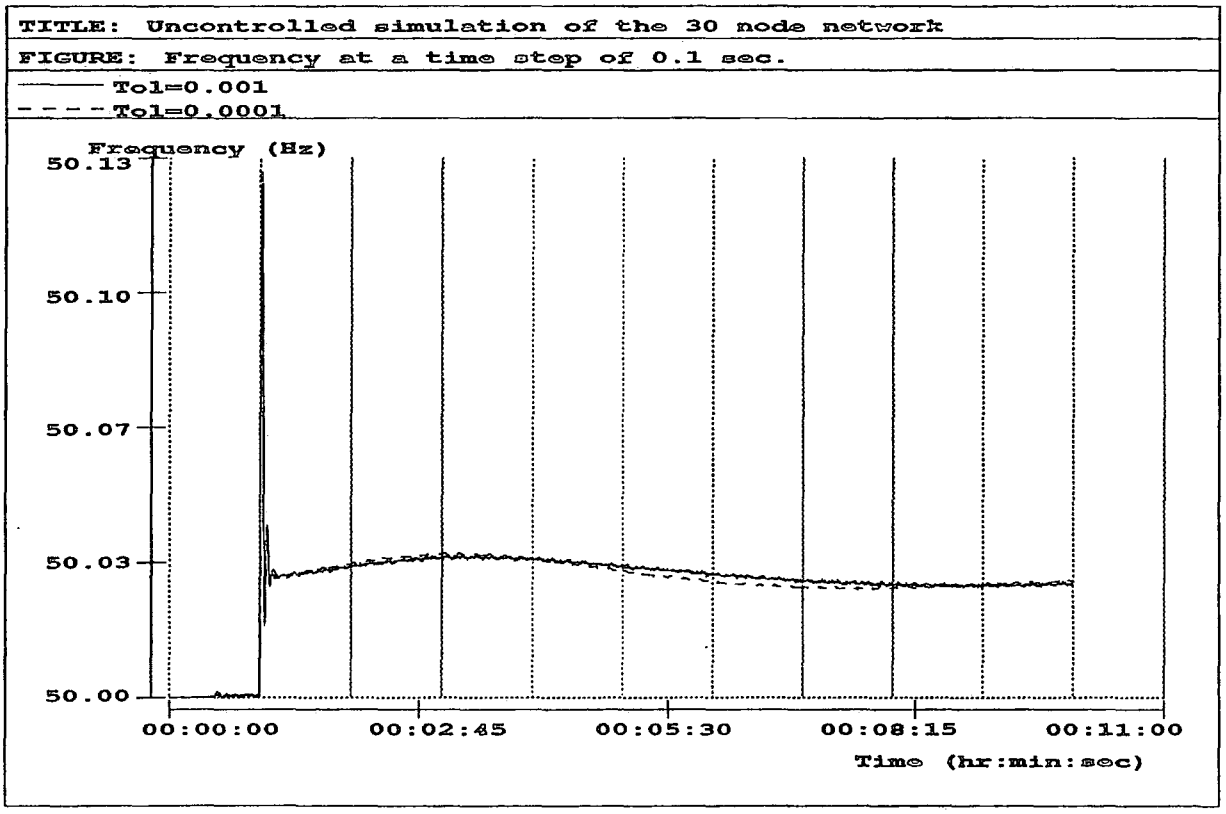


Figure 8.32

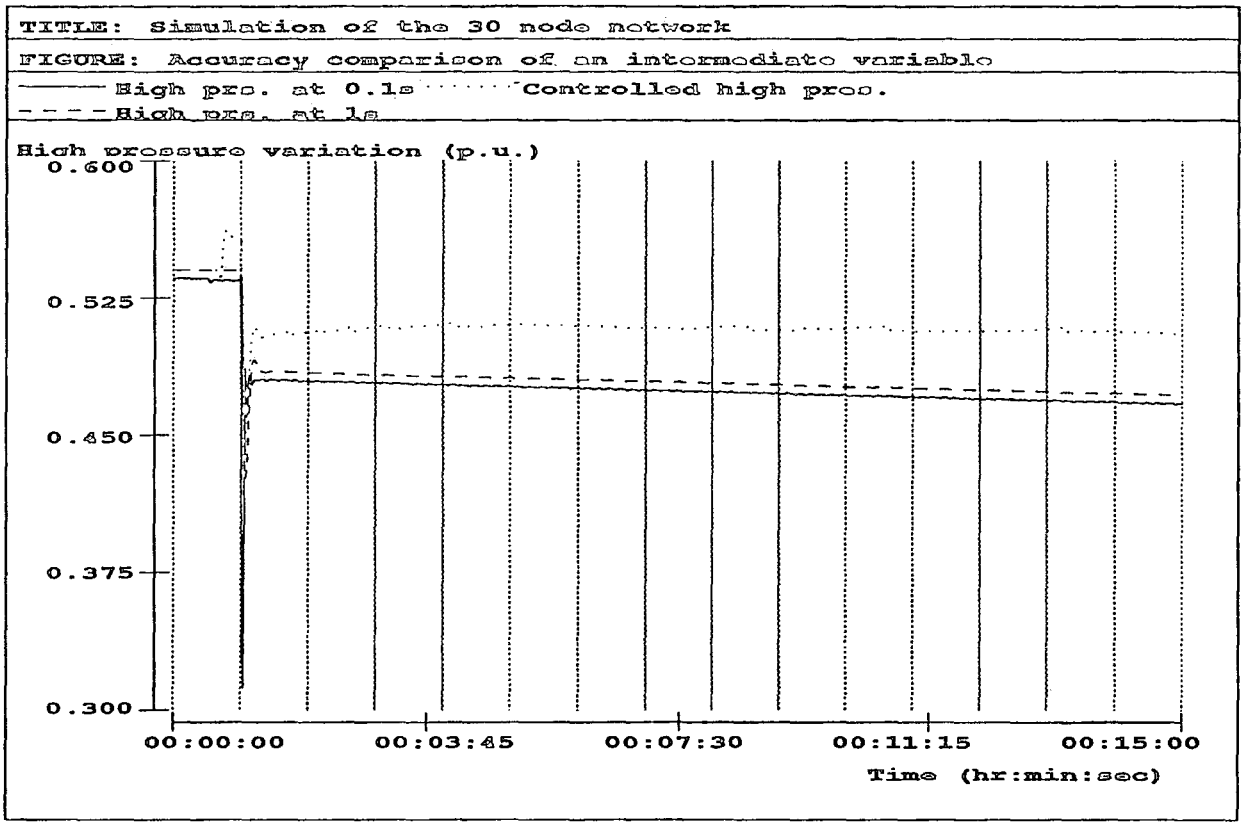


Figure 8. 33

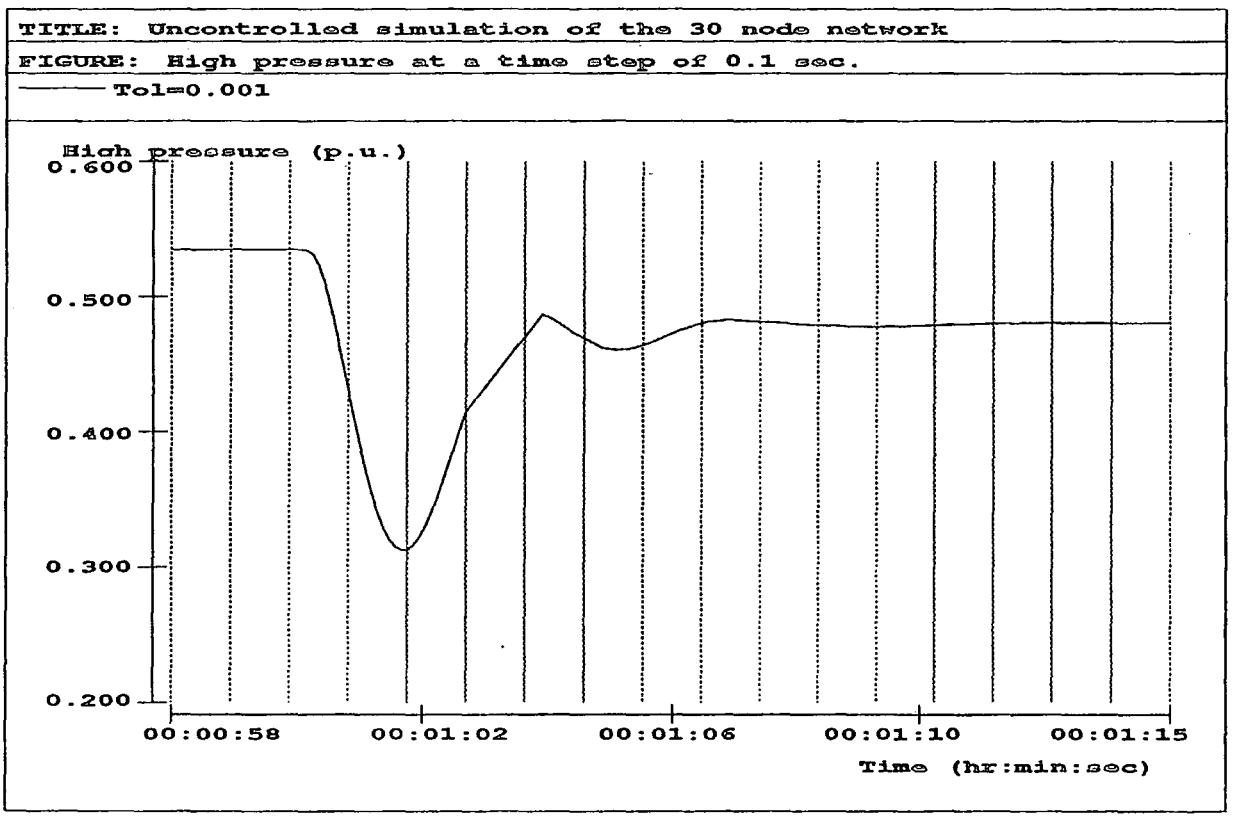


Figure 8.34

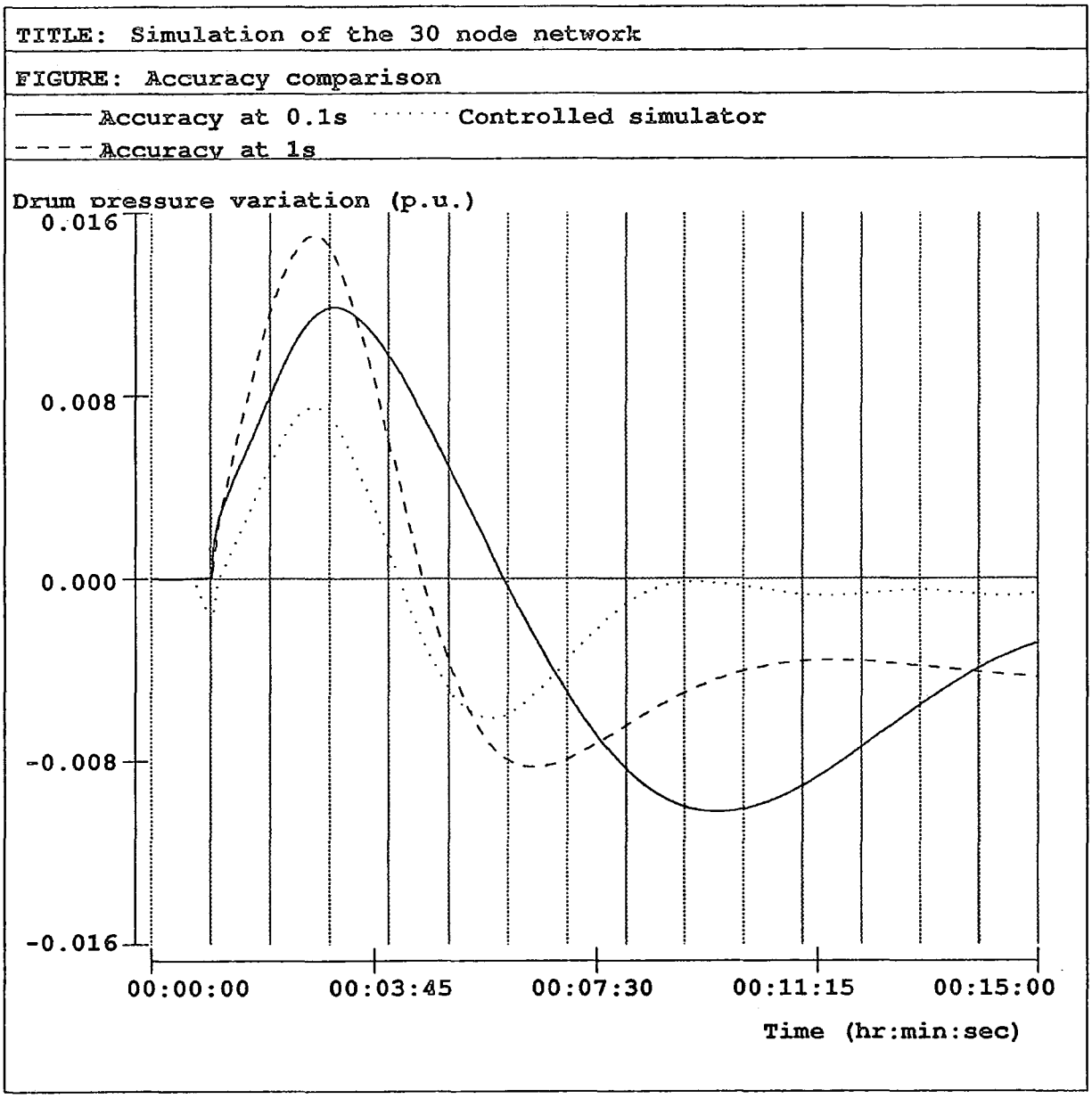

TITLE: Simulation of the 30 node network
FIGURE: Accuracy comparison
Accuracy at 0.1s
Controlled simulator
Accuracy at 1s
Drum pressure variation (p.u.)
0.016
0.008
0.000
-0.008
-0.016
00:00:00
00:03:45
00:07:30
00:11:15
00:15:00
Time (hr:min:sec)

Figure 8.35

The intermediate variable (turbine 1 high pressure) curves (figure 8.33) also show that the results obtained from using a time step of 0.1 second were different from those obtained by using a time step of 1 second during the first seconds following the disturbance. Fortunately, both curves damped out at close steady state values which were different to the controlled results.

Figure (8.35) shows the variation of the drum pressure. The controlled curve indicates that the boiler reaches the steady state faster when the control package is run (about 8 minutes after the disturbance). The smaller time steps are not of great importance for slow variables because of their large time constants.

Figures (8.30) and (8.34) show that the frequency and the turbine high pressure do not respond instantaneously to the load variation, as is the case for the 1 second time step test, when a 0.1 second time step is used. This shows the simulated time taken by the generator and the turbine to respond to the changes occurring in the system. The time taken by the turbine to react is greater than that taken by the generator since the latter is the first element in the system to respond to disturbances.

8.5 IMPROVEMENT OF THE STABILITY OF THE SYSTEM

The modifications brought to the simulator have not only improved the speed, but have also greatly improved the representation of the behaviour of the network elements under severe conditions.

An example of this is the instability which used to be caused during islanding if the simulator was running as a stand-alone package (figure 8.36). Since the governor limitation was not included to force the generator output within the permissible limits, the generator output reached unrealistic values (figure 8.38). Furthermore, the voltage limitations were modelled within the iteration loop preventing the system from recovering after a disturbance, since the voltage values which violate the limits were used as initial guesses for the next time step (figure 8.40). The modification of these models as well as the correction

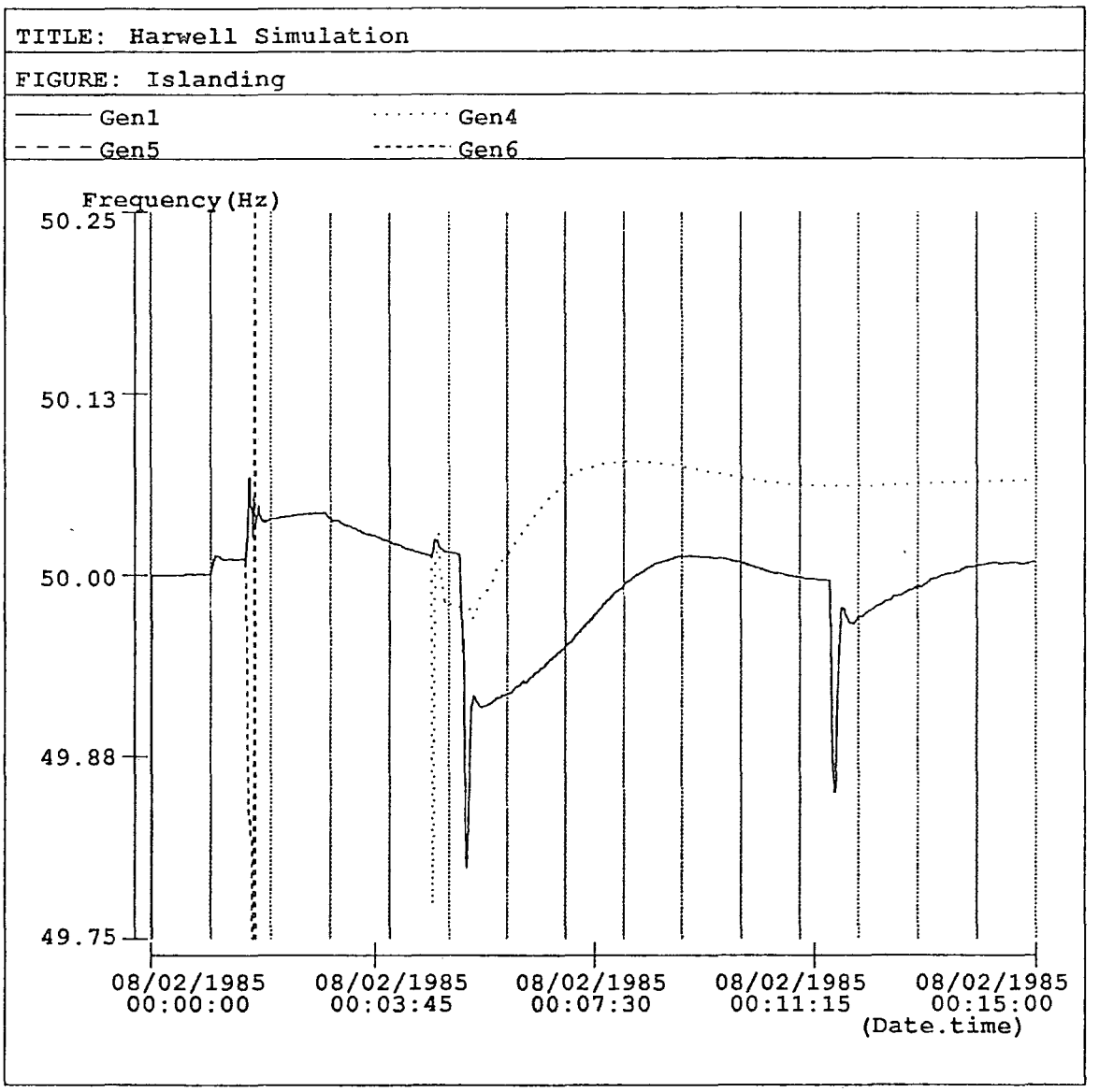

TITLE: Harwell Simulation
FIGURE: Islanding
Gen1
Gen4
Gen5
Gen6
Frequency(Hz)
50.25
50.13
50.00
49.88
49.75
08/02/1985 00:00:00
08/02/1985 00:03:45
08/02/1985 00:07:30
08/02/1985 00:11:15
08/02/1985 00:15:00
(Date.time)

Figure 8.36

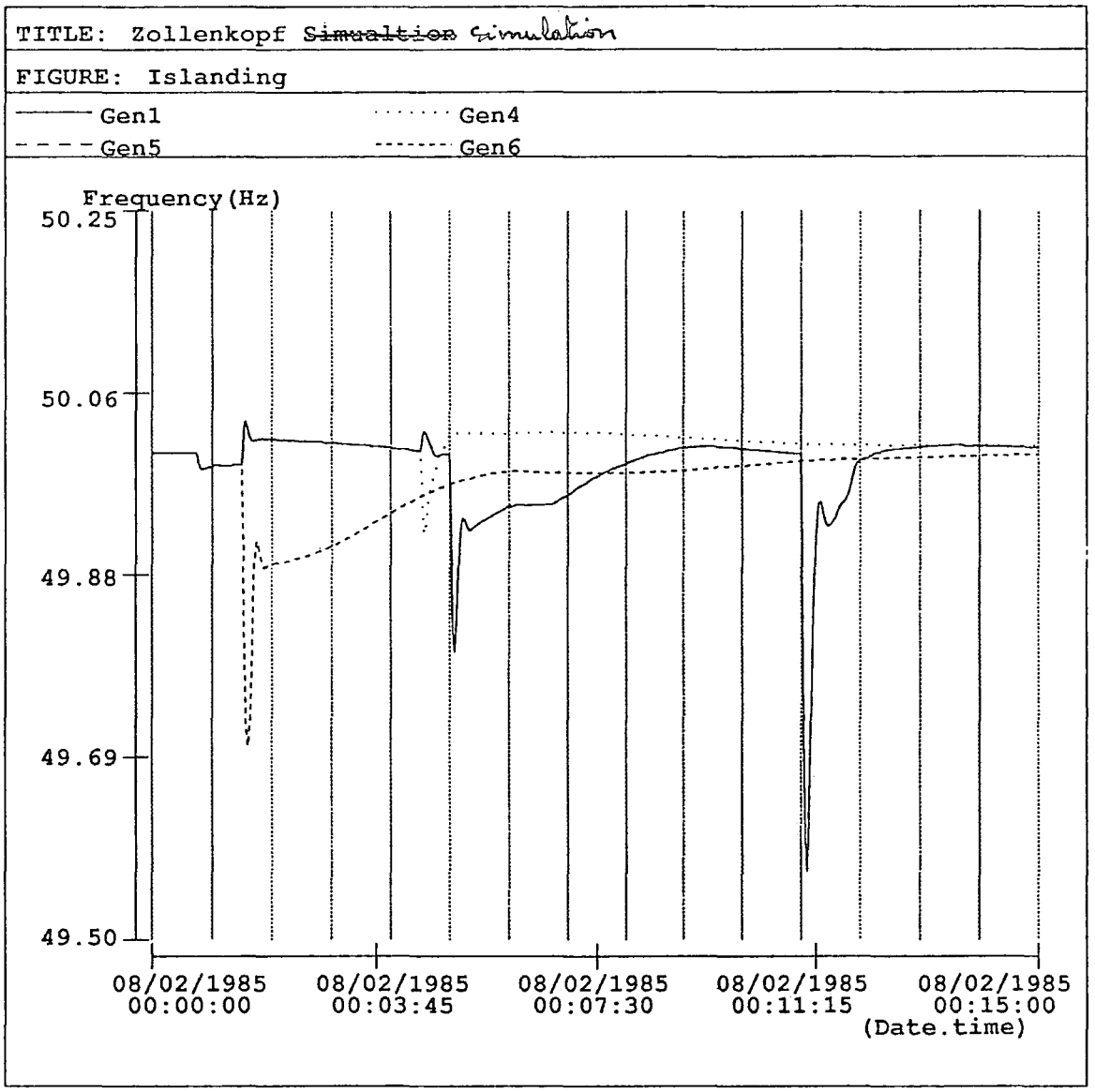

TITLE: Zollenkopf ~~Simualtion~~ simulation
FIGURE: Islanding
Gen1
Gen4
Gen5
Gen6
Frequency(Hz)
50.25
50.06
49.88
49.69
49.50
08/02/1985 00:00:00
08/02/1985 00:03:45
08/02/1985 00:07:30
08/02/1985 00:11:15
08/02/1985 00:15:00
(Date.time)

Figure 8.37

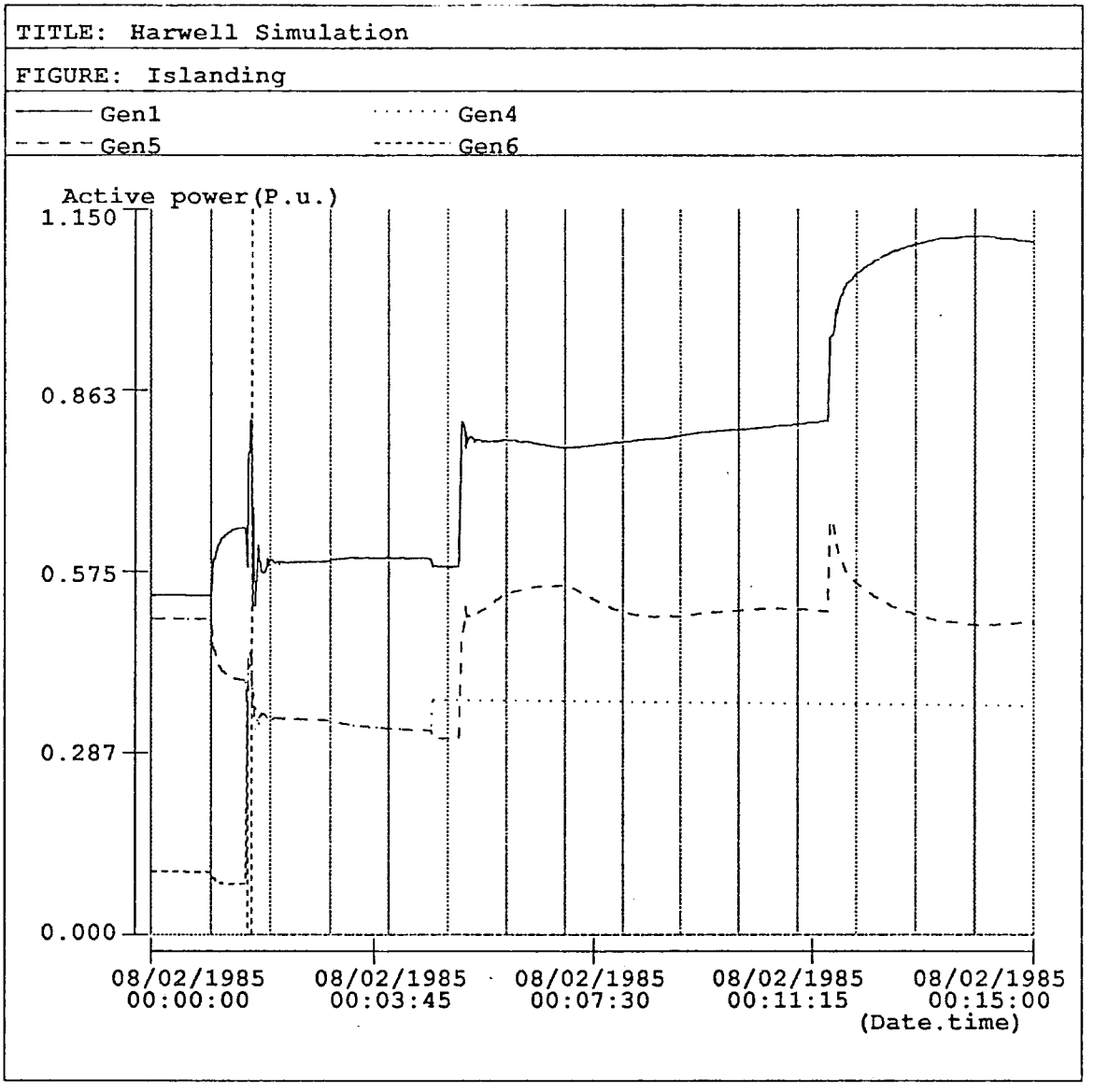

TITLE: Harwell Simulation
FIGURE: Islanding
Gen1
Gen4
Gen5
Gen6
Active power(P.u.)
1.150
0.863
0.575
0.287
0.000
08/02/1985 00:00:00
08/02/1985 00:03:45
08/02/1985 00:07:30
08/02/1985 00:11:15
08/02/1985 00:15:00
(Date.time)

Figure 8.38

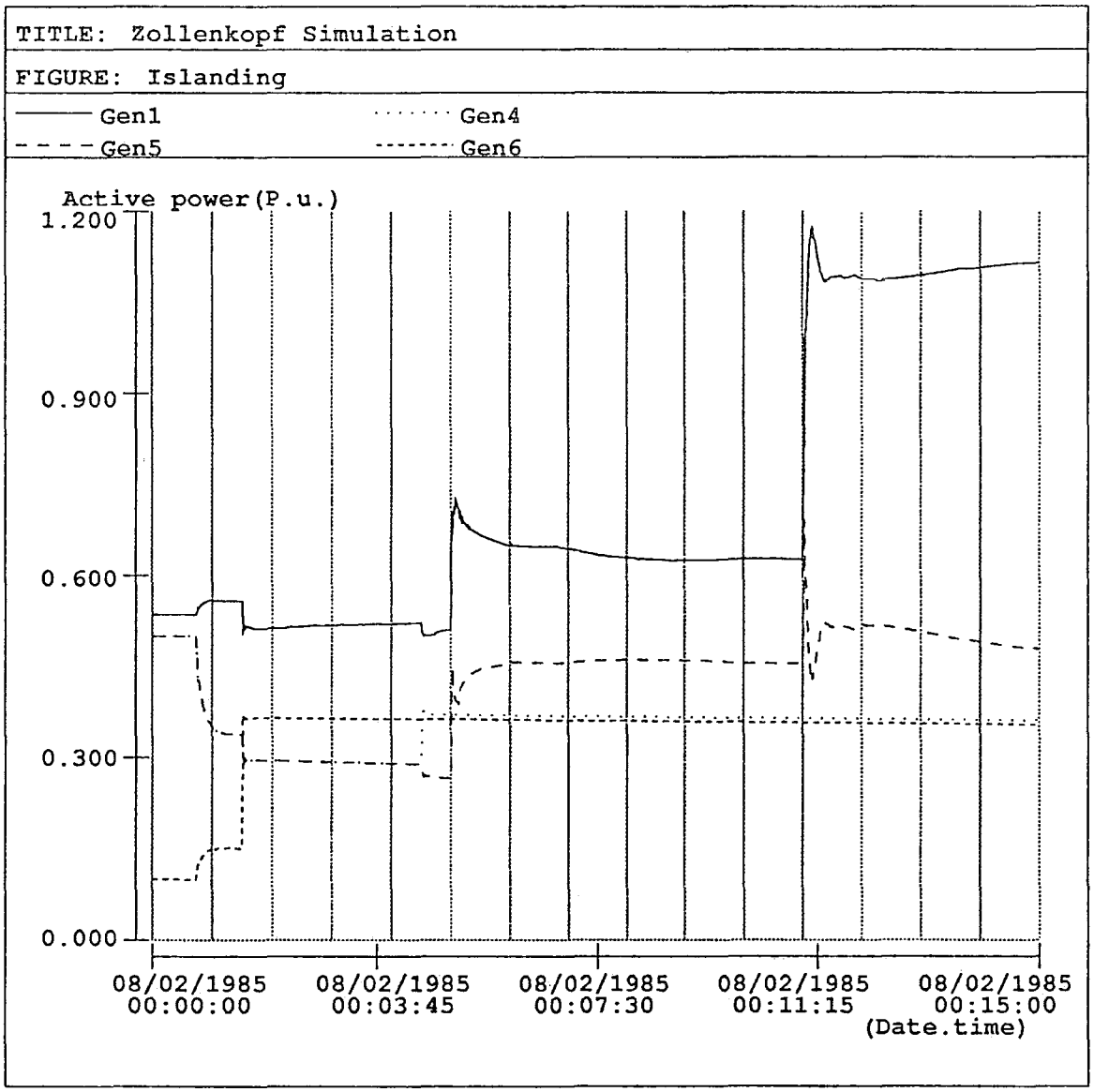

TITLE: Zollenkopf Simulation
FIGURE: Islanding
Gen1
Gen4
Gen5
Gen6
Active power(P.u.)
1.200
0.900
0.600
0.300
0.000
08/02/1985 00:00:00
08/02/1985 00:03:45
08/02/1985 00:07:30
08/02/1985 00:11:15
08/02/1985 00:15:00
(Date.time)

Figure 8.39

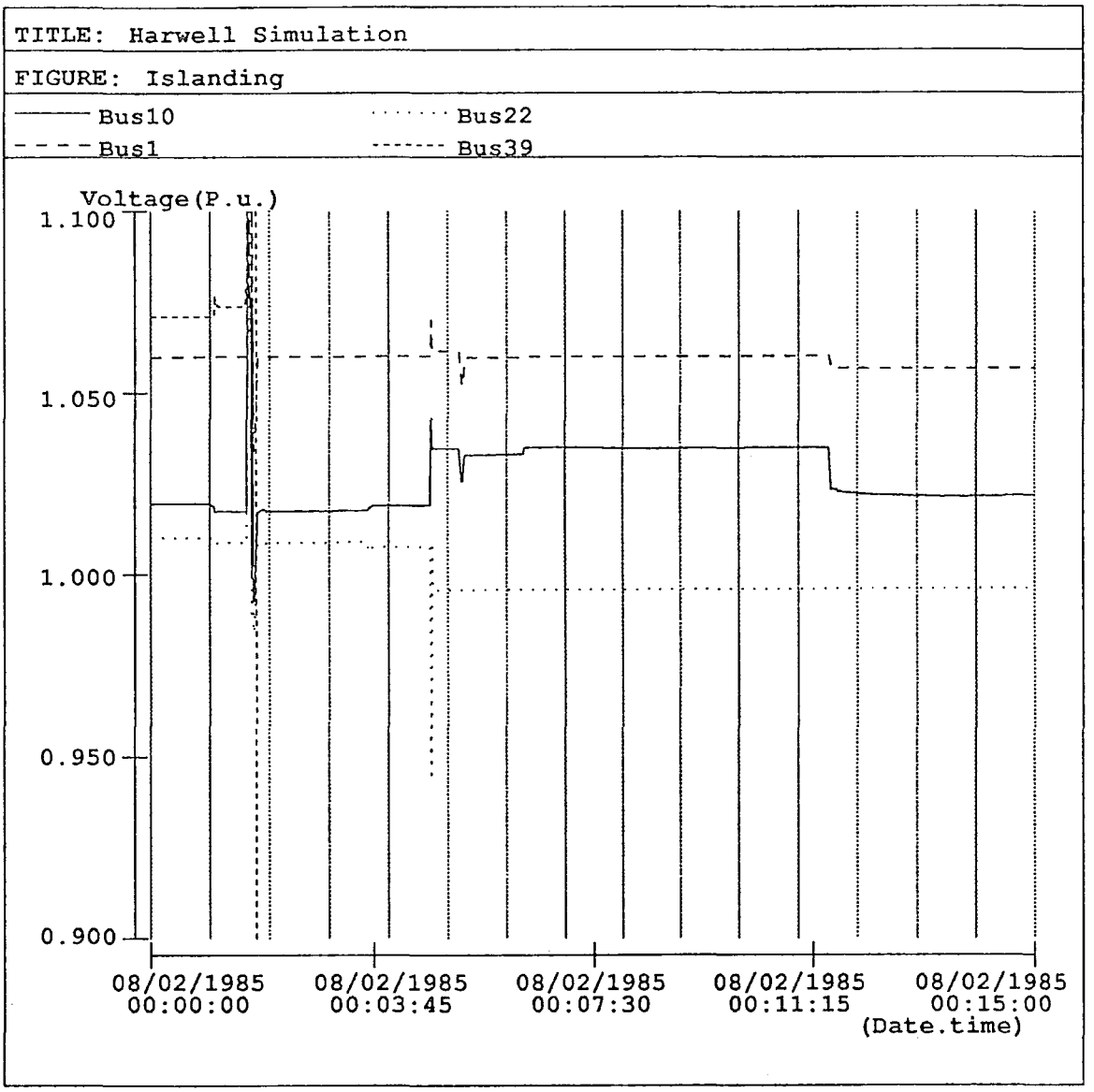

TITLE: Harwell Simulation
FIGURE: Islanding
Bus10
Bus22
Bus1
Bus39
Voltage(P.u.)
1.100
1.050
1.000
0.950
0.900
08/02/1985 00:00:00
08/02/1985 00:03:45
08/02/1985 00:07:30
08/02/1985 00:11:15
08/02/1985 00:15:00
(Date.time)

Figure 8.40

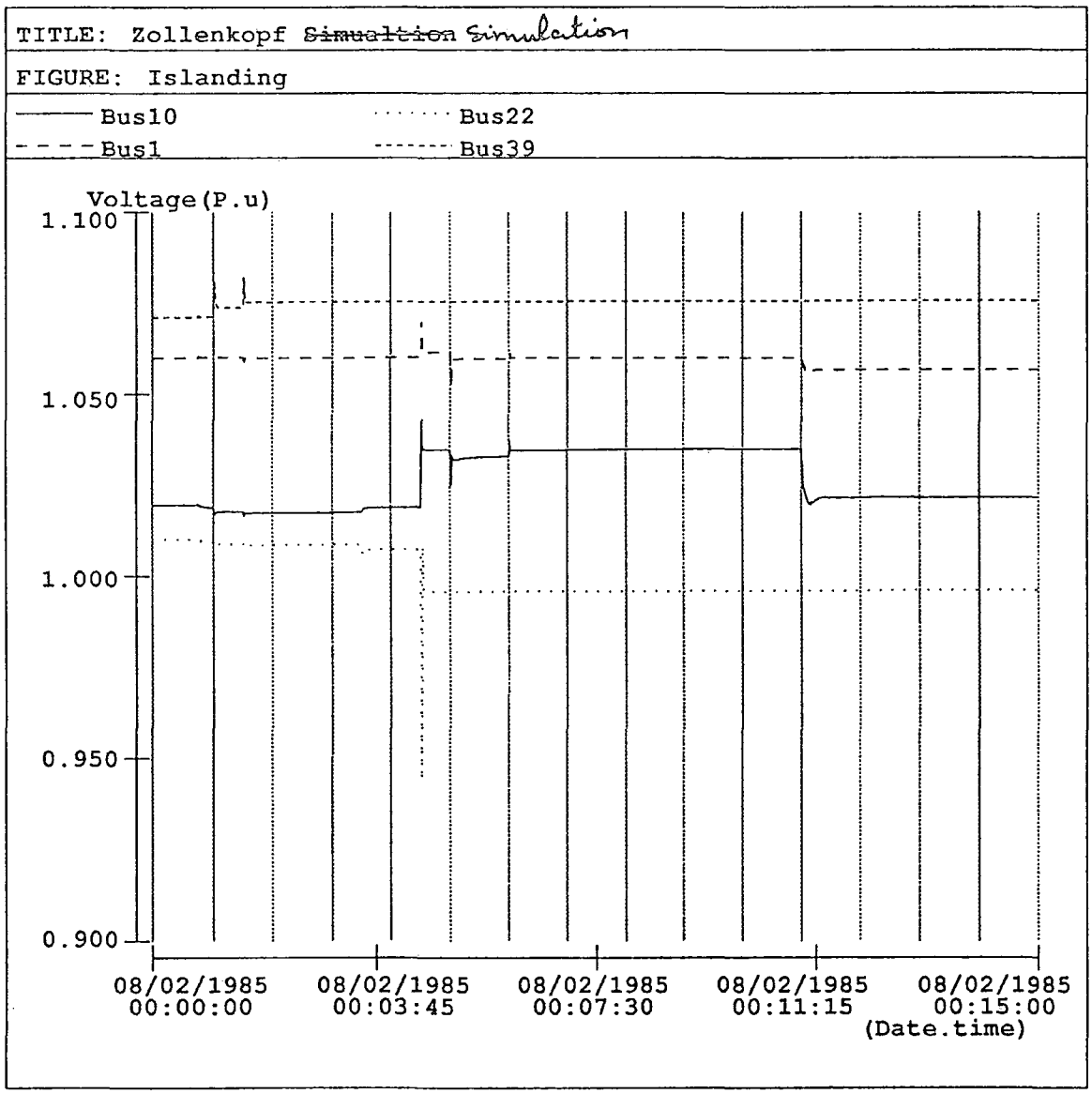

TITLE: Zollenkopf ~~Simualtion~~ Simulation
FIGURE: Islanding
Bus10
Bus22
Bus1
Bus39
Voltage(P.u)
1.100
1.050
1.000
0.950
0.900
08/02/1985 00:00:00
08/02/1985 00:03:45
08/02/1985 00:07:30
08/02/1985 00:11:15
08/02/1985 00:15:00
(Date.time)

Figure 8.41

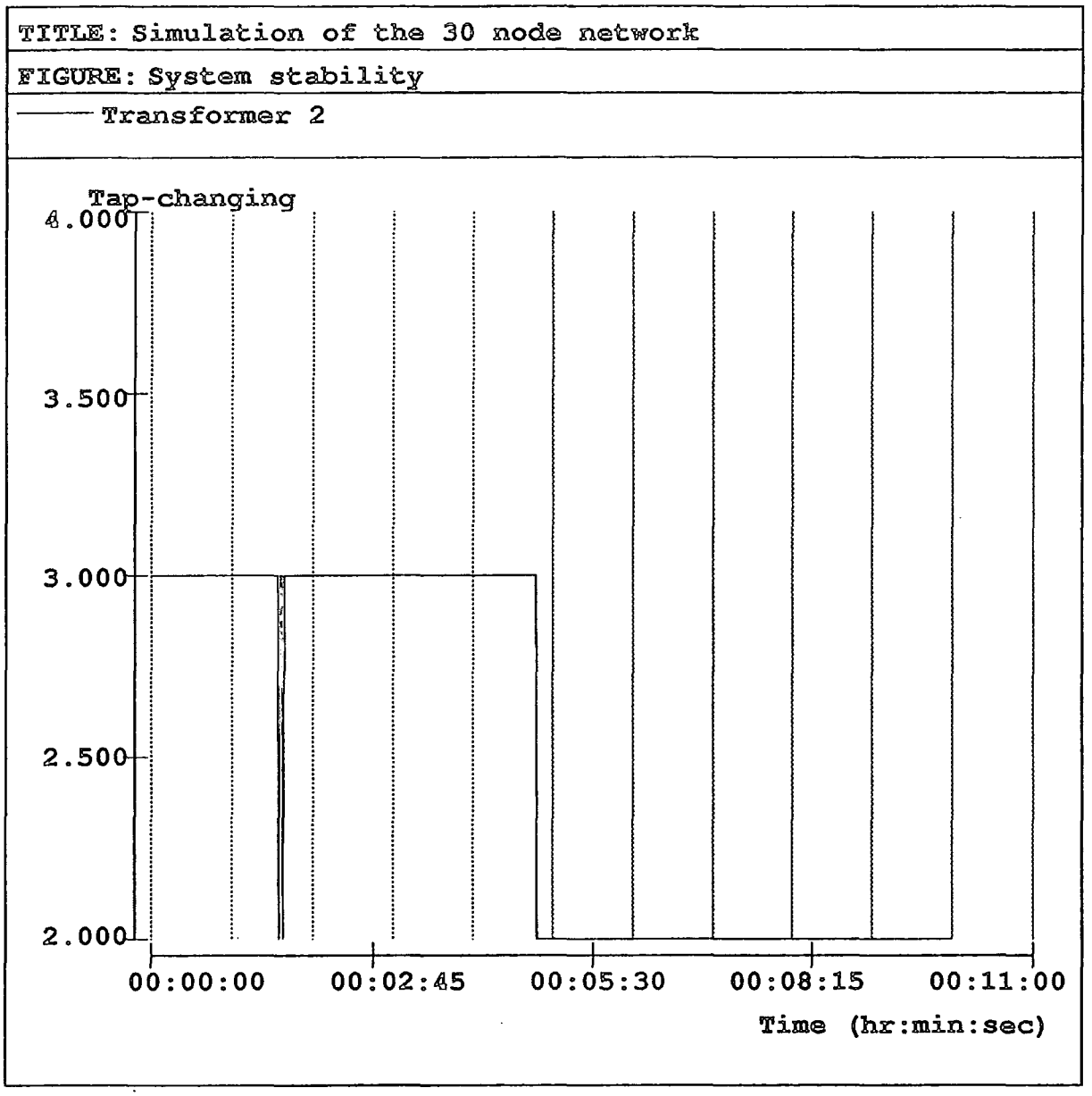

TITLE: Simulation of the 30 node network
FIGURE: System stability
Transformer 2
Tap-changing
4.000
3.500
3.000
2.500
2.000
00:00:00
00:02:45
00:05:30
00:08:15
00:11:00
Time (hr:min:sec)

Figure 8.42

of the load model have allowed the system to stabilise properly after islanding. Also the modification of protection modelling in general and particularly the automatic under-frequency load shedding is beneficial in preventing the total collapse of the system as shown in the islanding and re-synchronisation test (section 8.3).

This has highlighted another problem which involved the interdependence of the different island frequencies. The elimination of this problem permitted the different islands to behave as completely independent systems.

Figures (8.37), (8.39), and (8.41) compared to figures (8.36), (8.38), and (8.40) respectively show the present system to be more stable than the previous one. Figure (8.42) illustrates the transformer tap-changing to control the voltage in island 1.

8.6 THE SIMULATOR TIMING

The previous sections are mainly concerned with the behaviour of the dynamic and static components of the network. They stress the reliability, robustness and accuracy of the simulator. This section presents the performance of the present algorithm and the modifications made to the programs and mathematical models. The emphasis in this case is mainly on the computation speed of the simulator. In order to highlight the performance the new technique has been implemented in the parallel simulator running on the Array Processor hosted by a Perkin Elmer PE 3230 minicomputer, and in the centralised (host only) simulator. Two test networks have been used for this purpose: the 30, and 118 node IEEE standard systems. The developed algorithm has also been implemented in the parallel decomposed simulator running on a single Perkin Elmer minicomputer. The results are presented in the form of charts where the processing times to be compared are plotted together. The time steps are set to 1 second except for the timing of the CEGB system where a time step of 0.25 second is chosen. The CEGB network[61] has been used to show the performance of the simulator at smaller time steps and under severe network conditions.

8.6.1 Parallel simulator timing

This simulator is running on the Array Processor where the arithmetical calculations are performed, and a host computer which is used for the IO tasks as well as some minor calculations.

An average computation time, which includes the disconnection of load 4 in the case of the IEEE 30 and 118 node networks, over a time period of ten minutes is presented in figure (8.43). The average number of iterations and calls (solutions) to the subroutines to solve the set of linear equations are given in figure (8.44).

The performance of the new algorithm has been found to be far greater than the old version. When simulating the 30 node network the Array Processor (numerical calculation) execution time was nearly nine times faster than the previous OCEPS parallel version. The overall timing of this network has been improved by a factor of more than 3. For the standard IEEE 118 network the Array Processor timing has been improved by a factor of about 20 and the overall timing was about 15 times faster than the old parallel version. The new algorithm also performed very well on the CEGB network data and ran at real-time (0.25 seconds). Compared with the standard version, the Array Processor time was approximately 11 times faster and the overall timing was about 9 times faster.

Compared with the clock time the 30 node network achieved a speed of five and a half times faster than real-time. The 118 network was about 4 times faster than real-time.

It can be noticed that the timing does not vary linearly with the size of the system since this is closely linked with the sparsity of the network.

The performance of the Zollenkopf algorithm is not the only cause of this big improvement of the simulator. The application of extrapolation has reduced the number of iterations necessary for the Newton-Raphson method to converge. This has saved about 20 % of the Array Processor time.

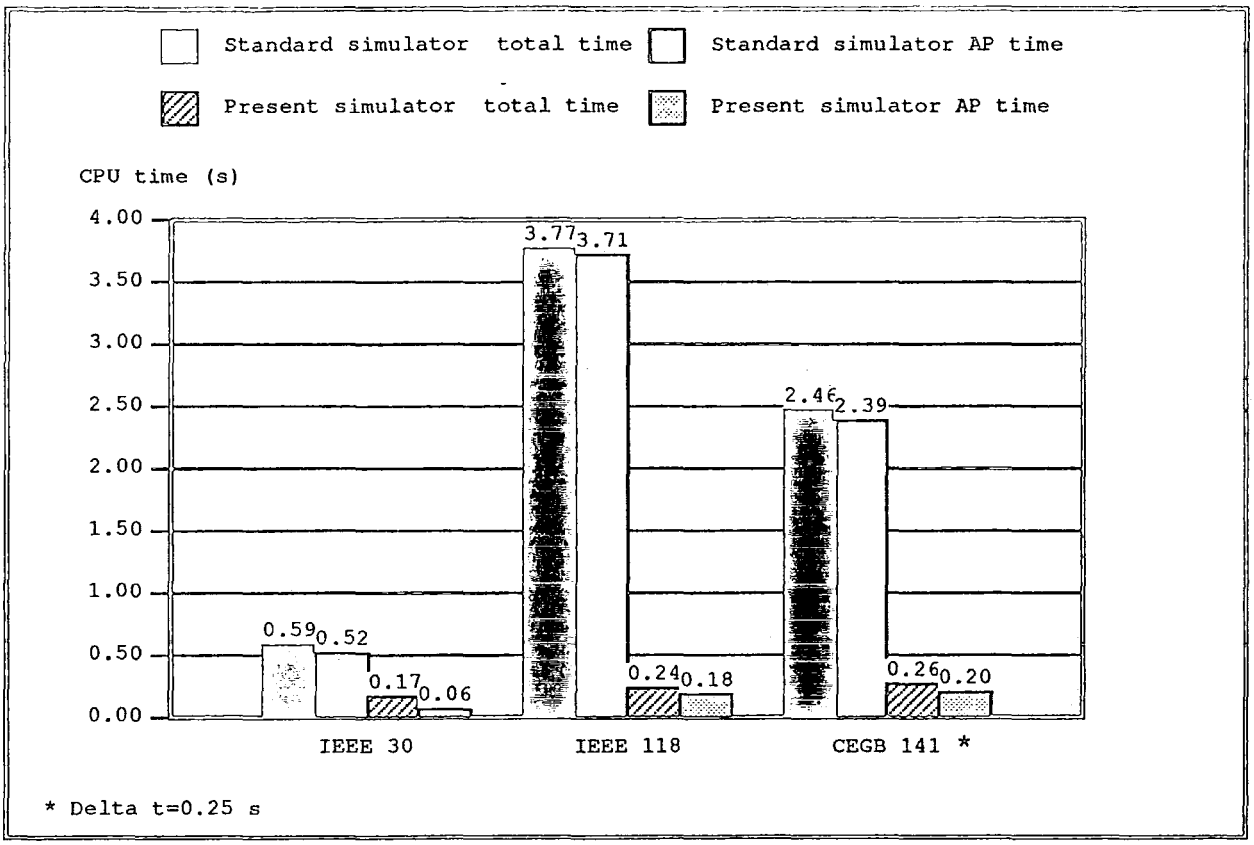


Figure 8.43 Parallel simulator timing

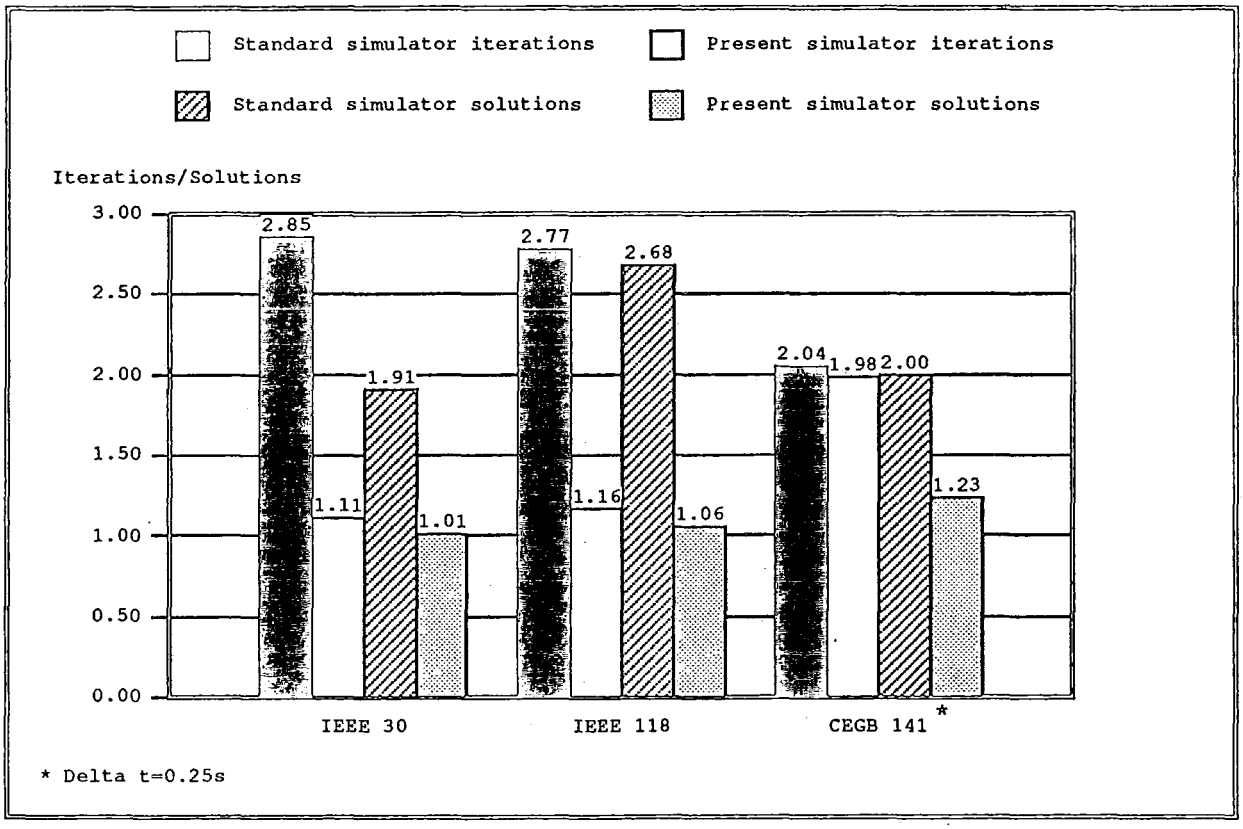


Figure 8.44 Number of iterations and solutions

8.6.2 Centralised simulator timing

In order to distinguish between the performance of the new developed algorithm and the speed-up due to the application of the Array Processor and the network decomposition, this test has been carried out. The comparison of the execution time obtained from applying the modified Zollenkopf algorithm and the Harwell library routines to both the parallel and host only simulators are presented in figure (8.45).

Compared with the results of the conventional single processor simulator (host computer only), the conventional parallel simulator was improved by a factor of about 2.5 when applied to the 30 node network, and a factor of about 2.24 when applied to the IEEE 118 node system. The performance of the simulator was reduced with the increase of the size of the network. Comparing the new centralised simulator execution time with the standard version a speed-up of about 4.48 was obtained in the case of simulating the 30 node system and a speed-up factor of about 6.08 was achieved when the 118 node network was tested. The 30 node system applied to the developed parallel simulator was improved by a factor of about 1.94 when compared with the new centralised simulator, and the 118 node network overall timing was improved by a factor of about 5.8. Thus it can be concluded that the contribution of the *bifactorisation* technique towards improving the execution time of the simulator is higher than the contribution of the Array Processor. Also this technique performs better on large systems (118 node network). This is due to the improved sparsity of large system (see tables (6.1) and (8.4)).

The developed algorithm is very fast and even the application of the simulator on the host only can simulate networks of at least 100 nodes in real-time if a time step of 1 second is used.

8.6.3 The Harwell and Zollenkopf routine timing

Figure (8.46) presents a comparison of the *Harwell* routines and the $2*2$ Zollenkopf algorithm used for repeatedly solving the set of linear equations. These results were obtained by simulating the IEEE 118 node system for one iteration on the centralised algorithm. The solution of the set of linear equations

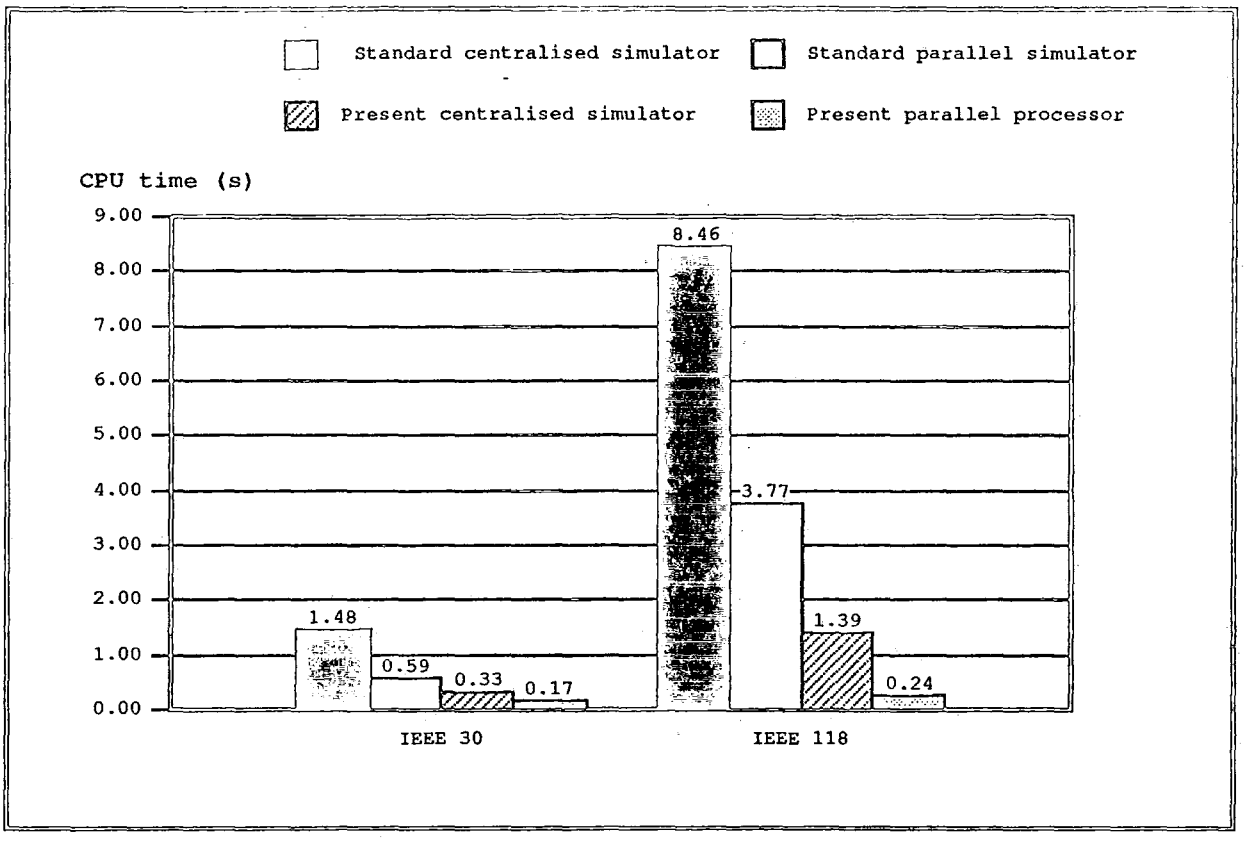


Figure 8.45 Performance of the centralised and parallel simulators

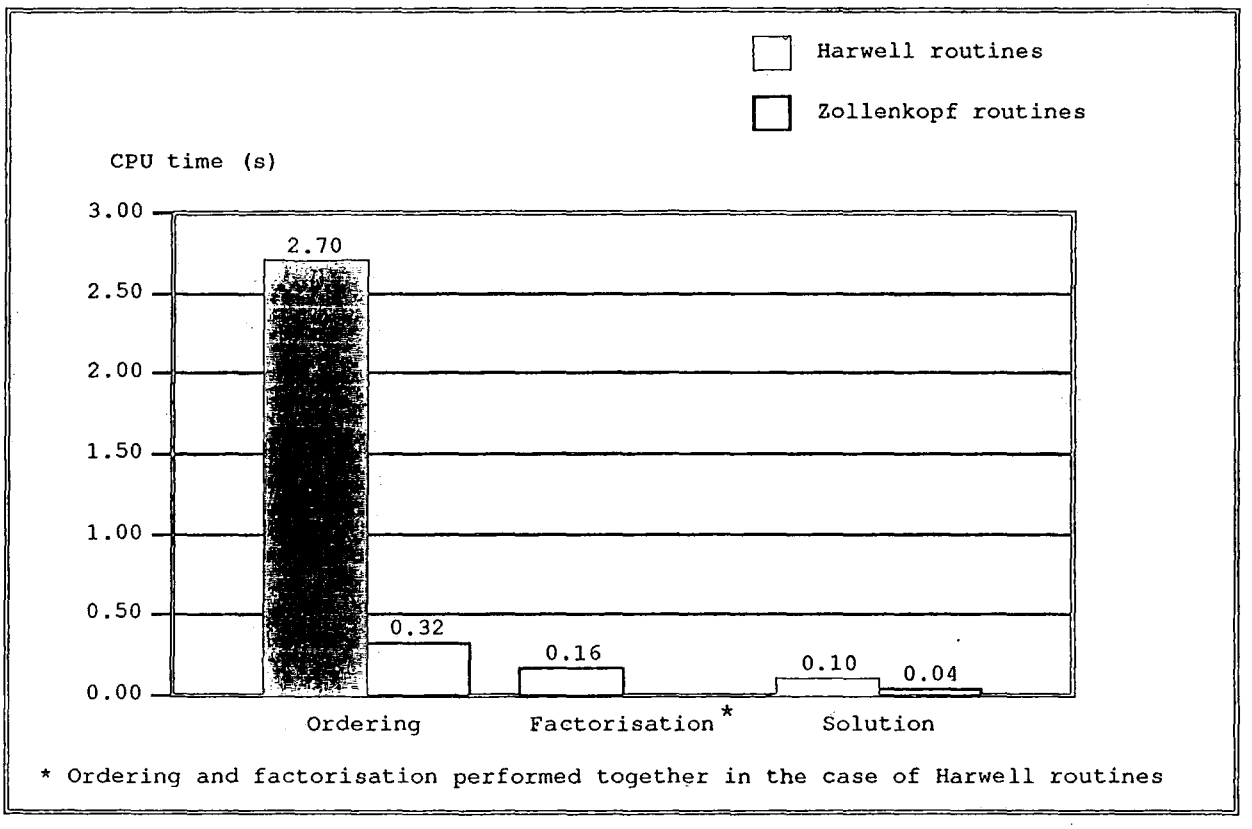


Figure 8.46 Comparison of the Harwell and Zollenkopf routines

using the modified Zollenkopf algorithm was faster than the *Harwell* routines, even if the ordering is performed. This can be attributed to the following factors

(1) The principle of the ordering scheme used in the first case necessitates less computation.
(2) The use of the 2*2 algorithm reduces the amount of calculation necessary to evaluate the state variable mismatch.
(3) The new simulator converges in fewer iterations.

This chart also shows that most of the solution time was spent in ordering. Since this is required only when the structure of the matrix is altered, a great amount of CPU time was saved.

8.6.3.1 Data transfer timing

The static data is transferred once prior to the start of the simulation in order to be initialised.

The purpose from transferring the selected dynamic data between the two machines (AP and PE) is to perform the measurements and telemetry, display the results graphically, and present the latest changes in the breakers and switch states etc. Therefore only the necessary data related to these functions together with the emergency messages are transferred from the Array Processor to the host computer. On the other hand the necessary dynamic data for the AP including the control signals and the topology information have to be sent from the host computer to the AP.

The direct memory access dynamic data transfer timing for one time step is gathered in table (8.3). This table also includes the size of the transferred dynamic data from and to the host computer.

From this table it can be concluded that most of the IO time is spent on the overheads. Therefore, the timing does not vary very much with the

size of the data. Consequently, transferring data is not a problem for larger networks.

Data transfer Host to AP					Data transfer Host from AP			
Network size	Data size (Kb)	Buffering time (s)	IO time (s)	Total time (s)	Data size (Kb)	Buffering time (s)	IO time (s)	Total time (s)
IEEE 30	1.98	0.009	0.008	0.017	2.68	0.011	0.008	0.019
IEEE 118	6.15	0.025	0.010	0.035	6.43	0.027	0.010	0.037
CEGB 141	7.91	0.032	0.010	0.042	7.27	0.030	0.010	0.040

Table 8.3 Host/AP data transfer timing

8.6.4 Decomposed simulator timing

The results presented in this chapter are obtained from the first algorithm presented in figure (7.2) since the calculation for both the off-line and on-line algorithms is the same.

The decomposed simulator was implemented on a single Perkin Elmer 3230 minicomputer and utilised the IEEE 30 and 118 standard networks. The latter was split into two and three areas and the former was split into three areas (figures 8.1-8.4). The only simulated event was the variation of the load over a period of 5 minutes. To illustrate the performance of this algorithm the computational time of the different areas of each system as well as the coordinating and the overall time were represented. The discretisation interval in this case was fixed to 1 second and the results presented are the average values over the simulated period. Since the speed of the simulator is closely linked with the number of iterations and calls to the routines for solving the linear sparse matrices, these are also included.

Since the systems cannot be split equally, a difference between the area timing can be minimised but not eliminated. The overall timing was based on the slowest area together with the coordinating time, as shown in figure (7.5).

Figures (8.47) and (8.48) show the performance of the standard decomposed simulator using the *Harwell* library routines to evaluate the area state

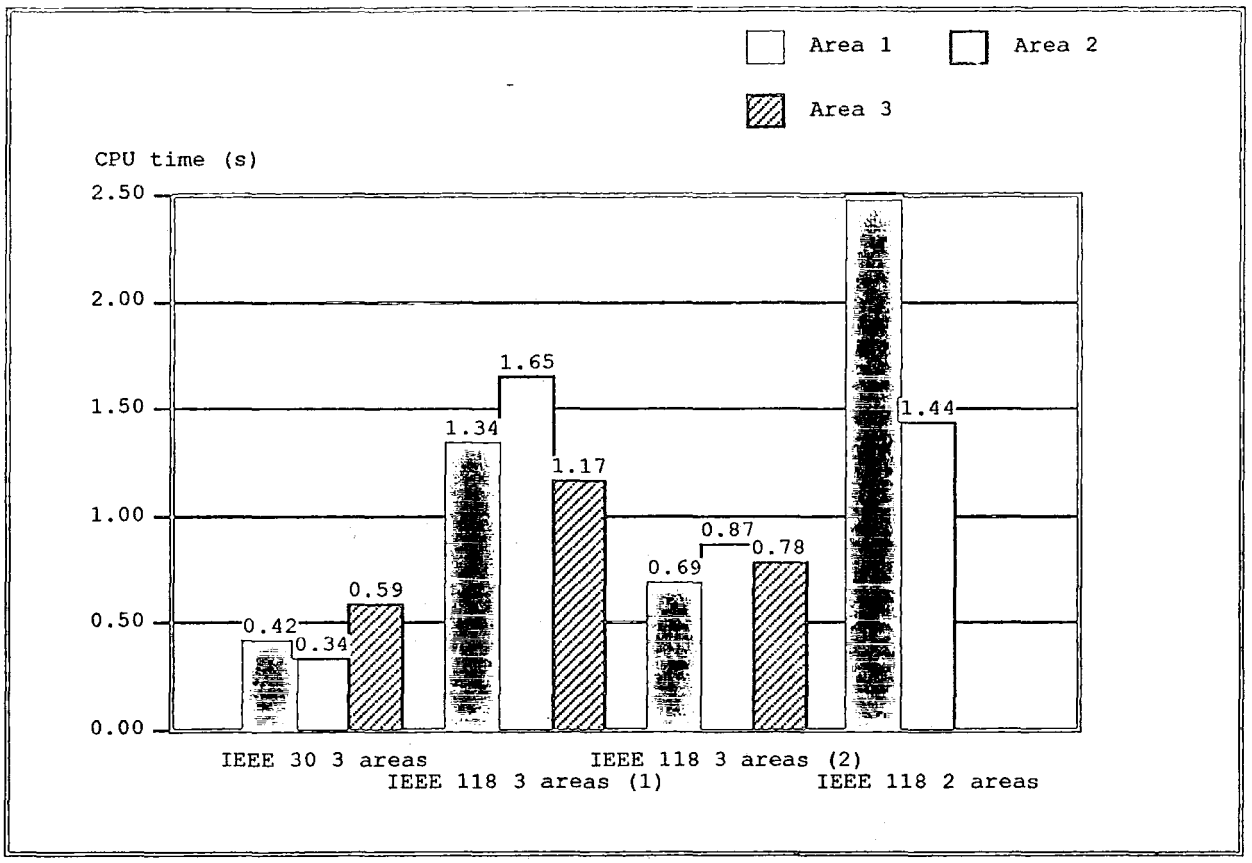


Figure 8.47 Standard decomposed simulator area timing

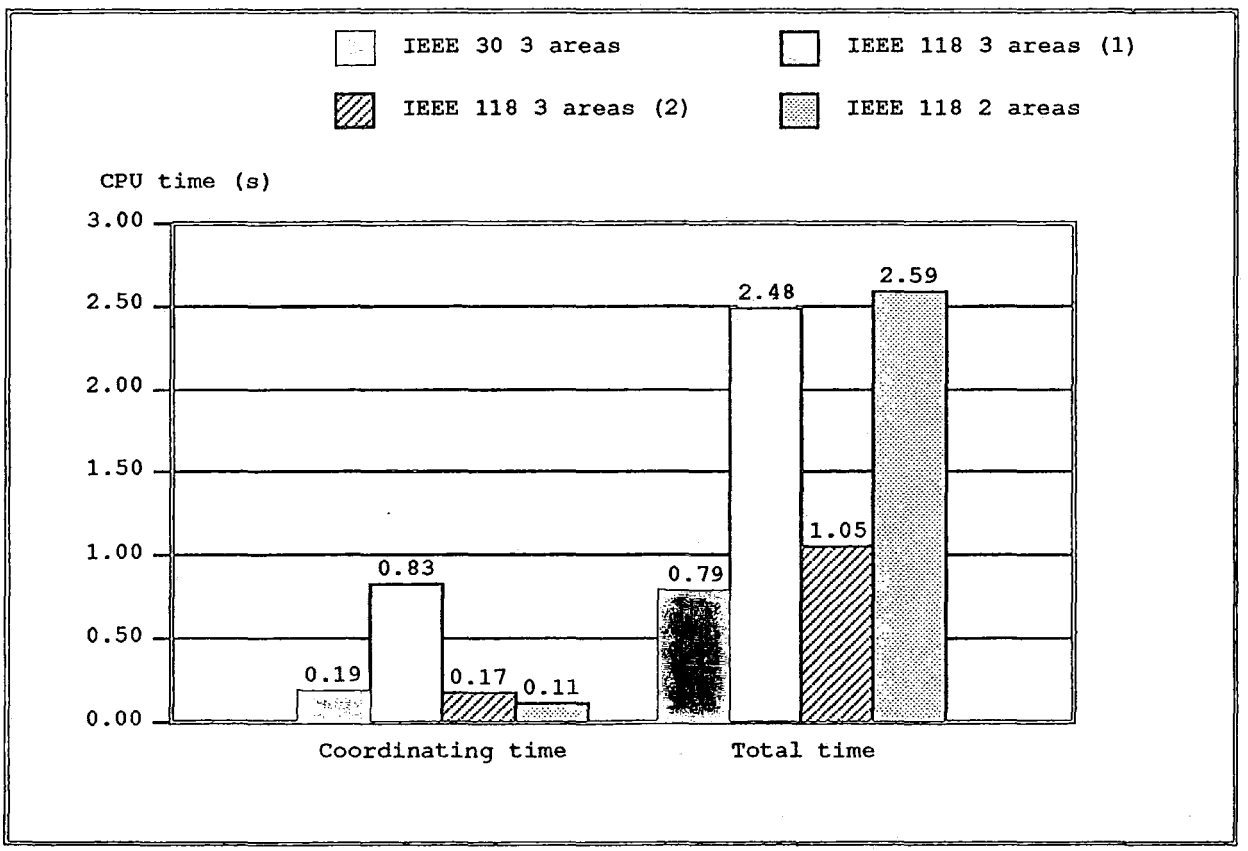


Figure 8.48 Standard decomposed simulator overall timing

variables. In this case the only test running in real-time was the 30 node network split into three areas. When the new algorithm was implemented all tests performed in real-time except the first case of decomposing the 118 node network into 3 areas as shown in figure (8.50). This is mainly due to the large number of tie lines. Splitting the 118 node network into 3 areas (first case) has resulted in 10 tie lines. The decomposition of this test network into 2 areas has resulted in 4 tie lines and the division of the 30 node system into 3 areas was achieved by disconnecting 6 tie lines. To illustrate the effect of the number of tie lines on the simulator execution time, especially the coordinating task speed, the 118 node network was subdivided into three areas by disconnecting 7 tie lines only. This new test has not only increased the speed of the main program (figure 8.50), it has also improved the numerical stability of the system by reducing the number of solutions of the set of linear equations as shown in figure (8.52). This has also reduced the execution time for the areas (figure 8.49). The first test (decomposition of the 118 node system into three areas by disconnecting 10 tie lines) needed more solutions to converge because one of the tie lines was a transformer.

The coordinating time shown in figures (8.48) and (8.50) show that both algorithms took about the same amount of time to converge. This is because the program coordinating the areas is the same in both the newly developed decomposed simulator and the standard one.

Although the 118 node system is larger than the 30 node system, the computation of the former was faster. This is due to the larger number of solutions (evaluation of the inverse matrix and state variable mismatch) required by the latter to converge and the fact that subsystems involved in the first case are more sparse as shown in table (8.4). Figure (8.52) shows that all the decomposed tests converged after one iteration and the centralised simulators took longer to converge. The present single processor simulator converged in fewer iterations and solutions than the old centralised version. This difference is due to the introduction of the linear extrapolation in the later case.

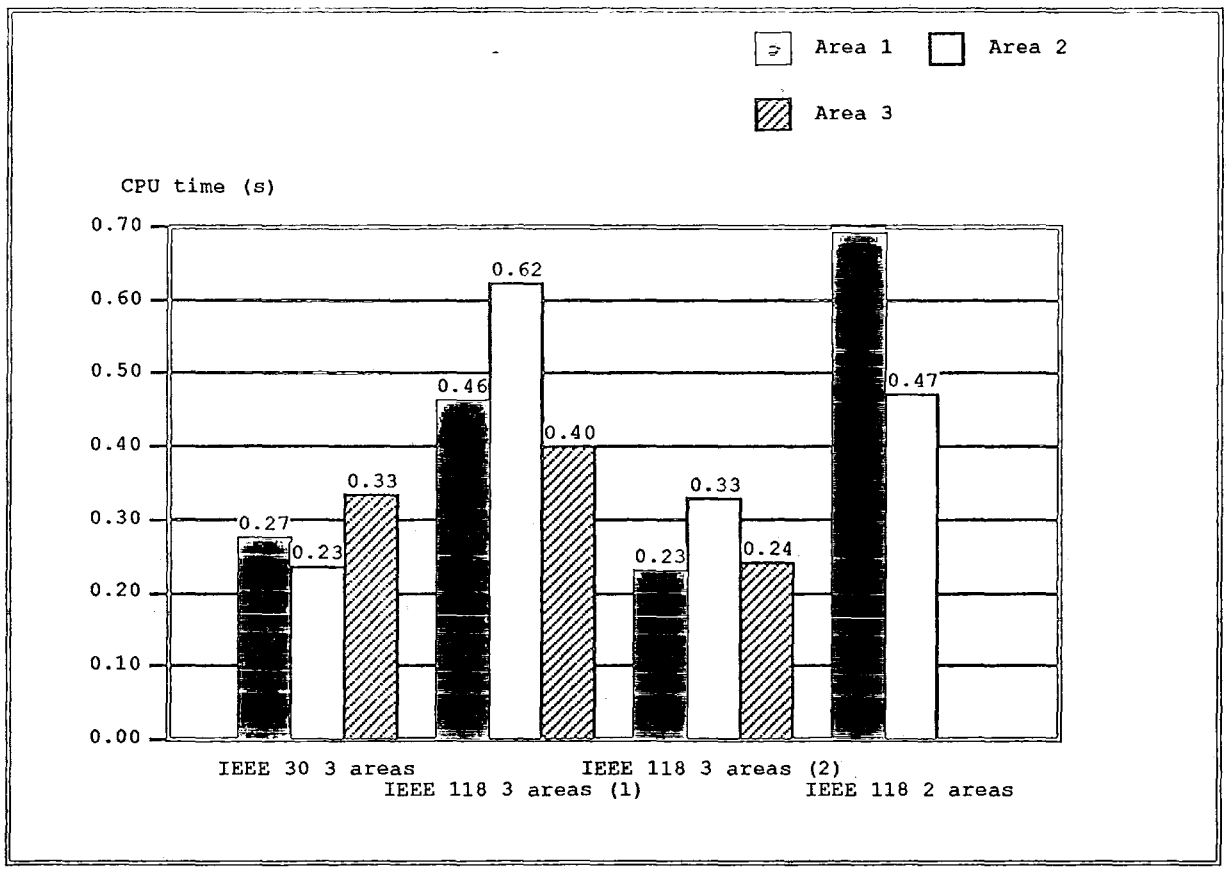


Figure 8.49 Present simulator area timing

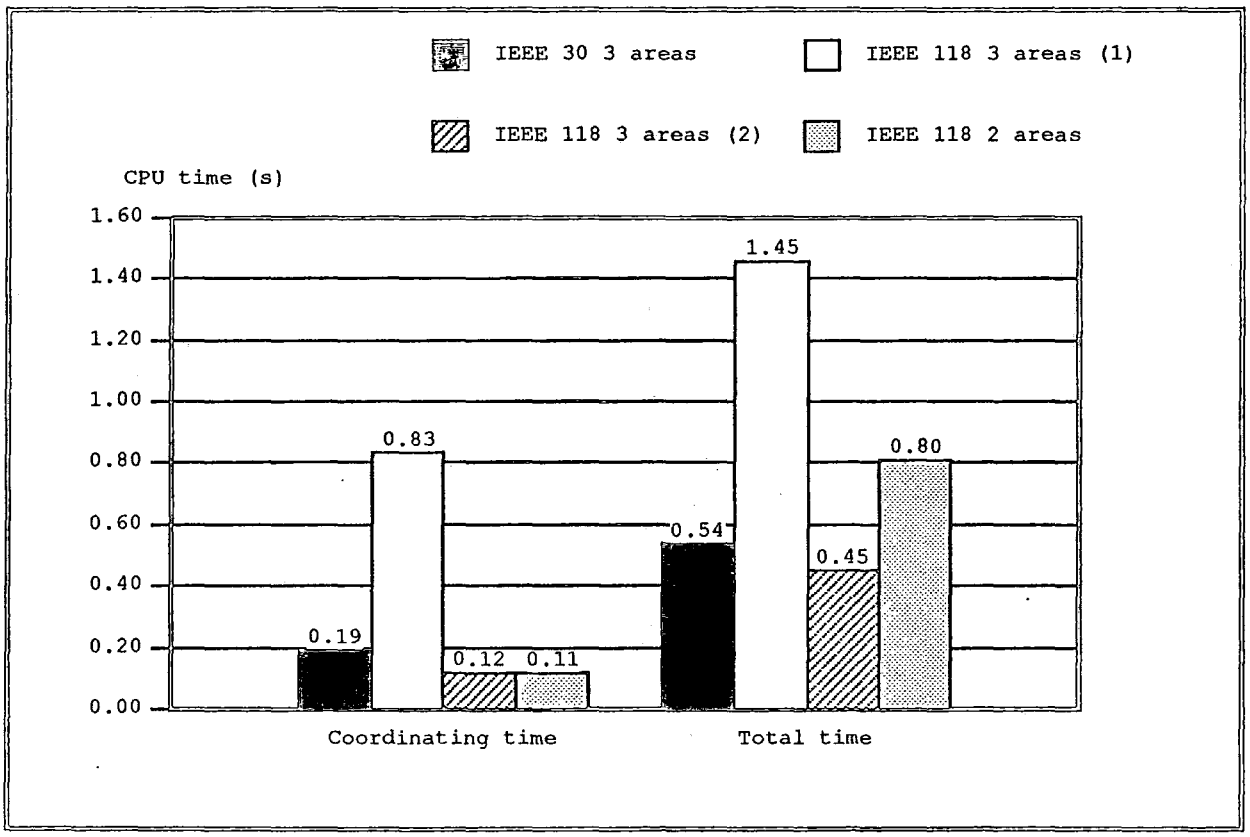


Figure 8.50 Present simulation overall timing

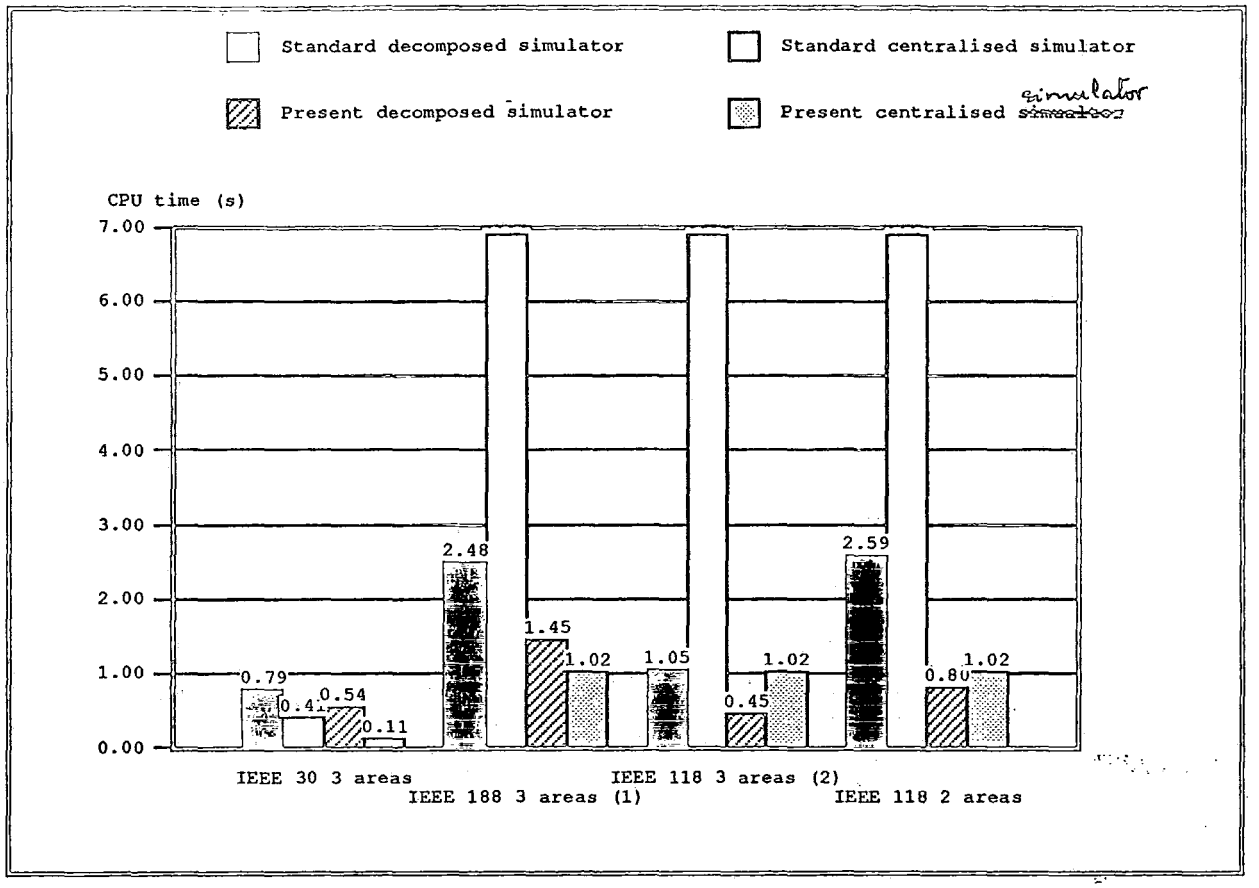


Figure 8.51 Comparison of the decomposed and centralised simulator timings

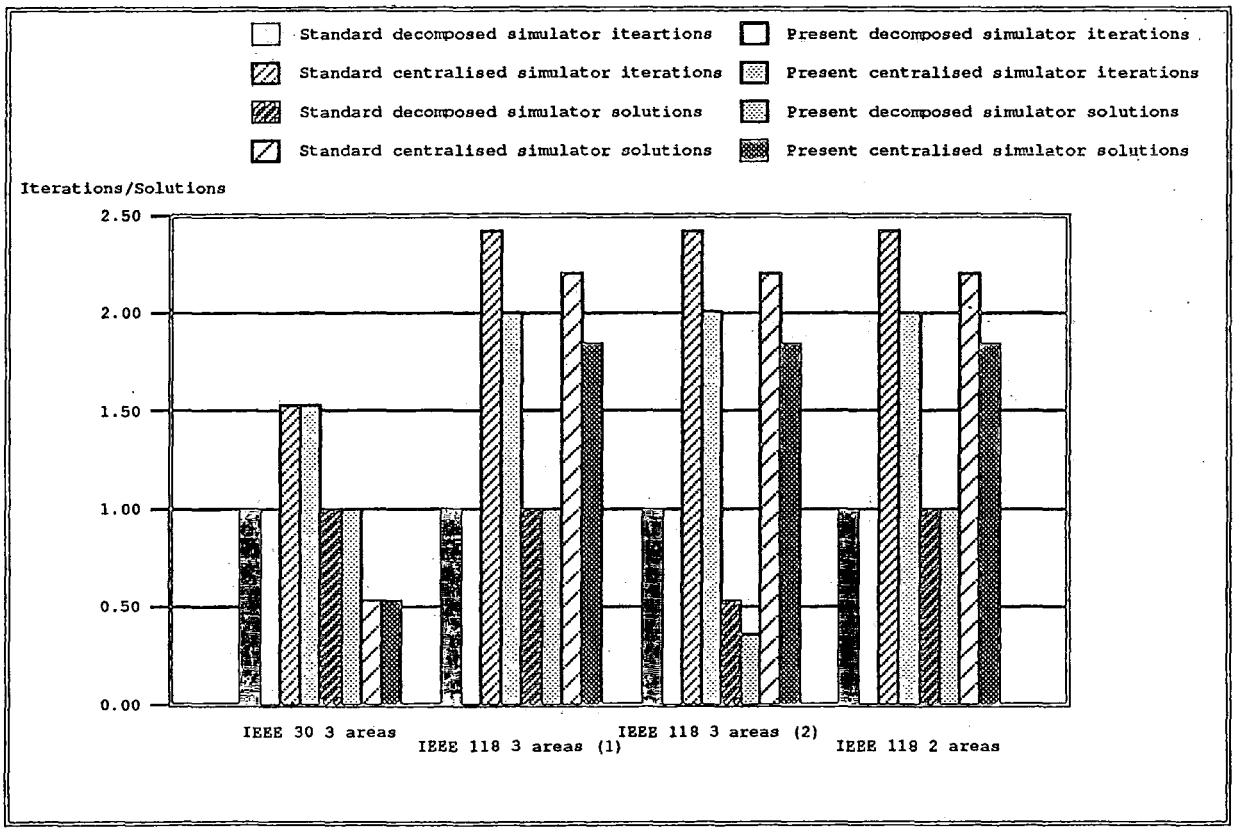


Figure 8.52 Decomposed/centralised simulator no. of iterations/solutions

Network size (area number)	n	N_g	N_l	Number of non zeros	Total number of elements	Sparsity(%)
30 (1)	7	3	9	79	961	8.22
30 (2)	7	1	7	39	255	15.29
30 (3)	16	2	19	90	1024	8.78
first 118 (1)	41	5	57	245	6561	3.73
first 118 (2)	38	8	55	292	10404	2.80
first 118 (3)	39	6	57	261	7569	3.44
second 118 (1)	36	5	49	224	5776	3.87
second 118 (2)	34	7	52	264	8100	3.25
second 118 (3)	48	7	71	316	10816	2.92
118 (1)	70	12	104	494	27556	1.79
118 (2)	48	7	71	316	10816	2.92

Table 8.4 Sparsity of 2 test networks decomposed into 2 and 3 areas using the $2 * 2$ sub-matrix algorithm

This table is built in the same way as table (6.1).

Compared with the results obtained from the centralised simulator as shown in figure (8.51), the newly developed decomposed simulator was slower when the 30 node system was simulated. This inadequacy is mainly due to the decomposition penalties mentioned in chapter 7 and the small size of the areas.

From figure (8.51) it is clear that the Zollenkopf algorithm performs better when the simulated system is large. This is because large systems are generally more sparse than small ones as shown in table (8.4), and the overheads tend to prevail in the CPU time in the case of small systems. For the 30 node system both decomposed simulators are slower than their equivalent centralised simulators. The average timing obtained from applying the new method to the 118 network decomposed into three areas (first case) also was slower than the single processor CPU time. Using the system with less tie lines greatly

improved the speed of the simulator and resulted in an improvement by a factor of more than 2 over the centralised program. Splitting the 118 network into 2 areas improved the execution time of the single processor by a factor of about 1.3.

The performance of the decomposed simulator is expected to decrease in the case of simulating a disturbance, because more solutions than normal are usually involved when a disturbance occurs.

8.7 CONCLUSION

The present simulator is a robust, stable and fast system capable of simulating severe events accurately in real-time. It is well known that the trapezoidal rule is stable for a wide range of time steps. However, to assess the influence of small time steps on the accuracy of the system tests have been carried out. The use of smaller time steps has proved that, with the implementation of control actions from the Energy Management System, the results are accurate enough for the purpose of the present study using a time step of 1 second.

The application of the *bifactorisation* technique to the Array Processor simulator is very encouraging, and provides a cost effective method by significantly increasing the computational speed. The results obtained from exposing the simulator to very harsh conditions without ~~loosing~~ losing stability and the speed performance, suggests that networks at least 4 or 5 times larger than the 118 system can be simulated in real-time when implemented on the parallel simulator running on the Array Processor. Also the present simulated systems may run in real-time even if their models are more comprehensive and complex.

The results obtained from the decomposed, centralised and parallel simulators clearly show that the *bifactorisation* technique is more suitable for large networks because the larger a power system is the more sparse it is, especially if it includes a large number of generators. As far as the Array Processor is concerned the sparsity of the solved system improves the performance of the

simulator, but this is far from fully exploiting the potentiality of this machine (12 Mflops) since the calculation involved in the solution of the set of linear equations (sparse matrices) is of a scalar nature, and Array Processors are conceived to operate at their maximum computational power when applied to vectors.

Theoretically a decomposed method is expected to be n times faster than the single processor solution (n is the number of machines used in parallel). However, this ideal performance is not achieved because of the penalties involved. The timing obtained in these tests can be considered as an estimate performance of the actual decomposed simulator since this algorithm is implemented on a single processor. The overall decomposed timing is calculated by adding the maximum area execution time to the coordinating program CPU time as well as the overheads involved in synchronising the different tasks and transferring data. If the algorithm is applied to a multiprocessor, the overheads are expected to increase since the communication between the slave computers and the master would involve independent memories.

The decomposed simulator results compare favourably with the centralised simulator (except when it was applied to the 30 node system). However, it is not easy to draw clear conclusions about the contribution of each of the speed improving factors. The improvement is partly due to the decomposition and the selection of the tie lines. On the other hand, the numerical stability also has a share in this improvement. For instance, figure (8.51) show a great improvement of the conventional decomposed simulator when compared with the standard centralised simulator results. In fact this improvement is better than the ideal results in the case of the decomposition of the 118 network into three areas (second test). This system has been speeded up by a factor of about 6.57 whereas the maximum speed-up factor is not supposed to exceed the number of areas. The extra speed is mainly due to the improvement of the stability of the Newton-Raphson algorithm.

The *bifactorisation* method compares favourably with the LU decomposition particularly in those cases where the sparsity is greatest (118 node

system and 141 node system) in all cases (parallel, decomposed, and centralised simulators). This is due to the improvement of sparsity of large systems (see tables 6.1). The developed algorithm has not only enormously speeded up the simulator, it has also greatly improved the stability of the system.

CHAPTER 9

CONCLUSION

With the increase of power systems size and complexity, the realisation of inexpensive real-time simulators became a challenge to the available hardware and numerical techniques. To reduce the computational burden at the expense of less accurate results during the few seconds following the disturbance, dynamic simulators have been adopted. Although dynamic simulators are not as comprehensive as some transient stability studies, nevertheless achieving real time or faster than real-time simulation for large systems is not straightforward. Since the bulk of the numerical work in power system simulation is the repeated and frequent solution of very large sparse matrices, investigations into new mathematical techniques and efficient hardware were deemed the most appropriate solution to this otherwise unsolvable problem.

Most of the numerical techniques used for solving the set of linear equations involve four subprograms:

(1) storage of the sparse matrix in a packed form (storage of non zero elements only);
(2) the ordering of the packed matrix;
(3) the factorisation of the ordered matrix; and
(4) the direct solution or evaluation of the unknown variables.

These four tasks can either be programmed separately or grouped in one or more subroutines. For instance, the factorisation stage can be performed with the ordering or with the direct solution. Also parts of these subprograms can be combined together to form a routine.

The consideration of the existing *Harwell* routines which are based on the LU decomposition revealed that some aspects of this algorithm could be neglected in the case of solving power system linear equations. Among these points there is:

(1) the stability test;
(2) ordering the matrix at each iteration; and
(3) moving the pivots for ordering the matrix.

The stability test requires a compromise between the consideration of the cheapest pivotal rows and columns (in terms of generated fill-ins) and the pivot values which should not be less than a given fraction of the largest element in the pivotal column or row. In this case as well as depending on the number of non zero elements, the ordering process depends also on the values of these non zero terms which may be updated at each iteration. Therefore, this scheme, combined with the factorisation process, is often performed at each iteration. It is well known that the ordering process is the most expensive phase in solving large sparse matrices. Since power systems do not require such tests in most cases, the necessity of finding a more economical way of ordering the sparse matrix became apparent.

In this project an alternative sparsity technique has been developed to replace the existing algorithm so that more speed can be achieved and larger networks can be efficiently simulated in real-time at a reasonable accuracy. The algorithm is based on the Zollenkopf *bifactorisation* technique. This method requires the following conditions:

(1) the sparse matrix should have a symmetrical structure;
(2) the diagonal elements should be large enough to avoid numerical instability; and
(3) the inverse of the matrix should exist.

The first condition can be satisfied by storing some zero elements. The second condition is usually satisfied because the diagonal elements of the coef-

ficient matrices related to power system dynamic simulation are often relatively large. However, a low limit is implemented to avoid instability problems. In this case, if the diagonal elements violate this limit, they are forced to take the limit value. Also double precision calculation can help to reduce the occurrence of this problem. To satisfy the third condition care should be taken to avoid the cases where a diagonal element can be null. This problem is also unlikely to happen when simulating power systems.

The characteristics of the Zollenkopf algorithm are:

(1) A semi-optimal ordering scheme is used which does not involve any stability test, thus becoming independent of the values of the non zero elements and depending on the number of non zero terms in each column only. Selecting the column with the least number of non zero elements is the only criteria adopted in performing the ordering to minimise the number of fill-ins. Since the ordering becomes dependent on the structure of the sparse matrix only, and this changes only when a topology variation occurs, it can be performed once for each matrix structure. This characteristic of the Zollenkopf algorithm saves a very large amount of CPU time.

(2) The factorisation is split into a dynamic process which involves the numerical calculation of the reduced and factor matrices, and a static process which involves element indexing. Since the static process is dependent on the structure of the matrix only it is integrated with the ordering scheme and performed once for each network structure.

(3) Storing the diagonal non zero elements separately from the off diagonal elements reduces to a certain extent the indexing involved. The diagonal elements do not need any indexing since they are stored in their natural order. Also indexing the off-diagonal non zero elements once for each matrix structure reduces the required CPU time.

(4) The nature of the power system coefficient matrices suggests the use of the $2 * 2$ sub-matrix algorithm to reduce the integer calculation overhead.

These characteristics together with the nature of the power system dynamic simulation models seemed to provide the answer to the shortcomings of the *Harwell* library routines in this application area.

The efficient *bifactorisation* sparsity technique was investigated and implemented in three types of simulators using three test networks: a parallel simulator running on an Array Processor FPS 5205 hosted by a Perkin Elmer PE 3230 minicomputer, a centralised simulator running on the host computer, and a decomposed simulator running also on the host computer PE 3230 (single machine). The purpose behind using this range of hardware possibilities was to discover the fastest alternative to be used as a dedicated machine for the simulator and highlight the performance of the algorithm which has been developed.

It is well known that Array Processors are very powerful in handling floating-point operations, especially if these are presented in vector form. Also the architecture of these machines exploit parallelism at most levels. It includes three arithmetical units: a floating-point multiplication and addition, and an integer unit which operate in parallel. Each of these units includes IO registers which ensure a quick data input to the arithmetical unit and a fast storage or use of the arithmetical unit outputs. Each of the addition and multiplication floating-point calculation is also performed in parallel by exploiting the pipeline architecture of the Array Processor. These characteristics are very attractive for solving problems which involve a lot of floating-point operations. On the other hand, in order to exploit the features of this machine, a small amount of integer calculation is required.

As well as requiring a large number of floating-point operations, the *bifactorisation* technique used to solve the set of linear equations involves much integer indexing. Also this approach, like all sparsity-oriented methods, involves scalar processing instead of vectors. This shortcoming is alleviated to a small extent by employing the $2 * 2$ sub-matrix strategy. This also helped greatly in reducing the undesirable integer calculation by at most 75 % if all $2 * 2$ sub-matrices are full.

Compared to its counterpart which used the *Harwell* library routines, a great improvement in the speed of the simulator has been achieved by implementing the modified Zollenkopf algorithm in the parallel simulator, especially if the simulated networks are large. Although this achievement is significant, the Array Processor is far from operating at its full potential. This is because the problem fails to conform with the ideal requirements of the Array Processor which has a maximum power of 12 Mgflops when operating on long vectors and matrices.

The Zollenkopf algorithm has also been implemented in the decomposed simulator. Ideally, when compared to a single processor package, the distributed simulator should result in a speed-up factor equal to the number of processors used (number of areas). However, since the decomposition often results in extra calculation to coordinate and synchronise the different processors, this ideal performance is not possible unless other speed-up factors are included. The decomposed simulator can be considered as successful if its overall timing is not dominated by the overheads. Compared to the single processor simulator a large speed-up was obtained for large systems which involve a small number of tie lines when splitting networks into areas. The speed-up has been obtained by combining many factors, among which there are, the application of the modified Zollenkopf method, improved convergence, the decomposition itself and the selection of the tie lines and their number. Therefore, evaluating the contribution of each of these factors towards the improvement of the execution time is not straightforward. The simulation of small decomposed systems is not as efficient as the simulation of large distributed systems. In fact the decomposition of the 30 node network into 3 areas resulted in a slower simulation than the centralised algorithm. This is mainly due to the importance of the overheads which can dominate the execution time for the small areas. This clearly shows that, unlike some of the numerical methods, the *bifactorisation* technique is efficient for all large sparse matrices independently of the type of simulation.

The results obtained from implementing the decomposed simulator on a rather slow minicomputer favour more research in this field using better

hardware. Instead of using a single processor the decomposed simulator, can be extended to an implementation using real parallelism, i.e. use more than one machine.

The centralised version was developed in order to assess the efficiency of the *bifactorisation* technique when compared with the *Harwell* library and isolate this performance effect from the simulator and machines used. By comparing each of the present three types of simulators execution time with their corresponding previous versions, it is clear that the newly developed centralised simulator performs best. This shows that the contribution of the new algorithm is greater than the contribution of the hardware.

The fastest of the three simulators is the one implemented on the Array Processor. This can be attributed to the parallel architecture of this machine and to a lesser extent to the length of its words (38 bits) which are longer than the Perkin Elmer minicomputer words (32 bits).

The objective of this study is to improve the speed of the simulator so that larger networks can be simulated in real-time or faster than real-time. Realising faster than real time power system simulation would allow for the prediction of the effect of a disturbance so that preventive and security measures can be taken in time to avoid the propagation of the disturbance and reduce the damage of such problems.

Achieving the goal of predicting the possible devastating effects of a major disturbance does not only depend on the speed of the simulator. Realistic results are as vital as the CPU time. Fast, stable, and quite accurate methods are required for this purpose. Therefore, the implicit trapezoidal rule was used to algebraise the differential equations, and the Newton-Raphson algorithm was used as an iterative process. These two approaches were chosen for their numerical stability and fast convergence respectively. Tests have been carried out to prove the validity and the accuracy of the results obtained by using different time steps and Newton-Raphson tolerances. These tests showed that, for the time range of interest, the use of a 1 second time step and tolerances of 0.001

(p.u.) and 0.005 (p.u.) was adequate. Although, in certain critical periods of the simulation (very severe disturbances), the Newton-Raphson algorithm may necessitate the use of smaller time steps, usually a quarter of a second.

The numerical stability of the system is also related to certain programming aspects and the correctness of the models. For instance the refinement of the load sub-Jacobian resulted in a better convergence, and therefore, a speed-up of the simulator. Besides the improvement of the speed the system was more stable. In this way a more realistic assessment of the behaviour of the system became possible, and the simulator became more reliable.

Although the stability test of the pivots has not been included in the new *bifactorisation* algorithm, the results obtained (as far as accuracy is concerned) were similar to those of the standard version. Under certain severe conditions the present simulator behaved even better as a result of introducing some of the neglected components and refining some of the models.

9.1 PROPOSALS FOR FUTURE WORK

The Array Processor pipeline and parallel architecture is efficiently exploited when vectors are used. Unfortunately, in our case the majority of the computational effort is spent in solving the sparse matrices whose non zero elements are processed as scalars. The use of the $2*2$ algorithm has alleviated this problem to a certain extent. This algorithm presents the computer with regular patterns which are required by its architecture, however retrieving and processing four elements at each attempt is far from being the ideal solution. Therefore, a flexible algorithm which can generalise the $2*2$ sub-matrix algorithm and identify the optimum sub-matrix size which achieves the best timing, could be developed. The multiplication and inverse calculation of these sub-matrices can be undertaken by the fast Array Processor maths library. However, this may involve large overheads which may end up increasing the actual execution time.

Another possible alternative is to implement the centralised algorithm on machines which are more powerful at handling the scalar and integer calculations.

The decomposed simulator requires further investigation to support all the OCEPS programs and to run under on-line control. Better performance could be achieved by transferring this algorithm to a new multiprocessor computer (e.g. the VAX 6440).

Compromising between the accuracy and the execution time was necessary to achieve real-time simulations for large systems. The models adopted are simplified to take account of mid- and long-term oscillations only. Therefore, if more accurate results are required, further investigations on these models is possible. This is allowed by the flexibility and ease of maintenance of the simulator.

Another promising area is the simulation of the transient stability. Most of the necessary models are well treated and available in the literature. The consideration of the transient stability would certainly result in a more comprehensive simulator which will be able to simulate accurately the behaviour of the power system during transient periods as well as long-term periods. The present simulator already uses a stable numerical method (the implicit trapezoidal rule) which can support a wide range of time steps. multi-step schemes could be used to solve the models involving small time constants more often than the slow models. This strategy may reduce the execution time.

Another aspect of research could be oriented towards developing a black start capability. This may be achieved by starting up one of the stations which can provide the necessary energy for the auxiliary equipment of a larger generating unit. In this way the energy supply to the rest of the system can be built up gradually until the whole system is re-energised. The success of this scheme necessitates stability studies and actual practice in generator black start, line charging voltage control, and so on [108].

This project has concentrated on numerical algorithms for enhancing the performance of the simulator. Enormous speed-up of the simulator has been achieved especially where the sparsity is the greatest. For instance, in the simulation of the 118 network on the parallel simulator a speed-up factor of about 15 times has been achieved for the overall timing, and the Array Processor calculation was about 20 times faster than the performance of the Array Processor version of the previous algorithm.

The present project has not only improved the speed of the simulator, but has also provided a great improvement in the stability of the solution. The development of the models by including fast components and the ability to starting the simulator from a blackout remain to be continued.

REFERENCES AND BIBLIOGRAPHY

1. ALVARADO, F.L., YU, D.C., BETANCOURT, R., "Partitioned sparse A^{-1} methods.", *IEEE Trans. on Power Systems*, May 1990, No. 2, Vol. 5, pp. 452 - 459.

2. ANDERSON, P.M., "Advanced computer concepts for large scale system analysis." *Advanced Stability Seminar, New York.* , Dec. 1979.

3. ANDERSON, P.M., "Mathematical modeling of boilers for dynamic stability simulation.", *Proceedings of power plant dynamics control and testing symposium, USA*, Feb. 1973, pp. 7/2 - 7/17.

4. ARRILLAGA, J., ARNOLD, C.P., HARKER, B.J., "Computer modelling of electrical power systems.", John Wiley & sons, 1983.

5. BARGIELA, A., IRVING, M.R., STERLING, M.J.H., "Observability determination in power system state estimation using a network flow technique.", *IEEE Trans. PWRS*, May 1986, No. 2, Vol. 1, pp. 108 - 114.

6. BERG, G.J., "Power system load representation.", *IEE Proceedings*, Mar. 1973, No. 3, Vol. 120, pp. 344 - 348.

7. BIRCH, A.P., "Adaptive load frequency control of electrical power systems.", Ph.D. Thesis, University of Durham, 1988.

8. BJÖRCK, A., "Methods for sparse linear least squares problems." , *in Bunch, J.R., Rose, D.J., (Ed.): 'Sparse Matrix Computations'*, Academic press, 1976, pp. 177 - 199.

9. BORSE, G.J., "FORTRAN 77 and numerical methods for engineers.", PWS engineering, Boston, 1985.

10. BRAMELLER, A., JOHN, M.N., SCOTT, M.R., "Practical diakoptics for electrical networks", Chapman & Hall, 1969.

11. BRAMELLER, A.,ALLAN, R.N., HAMAM, Y.M., "Sparsity.", Pitman, 1976.

12. BRASCH, F.M.JR., VAN NESS, J.E., KAUG, S.C., "Simulation of a multiprocessor network for power system problems.", *IEEE PAS*, Feb. 1982, No. 2, Vol. 101, pp. 295 - 301.

13. BURGUIN, P., "Distributed real time computing and simulation", *IEE Colloquium on system design and training aspects of simulators and real time simulation, UK*, May 1987.

14. CAFARO, G., PUGLIESE, P., VACCA, F., "Parallel solution of torn networks.", *Electrical Power & Energy Systems*, July 1984, No. 3, Vol. 6 , pp. 131 - 138.

15. CATE, E.G., HEMMAPLARDH, K., MANKE,J.W., et al., "Time frame notion and time response of the models in transient, mid-term and long-term stability programs.", *IEEE Trans. PAS*, Jan. 1984, No. 2, Vol. 103, pp. 143 - 151.

16. CHENG, D.T.Y., "Fast decoupled load flow with transformer representations and branch outages.", Internal research report, University of Durham, June 1983.

17. CONCORDIA, C., IHARA, S., "Load representation in power system stability studies.", *IEEE Trans. PAS*, Apr. 1982, No. 4, Vol. 101, pp. 969 - 977.

18. DANDENO, P.L., KUNDUR, P., "A non iterative transient stability program including the effects of variable load voltage characteristics.", *IEEE Trans. PAS*, Sep. 1973, No. 5, Vol. 92, pp. 1478 - 1484.

19. DAVIDSON, D.R., EWART, D.N., KICHMAYER, L.K., "Long-term dynamic response of power systems: An analysis of major disturbances.", *IEEE Trans. PAS*, May 1975, No. 3, Vol. 94, pp. 819 - 826.

20. DE MELLO, F.P., MILLS, R.J., B'RELLS, W.F., "Automatic generation control. Part I and II.", *IEEE Trans. PAS*, Mar. 1973, No. 2, Vol. 92, pp. 710 - 724.

21. DETIG, D.M., "Effects of special purpose hardware in scientific computation with emphasis on power system applications.", *IEEE Trans. PAS*, Feb. 1982, No. 2, Vol. 101, pp. 265 - 270.

22. DINELEY, J.L., FENWICK, P.J., "The effect of prime-mover and excitation control on the stability of large steam turbo-generator", *IEEE Trans. PAS*, Sep. 1974, No. 5, Vol. 93, pp. 1613 - 1623.

23. DODSON, D.S., "A study of sparse matrix algorithms applicable to electric power flow problems.", *Proceedings of the 1981 Array Conference* , Floating-Point Systems user society USA, 1981.

24. DOMMEL, H.W., SATO, N., "Fast transient stability solutions.", *IEEE Trans. PAS*, July 1972, No. 4, Vol. 91, pp. 1643 - 1650.

25. DUFF, I.S., "A survey of sparse matrix research.", *IEEE proceedings*, 1976, pp. 500 - 535.

26. DUGAN, R.C., DURHAM, I., TALUKDAR, S.N., "An algorithm for power system simulation by parallel processing.", *IEEE Engineering Society Summer Meeting*, July 1979.

27. DUNLOP, R.D., EWART, D.N., SCHULS, R.P., "Use of digital computer simulations to assess long-term power system dynamic response.", *IEEE Trans. PAS*, May 1975, No. 3, Vol. 94, pp. 850 - 857.

28. DURHAM, I., DUGAN, R.C., JONES, A.K., et al., "Power system simulation on a multiprocessor.", *IEEE Power Engineering Society Summer Meeting*, 1979, pp. 1 - 6.

29. DWARAKANATH, M.H., DEMBART, B., ERISMAN, A.M., et al., "A general methodology for modelling system components in power system dynamic simulation.", *IEEE Trans. PAS*, Jan. 1982, No. 1, Vol. 101, pp. 136 - 146.

30. DY LIACCO, T.E., "The role and implementation of state estimation in an energy management system.", *Electrical Power & Energy Systems*, Apr. 1990, No. 2, Vol. 12, pp. 75 - 79.

31. DY LIACCO, T.E., ENNS, M.K., SCHOEFFLER, J.D., et al., "Considerations in developing and ~~training~~ utilizing operator training simulators.", *IEEE Trans. PAS*, Nov. 1983, No. 11, Vol. 102, pp. 3672 - 3679.

32. ELDER, L., METCALFE, M.J., "An efficient method for real-time simulation of large power system disturbances.", *IEEE Trans. PAS*, Feb. 1982, No. 2, Vol. 101, pp. 334 - 339.

33. ELECTRICITY COUNCIL (ED.), "Power system protection. Vol. 2: Systems and models", Peter Peregrinus Ltd., 1981.

34. ELECTRICITY COUNCIL (ED.), "Power system protection. Vol. 3: Application", Peter Peregrinus Ltd., 1981.

35. EL-MARSATAWY, M.M., MENZIES, R.W., MATHUR, R.M., "A new, exact, diakoptic, fast-decoupled load-flow technique for very large power systems.", *IEEE Trans. PAS*, July 1979, Vol. 98, pp. 1 - 7.

36. EL-SHERBINY, M.K., MELITA, D.M., "Dynamic system stability. Part I: Investigation of the effect of different loading and excitation systems.", *IEEE Trans. PAS*, Oct. 1973, No. 5, Vol. 92, pp. 1538 - 1546.

37. ENNS, K.,M., TINNEY, W.F., "Sparse matrix inverse factors", *IEEE Trans. on Power systems*, May 1990, No. 2, Vol. 5, pp. 466 - 471.

38. EPRI, "Long-term power system dynamic. Phase (III): Analysis capability to simulate disturbances.", *General Electric Company Final Report*, EPRI report EL-983 project 764-2, Vol. 1, May 1982.

39. EPRI, "Optimisation of reactive Volt-ampere (VAR) sources in system planning.", *Vol. 1 Solution techniques, computing methods and results*, EPRI report EL-3729 project 2109-1, Nov. 1984.

40. FERRIS, L.L, SASSON, A.M., " Investigation of the load-flow problem.", *IEE proceedings*, Oct. 1968, No. 10, Vol. 115, pp. 1459 - 1470.

41. FLAXMAN, J.W., " Multi-processor power system simulation.", Ph.D. Thesis, Electrical Engineering Department, University of Durham, 1987.

42. FLOATING POINT SYSTEMS, "FPS 500 manual: FPS-500 Processor handbook", publication No. 860-7437-038A, Aug. 1983.

43. FONG, J., POTTLE, C., "Parallel processing of power system analysis problems via simple parallel microcomputer structures.", *IEEE Trans. PAS*, Sep. 1978, No. 5, Vol. 97, pp. 1834 - 1841.

44. FROWD, R.J., PODMORE, R., GIRI, J.C., "Transient stability and long-term dynamics unified.", *IEEE PES Winter Meeting*, Jan. 1982, pp. 1 - 9.

45. FULLER, S.H., OUSTERHOUT, J.K., RASKIN, L., et al., "Multi-processors: An overview and working example.", *IEEE Proceedings*, Feb. 1978, No. 2, Vol. 66, pp. 216 - 228.

46. GEAR, C.W., "Numerical initial value problems in ordinary differential equations.", Prentice hall, 1971.

47. GEORGE, A., LIU, J.W., "Computer solution of large sparse positive definite systems.", Prentice hall, 1981.

48. GIROTTI, T.B., TWEED, N.B., HOUSER, N.R., et al., "Real-time VAR control by SCADA", *IEEE Trans. on Power Systems*, Feb. 1990, No. 1, Vol. 5, pp. 61 - 63.

49. GROSS, G., BERGEN, A.R., "A calss of new multistep integration algorithms for the computation of power system dynamical response.", *IEEE Trans. PAS*, Jan. 1977, No. 1, Vol. 96, pp. 293 - 306.

50. GUILE, A.E., PATERSON, W., "Electrical power systems", Vol. 1, Oliver Boyd, 1969.

51. HAPP, H.H., "Piecewise methods and applications to power systems ", John Wiley & sons, New York, 1980.

52. HAPP, H.H., POTTLE, C., WIRGAU, K.A., "Future computer technology for large power system simulation.", *IFAC*, Aug. 1979, Vol. 15, pp. 621 - 629.

53. HAPP, H.H., YOUNG, C.C., "Tearing algorithms for large-scale network programs", *PICA conference*, June 1971, pp. 2639 - 2649.

54. HATCHER, W.L., BRASCH, F.M., VAN NESS, J.E., "A feasibility study for the solution of transient stability problems by multiprocessor structures.", *IEEE Trans. PAS*, Nov. 1977, No. 6, Vol. 96, pp. 1789 - 1797.

55. HEMMAPLARDH, K., LAMONT, J.W., "Consideration of a long-term dynamic simulation program.", *IEEE Trans. PWRS*, Feb. 1986, No. 1, Vol. 1, pp. 129 - 135.

56. HESTENES, M.R., STIEFEL, S., "Methods of conjugate gradients for solving linear systems", *Journal of research national bureau standards section B*, 1952, Pt. B, Vol. 49, pp. 409 - 436.

57. HEYDT, C.T., "Computer analysis methods for power systems.", Macmillan, USA, 1986.

58. HOCKNEY, R. W., JESSHOPE, C.R., "Parallel computers", Adam Hilger Ltd., UK, 1984

59. HON, T.K., "An interactive load shedding technique for electric systems control", Internal research report, University of Durham, 1986.

60. HOUSEHOLDER, A.S., "Principles of numerical analysis", McGraw-Hill, New York, 1953.

61. HWANG, K., "Advanced parallel processing with supercomputer architectures.", *Proceedings of the IEEE*, Oct. 1987, No. 10, Vol. 75, pp. 1348 - 1380.

62. IEEE COMMITTEE REPORT., "Computer representation of excitation systems.", *IEEE Trans. PAS*, June 1968, No. 6, Vol. 87, pp. 1460 - 1464.

63. IEEE COMMITTEE REPORT., "Dynamic models for steam and hydro turbines in power system studies.", *IEEE Trans. PAS*, Nov. 1973, No. 6, Vol. 92, pp. 1904 - 1915.

64. IEEE WORKING GROUP, "MW response of fossil fueled steam units.", *IEEE Trans. PAS*, Mar. 1973, No. 2, Vol. 92, pp. 455 - 463.

65. ILICETO, F., CEYHAN, A., RUCKSTUHL, G., "Behavior of loads during voltage dips encountered in stability studies. Field and laboratory tests.", *IEEE Trans. PAS*, Nov. 1972, No. 6, Vol. 91, pp. 2470 - 2479.

66. IRVING, M.R., STERLING, M.J.H., "Efficient Newton-Raphson algorithm for load-flow calculation in transmission and distribution networks. ", *IEE proceedings*, Sep. 1987, Pt. C, No. 5, Vol. 134, pp. 325 - 328.

67. IRVING, M.R., STERLING, M.J.H., "Optimal network tearing using simulated annealing.", *IEE Proceedings*, Jan. 1990, Pt. C, No. 1, Vol. 137, pp. 69 - 72.

68. IRVING, M.R., STERLING, M.J.H., "Power system simulation pilot study.", Internal CEGB report, University of Durham, Apr. 1988.

69. JEANBART, C., LOGEAY, Y., MUSART, M., "E.D.F. simulator for control centre operations.", *Cigré International conference on large high voltage electric systems*, Aug. 1988.

70. JERVIS, W.B., SCOTT, J.G.P., GRIFFITHS, H., "Future application of reactive compensation plant on the CEGB system to improve transmission network capability." *Cigré International conference on large high voltage electric systems, Paris*, Aug. 1988.

71. JOETTEN, R., WESS, T., WALTERS, J., "A new real-time simulator for power system studies.", *IEEE Trans. PAS*, Sep. 1985, No. 9, Vol. 104, pp. 2604 - 2611.

72. JOHNSON, R.B.I., CORY, B.J., SHORT, M.J., "A tunable integration method for the simulation.", *IEEE Trans. PWRS*, Nov. 1988, No. 4, Vol. 3, pp. 1530 - 1537.

73. JOHNSON, R.B.I, SHORT, M.J., CORY, B.J., "Improved simulation techniques for power system dynamics", *IEEE Trans. PWRS*, Nov. 1988, No. 4, Vol. 3, pp. 1691 - 1697.

74. JONES, A.K., CHANSLER, R.J., DURHAM, JR.I., "Programming issues raised by a multiprocessor.", *IEEE Proceedings*, Feb. 1978, No. 2, Vol. 66, pp. 229 - 237.

75. KEYHANI, A., "Development of an interactive power system research simulator.", *IEEE Trans. PAS*, Mar. 1984, No. 3, Vol. 103, pp. 516 - 521.

76. KIMBARK, E.W., "Power system stability: synchronous machines.", John Wiley & sons, inc., 1968.

77. LEY, J.B., "Computer aided analysis and design for electrical engineers.", Holt, Rinchart, and Winston Inc., 1970.

78. LLIEV, S., "Method for dynamic behaviour control of electric power systems.", *Electrical Power & Energy Systems.*, Oct. 1984, No. 4, Vol. 6, pp. 212 - 220.

79. LO, C.H., ANDERSON, M.D., RICHARDS, E.F., "An interactive power system analyser with graphics for educational use.", *IEEE Trans. PAS*, May 1986, No. 2, Vol. 1, pp. 174 - 181.

80. LOPEZ LOPEZ, A., "Dynamic simulation of power systems on multiple microprocessors.", Ph.D. Thesis, Electrical Engineering Department, Imperial college of science and technology, University of London, Nov. 1983.

81. MESCUA, J., "A decoupled method for systematic adjustments of phase-shifting and tap-changing transformers.", *IEEE Trans. PAS* , Sep. 1985, No. 9, Vol. 104, pp. 2315 - 2321.

82. NORIN, R.S., "Sparse matrix operations with AP-120B/190L Array Processors.", *Proceedings of the 1981 Array Conference*, Floating-Point Systems user society USA, 1981.

83. OHYAMA, T., WATANABE, A., NISHIMURA, K., et al., "Voltage dependence of composite loads in power systems.", *IEEE Trans. PAS*, Nov. 1985, No. 11, Vol. 104, pp. 3064 - 3073.

84. OREM, F.M., "A big program in a little box.", *Proceedings of the 1981 Array Conference*, Floating-Point Systems user society USA, 1981.

85. OTT, G.E., WALKER, L.N., WONG, D.T.Y., "Hybrid simulation for long-term dynamics.", *IEEE Trans. PAS*, June 1977, No. 3, Vol. 96, pp. 907 - 915.

86. PERKIN ELMER, "FORTRAN run time library real time extensions reference manual.", Interdata division, 1978.

87. PHADKE, A.G., THORP, J.S., HOROWITZ, S.H., "Impact of adaptive protection on power system control.", *Proceedings of the 9th PSCC, Cascais, Portugal*, Aug. 1987, pp. 283 - 290.

88. PRICE, W.W., WIRGAU, K.A., MURDOCH, A., et al., " Load modeling for power flow and transient stability computer studies.", *IEEE Trans. on power systems*, Feb. 1988, No. 1, Vol. 3, pp. 180 - 187.

89. RAFIAN, M., IRVING, M.R., STERLING, M.J.H., "A real-time power system Simulation.", *IEE second international conference on simulators, UK*, Sep. 1987, pp. 219 - 223.

90. RAFIAN, M., STERLING, M.J.H., IRVING, M.R., "Application of an Array Processor to real time power system simulation.", *ASME International Conference on Modelling and Simulation, Sorrento, Italy*, 1986, pp. 1 - 16.

91. RAFIAN, M., STERLING, M.J.H., IRVING, M.R., "Decomposed load-flow algorithm suitable for parallel processor implementation.", *IEE Proceedings*, Nov. 1985, Pt. C, No. 6, Vol. 132, pp. 281 - 284.

92. RAFIAN, M., STERLING, M.J.H., IRVING, M.R., "Parallel processing algorithm for power system simulation.", *IEE Proceedings*, July 1988, Pt. C, No. 4, Vol. 135, pp. 285 - 290.

93. RAFIAN, M., STERLING, M.J.H., IRVING, M.R., "Real-time power system simulation.", *IEE Proceedings*, May 1987, Pt. C, No. 3, Vol. 134, pp. 206 - 223.

94. RAINBOLT, S.M., LO, E.O., VIRMANI, S., "Implementation of electric power transient stability software for a VAX 11/780 and AP-120B.", *Proceedings of the 1981 Array Conference*, Floating-Point Systems user society USA, 1981.

95. RIBIERO, J.R., LANGE, F.J., "A new aggregation method for determining composite load characteristics.", *IEEE Trans. PAS*, Aug. 1982, No. 8, Vol. 101, pp. 2869 - 2875.

96. RIEDER, W.G., BUSBY, H.R., "Introductory engineering modeling emphasizing differential models and computer simulations. ",John Wiley & sons, 1986.

97. SAIKAWA, K., GOTO, M., IMAMURA, Y., et al., "Real-time simulation system of large scale power system dynamics for a dispatcher training simulator.", *IEEE Trans. PAS*, Dec. 1984, No. 12, Vol. 103, pp. 3496 - 3501.

98. SCHULZ, R.P., JONES, W.D., EWART, D.N., "Dynamic models of turbine generators derived from solid rotor equivalent circuits.", *IEEE Trans. PAS*, May 1973, No. 3, Vol. 92, pp. 926 - 933.

99. SHACKSHAFT, G., SYMONS, O.C., HADWICK, J.G., "General purpose model of power system loads.", *IEE Proceedings*, 1977, No. 8, Vol. 124, pp. 715 - 723.

100. SHOULTS, R.R., CHEN, M.S., DOMIJAN, A. JR., "Electric power system simulation laboratory and energy management system control center.", *IEEE Trans. PWRS*, Feb. 1987, No. 1, Vol. 2, pp. 293 - 246.

101. STAGG, G.W., EL-ABIAD, A.H., "Computer methods in power system analysis.", McGraw-Hill, New York, 1968.

102. STERLING, M.J.H., "Power system control.", Peter peregrims ltd., 1978.

103. STERLING, M.J.H., IRVING, M.R., "Performance of analysis and control software under network islanding and other emergency conditions.", *Proceedings of the 9th PSCC, Portugal*, Aug. 1987, pp. 860 - 866.

104. STERLING, M.J.H., IRVING, M.R., "Real-time simulation for the verification of advanced on-line control algorithms.", *IEE international conference on power systems monitoring and control, UK*, July 1986, pp. 265 - 269.

105. STERLING, M.J.H., SWIFT, J.S., IRVING, M.R., "Power system simulation developments at the University of Durham.", *IEE Colloquium on Power system simulation*, May 1989.

106. STOTT, B., "Power system dynamic response calculations.", *IEEE Proceedings*, Feb. 1979, No. 2, Vol. 67, pp. 219 - 241.

107. STOTT, B., ALSAÇ, O., "Fast decoupled load flow", *IEEE Trans. PAS*, May 1974, No. 3, Vol. 93, pp. 859 - 869.

108. STUBBE, M., BIHAIN, A., DEUSE, J., et al., "STAG- a new unified software program for the study of the dynamic behavior of electrical power systems.", *IEEE Trans. on power systems*, Feb. ~~1975~~ 1989, No. 1, Vol. 4, pp. 129 - 138.

109. SUSUMAGO, I., SUZUKI, M., MIYANA, K., et al., "Development of a large-scale dispatcher training simulator and training results.", *IEEE Trans. PWRS*, May 1986, No. 2, Vol. 1, pp. 67 - 75.

110. TAOKA, H., ABE, S., TAKEBA, S., "Fast transient stability solution using an Array Processor.", *IEEE Trans. PAS*, Dec. 1983, No. 12, Vol. 102, pp. 3835 - 3841.

111. TINNEY, W.F., WALKER, J.W., "Direct solutions of sparse network equation by optimally ordered triangular factorisation.", *IEEE proceedings*, Nov. 1967, No. 11, Vol. 55, pp. 1801 - 1809.

112. WALKER, L.N., STITH, J.V., GRAHAM, H.,"Dispatch operator training program utilizing the hybrid simulator facility.", *IEEE Trans. PAS*, Sep. 1982, No. 9, Vol. 101, pp. 3342 - 3348.

113. WOO, P.T., "Application of an Array Processor to sparse elimination." *Proceedings of the 1981 Array Conference*, Floating-Point Systems user society USA, 1981. ~~elimination."~~

114. WOOD, A.J., WOLLENBERG, B.F., "Power generation and control.", John Wiley & sons Inc., 1984.

115. ZHANG, G., BOSE, A., "Scenario building for operator training simulator using a transient stability program", *IEEE Trans. PWRS*, Jan. 1984, No. 1, Vol. 6, pp. 44 - 50.

116. ZHANG, B., NIANDE, X., SHYING, W., "Unified piecewise solution of power system networks combining both branch cutting and node tearing.", *Electrical power & energy systems*, Oct. 1989, No., 4, vol. 11, pp. 283 - 288.

117. ZOLLENKOPF, K., "Bi-factorization basic computational algorithm and programming techniques.", *in REID, J.K. (Ed.): 'Large Sparse Sets of Linear Equations.'*,

118. KRON, G., "Tensor analysis of networks.", John Wiley & sons, New York, 1939

APPENDIX A

A.1 DYNAMIC SUB-JACOBIAN

The algebraisation of the dynamic models related to the synchronous generator and its local controllers are included in this section. The Jacobian elements of this subset are derived from these functions with respect to the dynamic state variables.

A.1.1 Swing equation

Replacing P_m and P_g by their expressions from equation (4.33) and (4.15 (a)) respectively into equation (4.17) gives

$$\frac{d\omega}{dt} = K\{\frac{R_g}{|Z_g|^2}|V_g|^2 - D(\omega - \omega_a) + P_{hp}F_{hp} + P_{ip}F_{ip} + P_{lp}F_{lp} - \frac{|E_g|}{|Z_g|^2}[V_{ri}(X_d * \sin(\delta) + R_g * \cos(\delta)) + V_{imi}(R_g * \sin(\delta) - X_d * \cos(\delta))]\} \quad (A.1)$$

Where $K = (\frac{\pi * f_o}{H})$

Applying the trapezoidal rule to this equation will result in

$$\omega_t = \omega_{(t-\Delta t)} + \frac{K\Delta t}{2}\{\frac{R_g}{|Z_g|^2}({V_{ri}}^2 + {V_{imi}}^2) - D(\omega - \omega_a) + P_{hp}F_{hp} + P_{ip}F_{ip} + P_{lp}F_{lp} - \frac{|E_g|}{|Z_g|^2}[V_{ri}(X_d \sin(\delta) + R_g \cos(\delta)) + V_{imi}(R_g \sin(\delta) - X_d \cos(\delta))]\}_t + \frac{K\Delta t}{2}\{\frac{R_g}{|Z_g|^2}({V_{ri}}^2 + {V_{imi}}^2) - D(\omega - \omega_a) + P_{hp}F_{hp} + P_{ip}F_{ip} + P_{lp}F_{lp} - \frac{|E_g|}{|Z_g|^2}[V_{ri}(X_d \sin(\delta) + R_g \cos(\delta)) + V_{imi}(R_g \sin(\delta) - X_d \cos(\delta))]\}_{(t-\Delta t)} \quad (A.2)$$

This algebraised equation has the same form as equation (4.91) with the function h at time steps t and $(t-\Delta t)$ given by

$$h(\omega,\delta,|E_g|,P_{hp},P_{ip},P_{lp},V_{ri},V_{imi})_t = \omega_t - \frac{K\Delta t}{2}\{\frac{R_g}{|Z_g|^2}({V_{ri}}^2+{V_{imi}}^2) - D(\omega-\omega_a) + P_{hp}F_{hp} + P_{ip}F_{ip} + P_{lp}F_{lp} - \frac{|E_g|}{|Z_g|^2}[V_{ri}(X_d\sin(\delta)+R_g\cos(\delta)) + V_{imi}(R_g\sin(\delta)-X_d\cos(\delta))]\}_t \quad (A.3)$$

$$h(\omega,\delta,|E_g|,P_{hp},P_{ip},P_{lp},V_{ri},V_{imi})_{(t-\Delta t)} = \omega_{(t-\Delta t)} + \frac{K\Delta t}{2}\{\frac{R_g}{|Z_g|^2}({V_{ri}}^2+{V_{imi}}^2) - D(\omega-\omega_a) + P_{hp}F_{hp} + P_{ip}F_{ip} + P_{lp}F_{lp} - \frac{|E_g|}{|Z_g|^2}[V_{ri}(X_d\sin(\delta)+R_g\cos(\delta)) + V_{imi}(R_g\sin(\delta)-X_d\cos(\delta))]\}_{(t-\Delta t)} \quad (A.4)$$

$$C = DK\omega_a\Delta t \quad (A.5)$$

For simplicity and to conform with equation (4.104) notation, the function of equation (A.3) is renamed h_1. This can be rewritten as follows

$$h_1 = \omega_t(1+\frac{DK\Delta t}{2}) - \frac{K\Delta t}{2}\{\frac{R_g}{|Z_g|^2}({V_{ri}}^2+{V_{imi}}^2) + P_{hp}F_{hp} + P_{ip}F_{ip} + P_{lp}F_{lp} - \frac{|E_g|}{|Z_g|^2}[V_{ri}(X_d\sin(\delta)+R_g\cos(\delta)) + V_{imi}(R_g\sin(\delta)-X_d\cos(\delta))]\}_t \quad (A.6)$$

The derivatives of this function with respect to the related dynamic state variables are given below

$$\frac{d(h_1)}{d\omega} = (1+\frac{DK\Delta t}{2}) \quad (A.7)$$

$$\frac{d(h_1)}{d\delta} = \frac{K\Delta t}{2}\frac{|E_g|}{|Z_g|^2}[V_{ri}(X_d\cos(\delta)-R_g\sin(\delta))+V_{imi}(R_g\cos(\delta)+X_d\sin(\delta))] \quad (A.8)$$

$$\frac{d(h_1)}{d|E_g|} = \frac{K\Delta t}{2}\frac{1}{|Z_g|^2}[V_{ri}(X_d\sin(\delta)+R_g\cos(\delta))+V_{imi}(R_g\sin(\delta)-X_d\cos(\delta))] \quad (A.9)$$

$$\frac{d(h_1)}{dP_{hp}} = -\frac{K\Delta t}{2}F_{hp} \quad (A.10)$$

$$\frac{d(h_1)}{dP_{ip}} = -\frac{K\Delta t}{2}F_{ip} \quad (A.11)$$

$$\frac{d(h_1)}{dP_{lp}} = -\frac{K\Delta t}{2}F_{lp} \quad (A.12)$$

A.1.2 Rotor angle equation

Algebraising equation (4.20) will result in

$$\delta_t = \delta_{(t-\Delta t)} + \frac{\Delta t}{2}(\omega_t - \omega_o + \omega_{(t-\Delta t)} - \omega_o) \tag{A.13}$$

$$h(\omega,\delta)_t = \delta_t - \frac{\Delta t}{2}\omega_t \tag{A.14}$$

$$h(\omega,\delta)_{(t-\Delta t)} = \delta_{(t-\Delta t)} + \frac{\Delta t}{2}\omega_{(t-\Delta t)} \tag{A.15}$$

$$C = -\omega_o \Delta t \tag{A.16}$$

If we rename the function $h(\omega,\delta)_t$ as h_2 and obtain its derivative with respect to ω and δ we obtain the following two Jacobian elements

$$\frac{d(h_2)}{d\omega} = -\frac{\Delta t}{2} \tag{A.17}$$

$$\frac{d(h_2)}{d\delta} = 1 \tag{A.18}$$

A.1.3 The AVR equations

Replacing $|V|_g$ by its components in equation (4.24) gives

$$\frac{d|E_g|}{dt} = \frac{K_a[V_{ref} - (V_{ri}{}^2 + V_{imi}{}^2)^{1/2}] - |E_g|}{T'_{do}} \tag{A.19}$$

The algebraisation of this equation is as follows

$$\begin{aligned} |E_g|_t =& |E_g|_{(t-\Delta t)} + \frac{\Delta t}{2T'_{do}}\{K_a[V_{ref} - (V_{ri}{}^2 + V_{imi}{}^2)^{1/2}]_t - (|E_g|)_t \\ &+ K_a[V_{ref} - (V_{ri}{}^2 + V_{imi}{}^2)^{1/2}]_{(t-\Delta t)} - (|E_g|)_{(t-\Delta t)}\} \end{aligned} \tag{A.20}$$

This equation can be rewritten as follows

$$\begin{aligned} &|E_g|_t(1 + \frac{\Delta t}{2T'_{do}}) + (V_{ri}{}^2 + V_{imi}{}^2)_t{}^{1/2}(\frac{K_a \Delta t}{2T'_{do}}) = \\ &|E_g|_{(t-\Delta t)}(1 - \frac{\Delta t}{2T'_{do}}) - (\frac{K_a \Delta t}{2T'_{do}})(V_{ri}{}^2 + V_{imi}{}^2)^{1/2}{}_{(t-\Delta t)} + \frac{K_a \Delta t}{T'_{do}} V_{ref} \end{aligned} \tag{A.21}$$

The general function h at time steps t and $(t-\Delta t)$ of this equation is

$$h(|E_g|, V_{ri}, V_{imi})_t = |E_g|_t(1 + \frac{\Delta t}{2T'_{do}}) + (\frac{K_a \Delta t}{2T'_{do}})(V_{ri}{}^2 + V_{imi}{}^2)_t{}^{1/2} \tag{A.22}$$

$$h(|E_g|, V_{ri}, V_{imi})_{(t-\Delta t)} = |E_g|_{(t-\Delta t)}(1 - \frac{\Delta t}{2T'_{do}}) - (\frac{K_a \Delta t}{2T'_{do}})(V_{ri}{}^2 + V_{imi}{}^2)^{1/2}_{(t-\Delta t)} \tag{A.23}$$

$$C = \frac{K_a \Delta t}{T'_{do}} V_{ref} \tag{A.24}$$

For simplicity and to conform with the order in which the Jacobian matrix is built the function h of equation (A.22) is renamed h_3. The only dynamic Jacobian element related to this function is

$$\frac{d(h_3)}{d(|E_g|)} = (1 + \frac{\Delta t}{2T'_{do}}) \tag{A.24}$$

A.1.4 Governor model

Replacing Δf by $\frac{(\omega - \omega_{set})}{2\pi}$ into equation (4.30), and replacing the equation obtained into (4.31) the following differential equation can be obtained

$$\frac{dP_{gv}}{dt} = \frac{P_{set} - \frac{(\omega - \omega_{set})}{2\pi R} - P_{gv}}{T_c} \tag{A.25}$$

The algebraisation of this equation is as follows

$$\begin{aligned} P_{gv_t} =& P_{gv_{(t-\Delta t)}} + \frac{\Delta t}{2T_c}\{P_{set} - \frac{(\omega - \omega_{set})}{2\pi R} - P_{gv}\}_t \\ &+ \frac{\Delta t}{2T_c}\{P_{set} - \frac{(\omega - \omega_{set})}{2\pi R} - P_{gv}\}_{(t-\Delta t)} \end{aligned} \tag{A.26}$$

This equation can be rewritten as follows

$$\begin{aligned} P_{gv_t}(1 + \frac{\Delta t}{2T_c}) + \frac{\Delta t}{4\pi T_c R}\omega_t =& P_{gv_{(t-\Delta t)}}(1 - \frac{\Delta t}{2T_c}) - \frac{\Delta t}{4\pi T_c R}\omega_{(t-\Delta t)} \\ &+ \frac{\Delta t}{T_c}P_{set} + \frac{\Delta t}{2\pi T_c R}\omega_{set} \end{aligned} \tag{A.27}$$

The function h at time steps t and $(t-\Delta t)$ and the constant C can be deduced from (A.27).

$$h(\omega, P_{gv})_t = P_{gv_t}(1+\frac{\Delta t}{2T_c}) + \frac{\Delta t}{4\pi T_c R}\omega_t \qquad (A.28)$$

$$h(\omega, P_{gv})_{(t-\Delta t)} = P_{gv_{(t-\Delta t)}}(1-\frac{\Delta t}{2T_c}) - \frac{\Delta t}{4\pi T_c R}\omega_{(t-\Delta t)} \qquad (A.29)$$

$$C = \frac{\Delta t}{T_c}P_{set} + \frac{\Delta t}{2\pi T_c R}\omega_{set} \qquad (A.30)$$

If $h(\omega, P_{gv})_t$ is renamed as h_4, the Jacobian elements related to the governor will be expressed as follows

$$\frac{d(h_4)}{d\omega} = \frac{\Delta t}{4\pi T_c R} \qquad (A.31)$$

$$\frac{d(h_4)}{dP_{gv}} = (1+\frac{\Delta t}{2T_c}) \qquad (A.32)$$

A.1.5 Turbine models

The Jacobian elements related to the low pressure cylinder are given in chapter 4 (equations 4.96-4.97). The algebraisation of this model is given by equation (4.88-4.93).

A.1.5.1 Turbine high pressure cylinder model

Replacing ΔP from equation (4.35) into equation (4.34) gives

$$\frac{dP_{hp}}{dt} = \frac{P_{gv} + K_1\Delta(DP) - P_{hp}}{T_{ch}} \qquad (A.33)$$

The algebraisation of this equation gives

$$\begin{aligned} P_{hp_t} =& P_{hp_{(t-\Delta t)}} + \frac{\Delta t}{2T_{ch}}[P_{gv} + K_1\Delta(DP) - P_{hp}]_t \\ &+ \frac{\Delta t}{2T_{ch}}[P_{gv} + K_1\Delta(DP) - P_{hp}]_{(t-\Delta t)} \end{aligned} \qquad (A.34)$$

This equation can be rewritten as follows

$$\begin{aligned} &P_{hp_t}(1+\frac{\Delta t}{2T_{ch}}) - \frac{\Delta t}{2T_{ch}}P_{gv_t} - K_1\frac{\Delta t}{2T_{ch}}\Delta(DP)_t = \\ &P_{hp_{(t-\Delta t)}}(1-\frac{\Delta t}{2T_{ch}}) + \frac{\Delta t}{2T_{ch}}P_{gv_{(t-\Delta t)}} + K_1\frac{\Delta t}{2T_{ch}}\Delta(DP)_{(t-\Delta t)} \end{aligned} \qquad (A.35)$$

The algebraic form of this equation is as follows

$$h(P_{gv}, P_{hp}, \Delta(DP))_t = P_{hp_t}(1 + \tfrac{\Delta t}{2T_{ch}}) - \tfrac{\Delta t}{2T_{ch}} P_{gv_t} - K_1 \tfrac{\Delta t}{2T_{ch}} \Delta(DP)_t \tag{A.36}$$

$$h(P_{gv}, P_{hp}, \Delta(DP))_{(t-\Delta t)} = P_{hp_{(t-\Delta t)}}(1 - \frac{\Delta t}{2T_{ch}}) + \frac{\Delta t}{2T_{ch}} P_{gv_{(t-\Delta t)}} + K_1 \frac{\Delta t}{2T_{ch}} \Delta(DP)_{(t-\Delta t)} \tag{A.37}$$

$$C = 0 \tag{A.38}$$

Renaming the function h of equation (A.36) to h_5 and differentiating it with respect to the related state variables gives the following three Jacobian elements

$$\frac{d(h_5)}{dP_{gv}} = -\frac{\Delta t}{2T_{ch}} \tag{A.39}$$

$$\frac{d(h_5)}{dP_{hp}} = (1 + \frac{\Delta t}{2T_{ch}}) \tag{A.40}$$

$$\frac{d(h_5)}{d\Delta(DP)} = -\frac{K_1 \Delta t}{2T_{ch}} \tag{A.41}$$

A.1.5.2 Turbine intermediate pressure cylinder model

Algebraising equation (4.36) gives

$$P_{ip_t} = P_{ip_{(t-\Delta t)}} + \frac{\Delta t}{2T_{rh}}[(P_{hp} - P_{ip})_t + (P_{hp} - P_{ip})_{(t-\Delta t)}] \tag{A.42}$$

This equation can be rewritten as follows

$$P_{ip_t}(1 + \frac{\Delta t}{2T_{rh}}) - \frac{\Delta t}{2T_{rh}} P_{hp_t} = P_{ip_{(t-\Delta t)}}(1 - \frac{\Delta t}{2T_{rh}}) + \frac{\Delta t}{2T_{rh}} P_{hp_{(t-\Delta t)}} \tag{A.43}$$

The general function h at time steps t and $(t - \Delta t)$ of this equation is

$$h(P_{hp}, P_{ip})_t = P_{ip_t}(1 + \frac{\Delta t}{2T_{rh}}) - \frac{\Delta t}{2T_{rh}} P_{hp_t} \tag{A.44}$$

$$h(P_{hp}, P_{ip})_{(t-\Delta t)} = P_{ip_{(t-\Delta t)}}(1 - \frac{\Delta t}{2T_{rh}}) + \frac{\Delta t}{2T_{rh}} P_{hp_{(t-\Delta t)}} \quad (A.45)$$

$$C = 0 \quad (A.46)$$

Renaming the function h of equation (A.44) to h_6 and taking its derivative with respect to the related state variables gives the following two Jacobian elements

$$\frac{d(h_6)}{dP_{hp}} = -\frac{\Delta t}{2T_{rh}} \quad (A.47)$$

$$\frac{d(h_6)}{dP_{ip}} = (1 + \frac{\Delta t}{2T_{rh}}) \quad (A.48)$$

A.1.6 Boiler storage model

Replacing equation (4.44) and Δf into equation (4.43) and assuming that $\Delta(SP)$ is constant over a time step, equation (4.43) becomes

$$\frac{d(\Delta(DP))}{dt} = \frac{\Delta(SP) - P_{hp} + P_{set} - K_3 \frac{(\omega - \omega_{set})}{2\pi}}{C_b} \quad (A.49)$$

Algebraising this equation gives

$$\begin{aligned} \Delta(DP)_t =& \Delta(DP)_{(t-\Delta t)} + \frac{\Delta t}{2C_b}\{\Delta(SP) - P_{hp} + P_{set} - K_3 \frac{(\omega - \omega_{set})}{2\pi}\}_t \\ &+ \frac{\Delta t}{2C_b}\{\Delta(SP) - P_{hp} + P_{set} - K_3 \frac{(\omega - \omega_{set})}{2\pi}\}_{(t-\Delta t)} \end{aligned} \quad (A.50)$$

This equation can be rewritten as follows

$$\begin{aligned} \Delta(DP)_t + \frac{K_3 \Delta t}{4\pi C_b}\omega_t + \frac{\Delta t}{2C_b} P_{hp_t} =& \Delta(DP)_{(t-\Delta t)} - \frac{K_3 \Delta t}{4\pi C_b}\omega_{(t-\Delta t)} - \frac{\Delta t}{2C_b} P_{hp_{(t-\Delta t)}} \\ &+ \frac{\Delta t}{C_b} P_{set} + \frac{K_3 \Delta t}{2\pi C_b}\omega_{set} \end{aligned} \quad (A.51)$$

The general function h at time steps t and $(t - \Delta t)$ of this equation is

$$h(\omega, P_{hp}, \Delta(DP))_t = \Delta(DP)_t + \frac{K_3 \Delta t}{4\pi C_b}\omega_t + \frac{\Delta t}{2C_b} P_{hp_t} \quad (A.52)$$

$$h(\omega, P_{hp}, \Delta(DP))_{(t-\Delta t)} = \Delta(DP)_{(t-\Delta t)} - \frac{K_3 \Delta t}{4\pi C_b}\omega_{(t-\Delta t)} - \frac{\Delta t}{2C_b}P_{hp(t-\Delta t)} \quad (A.53)$$

$$C = \frac{\Delta t}{C_b}P_{set} + \frac{K_3 \Delta t}{2\pi C_b}\omega_{set} \quad (A.54)$$

Renaming the function h of equation (A.52) to h_8 and taking the derivative with respect to the related state variables gives the following three Jacobian elements

$$\frac{d(h_8)}{d\omega} = \frac{K_3 \Delta t}{4\pi C_b} \quad (A.55)$$

$$\frac{d(h_8)}{dP_{hp}} = \frac{\Delta t}{2C_b} \quad (A.56)$$

$$\frac{d(h_8)}{d\Delta(DP)} = 1 \quad (A.57)$$

A.2 INTERFACE SUB-JACOBIAN

A.2.1 Interface generator/network

This represents the dependence of the non-linear swing equation and AVR model on the bus voltage. The dynamic/algebraic interface sub-Jacobian is represented by the derivatives of these dynamic functions with respect to network state variables (bus voltage).

A.2.1.1 Swing equation

The interface elements related to the synchronous generator are given by the derivative of equation (A.6) with respect to the real and imaginary components of the terminal voltage.

$$\frac{d(h_1)}{d(V_{ri})} = \frac{K\Delta t}{2}\frac{|E_g|}{|Z_g|^2}(X_d \sin(\delta) + R_g \cos(\delta)) - K\Delta t \frac{R_g}{|Z_g|^2}V_{ri} \quad (a)$$

(A.58)

$$\frac{d(h_1)}{d(V_{imi})} = \frac{K\Delta t}{2}\frac{|E_g|}{|Z_g|^2}(R_g \sin(\delta) - X_d \cos(\delta)) - K\Delta t \frac{R_g}{|Z_g|^2}V_{imi} \quad (b)$$

A.2.1.2 The AVR equations

The interface elements related to the AVR are obtained from differentiating equation (A.22) with respect to the voltage components are given by

$$\frac{d(h_3)}{d(V_{ri})} = (\frac{K_a \Delta t}{2T'_{do}})\frac{V_{ri}}{|V|} \qquad (a)$$

$$\frac{d(h_3)}{d(V_{imi})} = (\frac{K_a \Delta t}{2T'_{do}})\frac{V_{imi}}{|V|} \qquad (b) \qquad (A.59)$$

A.2.2 Interface network/generator

The Jacobian elements related to the static/dynamic interface are represented by the derivatives of the network models with respect to the dynamic state variables (ω and δ). These elements are derived from equation (4.51) and given below

$$\begin{bmatrix} \frac{dI_r}{d|E|} \\ \\ \frac{dI_{im}}{d|E|} \end{bmatrix}_{gi} = \begin{bmatrix} -G & -B \\ B & -G \end{bmatrix}_{gi} \times \begin{bmatrix} \cos(\delta) \\ \sin(\delta) \end{bmatrix}_{gi} \qquad (A.60)$$

$$\begin{bmatrix} \frac{dI_r}{d\delta} \\ \\ \frac{dI_{im}}{d\delta} \end{bmatrix}_{gi} = \begin{bmatrix} -G & -B \\ B & -G \end{bmatrix}_{gi} \times \begin{bmatrix} -\sin(\delta) \\ \cos(\delta) \end{bmatrix}_{gi} \qquad (A.61)$$

A.3 STATIC SUB-JACOBIAN

To illustrate the layout of the Jacobian matrix suppose that a generator g, a load ld, a compensator c are linked to node i which is linked to node j through a line ln and a transformer t. The voltages of all these elements are equal since they are located at node i.

Following Kirchhoff's first law the diagonal elements of this sub-Jacobian are given by

$$\frac{dI_{ri}}{dV_{ri}} = \frac{dI_{rlni}}{dV_{ri}} + \frac{dI_{rti}}{dV_{ri}} + \frac{dI_{rgi}}{dV_{ri}} + \frac{dI_{rldi}}{dV_{ri}} + \frac{dI_{rci}}{dV_{ri}} \qquad (A.62)$$

$$\frac{dI_{ri}}{dV_{imi}} = \frac{dI_{rlni}}{dV_{imi}} + \frac{dI_{rti}}{dV_{imi}} + \frac{dI_{rgi}}{dV_{imi}} + \frac{dI_{rldi}}{dV_{imi}} + \frac{dI_{rci}}{dV_{imi}} \qquad (A.63)$$

$$\frac{dI_{imi}}{dV_{ri}} = \frac{dI_{imlni}}{dV_{ri}} + \frac{dI_{imti}}{dV_{ri}} + \frac{dI_{imgi}}{dV_{ri}} + \frac{dI_{imldi}}{dV_{ri}} + \frac{dI_{imci}}{dV_{ri}} \qquad (A.64)$$

$$\frac{dI_{imi}}{dV_{imi}} = \frac{dI_{imlni}}{dV_{imi}} + \frac{dI_{imti}}{dV_{imi}} + \frac{dI_{imgi}}{dV_{imi}} + \frac{dI_{imldi}}{dV_{imi}} + \frac{dI_{imci}}{dV_{imi}} \qquad (A.65)$$

$$\frac{dI_{rj}}{dV_{rj}} = \frac{dI_{rlnj}}{dV_{rj}} + \frac{dI_{rtj}}{dV_{rj}} \qquad (A.66)$$

$$\frac{dI_{rj}}{dV_{imj}} = \frac{dI_{rlnj}}{dV_{imj}} + \frac{dI_{rtj}}{dV_{imj}} \qquad (A.67)$$

$$\frac{dI_{imj}}{dV_{rj}} = \frac{dI_{imlnj}}{dV_{rj}} + \frac{dI_{imtj}}{dV_{rj}} \qquad (A.68)$$

$$\frac{dI_{imj}}{dV_{imj}} = \frac{dI_{imlnj}}{dV_{imj}} + \frac{dI_{imtj}}{dV_{imj}} \qquad (A.69)$$

and the off-diagonal elements are

$$\frac{dI_{ri}}{dV_{rj}} = \frac{dI_{rlni}}{dV_{rj}} + \frac{dI_{rti}}{dV_{rj}} \qquad (A.70)$$

$$\frac{dI_{ri}}{dV_{imj}} = \frac{dI_{rlni}}{dV_{imj}} + \frac{dI_{rti}}{dV_{imj}} \qquad (A.71)$$

$$\frac{dI_{imi}}{dV_{rj}} = \frac{dI_{imlni}}{dV_{rj}} + \frac{dI_{imti}}{dV_{rj}} \qquad (A.72)$$

$$\frac{dI_{imi}}{dV_{imj}} = \frac{dI_{imlni}}{dV_{imj}} + \frac{dI_{imti}}{dV_{imj}} \qquad (A.73)$$

$$\frac{dI_{rj}}{dV_{ri}} = \frac{dI_{rlnj}}{dV_{ri}} + \frac{dI_{rtj}}{dV_{ri}} \qquad (A.74)$$

$$\frac{dI_{rj}}{dV_{imi}} = \frac{dI_{rlnj}}{dV_{imi}} + \frac{dI_{rtj}}{dV_{imi}} \qquad (A.75)$$

$$\frac{dI_{imj}}{dV_{ri}} = \frac{dI_{imlnj}}{dV_{ri}} + \frac{dI_{imtj}}{dV_{ri}} \qquad (A.76)$$

$$\frac{dI_{imj}}{dV_{imi}} = \frac{dI_{imlnj}}{dV_{imi}} + \frac{dI_{imtj}}{dV_{imi}} \qquad (A.77)$$

All these derivatives are given in section (4.5.3) in equations (4.98-4.103).