A FAULT TOLERANT ARCHITECTURE FOR NUCLEAR POWER PLANT CONTROL SYSTEMS

VOL. 1

by

Claudio Daniel ANTONINI

ING. Buenos Aires Institute of Technology (1978)

SUBMITTED TO THE DEPARTMENT OF
NUCLEAR ENGINEERING IN PARTIAL
FULFILLMENT OF THE
REQUIREMENTS FOR THE
DEGREES OF

MASTER OF SCIENCE

and

NUCLEAR ENGINEER

at the

MASSACHUSSETS INSTITUTE OF TECHNOLOGY

January 1982

Signature of author [illegible]
Department of Nuclear Engineering
January 28, 1982

Certified by [illegible]
David D. Lanning
Thesis supervisor

Certified by [illegible]
Paul J. Nicholson
Thesis reader

Approved by [illegible]
Allan F. Henry
Chairman, Departamental graduate committee

VoL. 1

A FAULT TOLERANT ARCHITECTURE FOR NUCLEAR POWER PLANT CONTROL SYSTEMS

by Claudio Daniel ANTONINI

Submitted to the Department of Nuclear Engineering on January 28, 1982, in partial fulfillment of the requirements for the degrees of Master of Science and Engineer in Nuclear Engineering

ABSTRACT

A distributed fault tolerant architecture is suggested to monitor and control processes of a nuclear power plant. The purpose of the system is to enhance the continuous safe operation of the plant and, simultaneously, improve its productivity.

A methodology for the design is suggested and, from the requirements of the system, the characteristics of the control system are determined.

The proposed system uses selected fault tolerant techniques largely based on the use of triads for survivability purposes.

The system consists of a fault tolerant central computer, fault tolerant distributed processors, a fault tolerant network and microcomputers interfaced to the sensors of the plant.

A number of design issues are treated: a) the layout of the elements of the control system in the plant, b) the determination of computational time and memory requirements for a Kalman Filter and a Hypothesis Testing algorithm and c) the reliability and availability of the system to meet possible regulatory requirements.

Examples and recommendations for the design are given.

Thesis Supervisor: Prof. David D. LANNING
Title: Professor of Nuclear Engineering
Thesis Reader : Dr. Paul J. NICHOLSON
Title: Visiting Scientist

ACKNOWLEDGMENTS

I acknowledge as one of the most powerful driving forces during my activities at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY the exemplifying atmosphere of this 'institution', 'established for the advancement and development of science, its applications to the industry, the arts, agriculture and commerce'.

In particular, the counseling and direction provided by Dr. David D. Lanning, and the suggestions given by Dr. Paul J. Nicholson are deeply appreciated.

During the initial phases of the work, material and suggestions provided by Dr. Albert Hopkins proved very useful for the design recommended in this Thesis and the concepts behind Fault-Tolerance.

Discussions with collegues, in particular Youcef Tumi and Les Gulko, provided with insight into some particular problems and suggestions for future work.

The suggestions given by Commander Domingo Giosetti with respect to my activities and interests are sincerily appreciated.

Finally, I thank the continuous and uncompromised support provided by my wife, Patricia.

Funds for this work were provided by the Argentine Navy.

BLANK PAGE

TABLE OF CONTENTS

CHAPTER 1

OVERVIEW

The objective of this introduction is to show how modern techniques, as applied to data acquisition, have not been considered to their full extent in the recent past and to describe methods which seem promising for future incorporation in the nuclear field. Given the scope of the subject, which justifies separate in-depth work, no attempt is made for completeness. The interested reader should consult the referenced material.

1.1 BACKGROUND

Nuclear reactor operations in the US prior to the Three Mile Island (TMI) event had not benefitted from the use of techniques that are currently available and which represent normal practice in other industries and in other countries. To give an example Dr. Kemeny made a comment about the obsolescence of control rooms which is applicable to our discussions. In 1980, as soon as he was in charge of the

President's Commission he visited some control rooms. In those opportunities he observed that these installations were at last twenty years out of date. Afterwards, he read in a document of the Nuclear Regulatory Commission (NRC), written ten years before, that control rooms were considered 'then' to be twenty years out of date.

Some of the reasons that allowed that situation are:

- the nuclear industry depended on heavy regulations that did not encourage application of state-of-the-art techniques to daily power operations. For example, computers were not used in control room environments because of seismic and safety class regulations. The regulatory process imposed delays that no utility was willing to face [GOLA80]. Lately, some improvements have been noted in the regulatory process.
- there were no formal studies considering the economic benefits due to the introduction of modern techniques in the operation of the plant. Therefore, it was feared that new regulations could be enforced without 'demonstrated' economical incentives. However just one of these techniques (disturbance analysis) has been estimated to increase to availability of the plant by 2.2 percent or, equivalently, 8 days per year [COMB80]. Considering that utility losses are from 0.6 to 1.0

million dollars per day if the plant is not in operation (due to replacement costs, labor, materials), in one year that utility could be saving from 5 to 8 million dollars or, approximately, one half or more the cost of a computer system.

- not enough space to allow CRTs was included in the control room of the early designs (ie the CRTs had to be imbedded in existing consoles, fixed to the ceiling, etc).
- the allocations of tasks between man or machine and automatic or manual control were not well defined.

1.2 EXPERIENCE WITH PROCESS COMPUTERS

The role of process computers has traditionally been reduced to that of a powerful data logger, printer-driver and off-line core-oriented calculator. They have proved useful in helping the operators to determine the 'state' of the plant, although much more could have been obtained even with the technologies available at the time the plants were designed.

The importance of process computers in plant operations is described in [MACR77], a very well documented report and rather complete checking list for the process control computer designer. In the study 156 US nuclear and fossil power plants were surveyed. Some of their findings and conclusions are listed below.

- When the utility personnel were asked questions referring to the goals or benefits to be achieved by the computer system it was surprising to note that their responses consistently failed to identify goals. They simply concentrated on describing computer functions. For example, if they considered a better understanding of the 'state' of the plant (goal) they mentioned that they needed more data logging (function) at the higher level (the higher level corresponds to the central node or central computer and the lower level is represented by the elements directly connected to the sensors).
- Some to the functions suggested for incorporation into the computer system were actually counterproductive. By flooding the operator with non-predigested (raw) data, the computer output could easily saturate him and lead to reduced understanding of the process.
- A more advisable solution would be to combine some data (derived or processed as oppossed to raw data or direct measurements) in a more meaningful way to the operator, so that he could understand the situation immediately, without doing any calculation or having to transform direct measurements into recognizable parameters. As an

example, in the OVERVIEW of reference [KEME79], it is said with respect to the presentation of data in the TMI-II control room that 'although the pressure and temperature within the reactor coolant system were shown, there was no direct indication that the combination of pressure and temperature meant that the cooling water was turning into steam'. Therefore, a failure to recognize an important objective of the control system (to improve the safety of the plant by improving the operator's undestanding of its behavior) and a rather inappropriate implementation of the means to achieve that objective (presentation of data in the control room) led to degraded performance of the operator and, consequently, to further problems in the plant.

- There was no response from the 156 power plants stating that they did not need a process computer; instead, some of them said that 'as it was designed' it was not possible to obtain the best results from it.
- The malfunctions of the process computers were originally intended to have no impact on the operation of the plant. However, 78 plant superintendents indicated that unavailability of the computer would delay start-up or hold power generation or derate the plant.

- The installed computer systems were rated as follows: Good = 17%, Fair = 50% and Poor = 33%.
- Long construction delays has led to unavoidable obsolescence. Due to the time spent and the delays inherent in the construction process, by the time the utility started to operate the system it was faced with the fact that the computer manufacturer did not exist anymore or that most of the modifications were very difficult to perform due to differences in the technologies.

1.3 POST-TMI SCENARIO

Due to the TMI incident part of the situation has changed [LIVI80]. Some of the recommendations of the Kemeny Commission [KEME79] have been incorporated in new regulations [NUCL80,1] and designs based on the spirit of these guidelines have been emerging [NICH80]. These designs are all intended to minimize the risk of accidents involving the public. In this discussion 'risk' is defined as the probability of an occurrence times the consequences of that occurrence. To satisfy that goal a system is required that allows the determination of the state of the plant more accurately and efficiently than is done nowadays. A simple statement like this one leads to new questions and conclusions:

- although every designer and user accepts that the 'state' of the plant has to be determined, there is no consensus on the parameters that form the 'state', the list changing with vendor, type of reactor (PWR, BWR), operational state (start-up, 100% power, refueling), etc.
- assuming that a system like the one suggested is designed and operates succesfully, its implementation would benefit the operation of the plant enormously, through a decrease in partial reductions, minimization of unnecessary trips, reduction in outage duration (planned or forced) and early detection of incipient failures to prevent catastrophic ones [LONG77]: in summary, improved plant operation and maintenance. All these could imply that a Productivity aspect may be incorporated to the Safety aspect of the design.

1.4 ADVANCED TECHNIQUES APPLIED TO NUCLEAR REACTOR OPERATIONS

Application of advanced techniques (noise analysis, disturbance analysis, parameter estimation, adaptive and optimal control, digital signal processing, analytic redundancy, fuzzy sets, artificial intelligence, supervisory control, pattern recognition, man machine interface, etc) in nuclear power plants had very minimal use in the US and are by no means standart; however, some have been tested in research reactors like the Halden project in Norway (direct digital control) or the MIT Reactor (sensor validation and CRT display development).

The following succint list references some of these techniques applied to the nuclear field:

- data acquisition and performance monitoring [SPIT80],
- core thermal hydraulics and fault detection and identification [ORNE81],
- modeling of the pressurizer [GEFF80],
- linearized model of the primary system [NAKA81],
- modeling of the feedwater system [MOLI81],
- modeling of the plant and set theoretic control [CHEN80],
- set theoretic control applied to load following operations [GULK81],
- displays of multivariate plant data [BORG81],
- measurements of mental workload in supervisory control [DARY80],
- allocation of tasks between man and machine in process control [ROUS81].

Some of these projects and thesis have been done as part of a joint Charles Stark Draper Laboratory - MIT project and others are in progress at the time of this writing.

The ultimate desire of a control engineer - to provide full automatic control of a plant - is only in the talking stages due to:

- licensing requirements that question the reliability of an automatic control system and the possibility of interaction between the control system and the 'engineered safeguards system'.
- research studies with conclusions that, for single-input single-output systems, the effort to design LQG controllers was not compensated by the results because they were similar [BLOM77] or scarcely better [GRUM68] than conventional analog PID controllers. However, LQG theory and its automation had advanced to a degree in which these statements may not be true anymore, specially in the more difficult multiple-input multiple-output case.
- nonlinearities, time delays, noises and space-time kinetics that make the analysis of the plant (and the reactor in particilar) far from straigthforward.

In references [FROG78], [McMO79] and [TZAF80] descriptions of approaches to the control of nuclear reactors are given. In particular, multivariable methods are treated with special detail due to their promising results.

1.5 MULTIVARIABLE METHODS

Multivariable methods can be broadly divided into two categories: time-domain and frequency domain. The main outstanding difference is that synthesis procedures were developed to obtain the feedback system design directly from the problem specifications in the time-domain, a situation that it is not characteristic of frequency-domain approaches. The latter can be thought of as being a design tools more than synthesis methods.

In the time-domain (also called the state-space approach) the differential equations describing the behavior of the plant are expressed in terms of a set of first order equations. This is a way of introducing the influence of the past history into its present dynamics, as oppossed to the 'instantaneous' relationship between inputs and outputs in the frequency-domain. See references [ATHA71] and [DOYL80].

The technique most widely used is the Linear-Quadratic-Gaussian (LQG) design, in which the plant is represented by linear first-order differential equations; the objective is to minimize a performance index which has quadratic weigths (on a combination of states, measurements and controls) and noises are gaussian. Another promising technique is the Set-Theoretic control method, in which the plant is also linear, but now the disturbances may have unspecified statistics, the only requirement being that they are bounded. Bounds are also specified on the states and the controls. The objective of this procedure is to treat the states and control constraints explicitly and to provide the feedback law just with these specifications. An initial study on the potential applications to nuclear plants was developed by Chenini in [CHEN80].

In general, some of the advantages of these state-variable methods are that:

- they allow control of decoupled loops and highly interacting loops with equal facility.
- they can very easily handle the nonstationary case (non-constant matrices in the standard formulation).
- in some cases they can even work with limited nonlinearities (ie the extended Kalman Filter).
- there is no requirement for the model to have the same structure as the plant: the state variables can be determined by any parameter estimation techniques. However, the use of an appropriate structure may reduce the sensitivity of the control law to modeling errors.
- many of the performance criteria which the final control schemes have to satisfy can be interpreted as economic constraints. For example, in the nuclear reactor industry the criteria may be to

minimize the amount of xenon so that the reactor is not shutdown for long periods of time.

But perhaps the most important advantage is that they provide a systematic methodology to assure the 'stability' (in a broad sense) of the design in the presence of various uncertainties (modeling uncertainties or errors, plant or measurement noises, error in the determination of initial conditions) and particularly in a very important situation: the detection and/or identification of sensor or actuator failures that may 'open' the closed loops.

The conditions under which this is succesfully accomplished impose very mild constraints on the formulation of the control specifications. The robustness of the design and the possibility of improving it systematically are characteristics lacking in most of the frequency-response methods. For a rigorous definition of robustness and its application to multivariable systems see references [SAFO77] and [LEHT81].

The disadvantages of state-space methods are:

- they increase the order of the system. In the worst case they double it and hence it may expensive to implement it in real time.

- they tend to be too 'sophisticated' to be accepted for use by industrial engineers brought-up on sigle-input single-output frequency-response ideas who essentially needed to use a mixture of physical insight and straightforward techniques such as the use of integral and derivative controllers.

With respect to frequency-response methods, the systematic attack on the development of analysis and design theory for multivariable feedback systems has been initiated by Rosenbrock [ROSE69], rejuvenating the frequency-response approach. His use of a particular technique, the Inverse Nyquist Array method, opened up a new line of development by seeking to reduce a multivariable problem to one amenable to classical techniques (Bode plots, Nyquist diagrams, etc) by reducing the interaction between different loops. In that way, single loop techniques could be employed.

The frequency-response model of the plant can be obtained from step-response tests or frequency-response measurements exciting the plant with appropriate signals. The method is not sensitive to model innacuracies since the user can allow for reasonable uncertainties as he develops the design. If it is succesful it leads to the possibility of a simple controller. However, in reference [LEHT81], this technique is shown not to guarantee robustness. Specifically, the robustness can not be guaranteed at the essential interface between the plant and the sensors and actuators; an interface where failure of these components can compromise the safety of the plant. The same drawback applies to another technique: the Characteristic Locus, which is based on the complex transfer function matrix. Here, the eigenvectors obtained from this matrix are a measure of dynamic interaction and is one of the paramenters that the designer uses for his interactive design; by 'judiciously' choosing matrix operators he attempts to modify them and to obtain suitable closed-loop stability.

Some of the advantages of frequency-response methods are that:

- there is no need to obtain a great physical insight into the process that occur inside the controlled plant (although this will always help) because the transfer function matrix can be obtained fairly simply.
- even crude estimates of dynamics and interaction may give good results.
- there is no concern for state estimation, uncontrollable or unobservable modes (the requirements for the state-space approach to work).

The drawbacks of these methods consist of:

- lack of systematization. Frequency-response methods cannot be qualified as synthesis tools, but only as design or analysis tools.
- difficulty of interpretation of results. Their concepts are even more difficult to be interpreted by control engineers than time-domain techniques.
- difficult to use in highly interacting loops and high order systems.

Other characteristics of these methods and their application to nuclear power plants can be seen in reference [McMO79].

If the final objective is "applying the available modern control techniques to full-scale reactor plants including all important controlled variables, all phenomena taking place and all subsystems involved from the reactor core to the external load" [TZAF80], some more work is necessary.

1.6 THESIS OVERVIEW

Chapter 2 presents a recommended systematic method of design that takes in account all aspects of the life-cycle of the system, the requirements for the system to be designed, an explanation of the concepts of distributed systems and fault-tolerance and a proposed preliminary design.

In Chapter 3 the connection of sensors, multiplexers, micro-minicomputers and central computer, as proposed in the preliminary design, is solved in the case of a PWR plant.

Two examples of the process the designer has to follow to verify that his design will fulfill real-time requirements are presented in Chapter 4 -applied to a Kalman Filter- and in Chapter 5 -in the case of a Hypothesis Testing algorithm-.

The modeling of the reliability and availability of the nodes and the network is considered in Chapter 6. As a result of the insight gained with the use of these models, recommendations for the design of a network are given in Chapter 7.

BIBLIOGRAPHY FOR CHAPTER 1

[ATHA71] ATHANS, M., Special Issue on Linear Quadratic Gaussian Control, IEEE Trans. on Autom. Control, December 1971

[BLOM77] BLOMSNES, B., Two applications of LQ theory to LWR plant control, Proc. 1977 Joint Automatic Control Conference, San Francisco, CA

[BORG81] BORGHESE, J., Human performance using column and star format multivariate color CRT displays, MS thesis, MIT, 1981

[CHEN80] CHENINI, A., Set-theoretic control of a PWR plant, NE Thesis, MIT, 1980

[COMB80] COMBUSTION ENGINEERING, INC. et al, On line power plant alarm and disturbance analysis system, EPRI-NP-1379, April 1980

[DARY80] DARYANIAN, B., Subjective scaling of mental workload in a multi-tasking environment, MS Thesis, MIT, 1980

[DOYL80] DOYLE,J. et al., Multivariable feedback design: concepts for a classical/modern synthesis, Laboratory for Information and Decision Science, MIT, LIDS-P-996, May 1980

[FROG78] FROGNER, B. et al., Control of nuclear power plants, IEEE Trans. Aut. Control, June 1978

[GEFF80] GEFFRAY, C., Nuclear reactor (PWR) pressurizer real-time modeling for sensor validation, MS Thesis, MIT, 1980

[GOLA80] GOLAY, M., Nuclear regulation in America, Technology Review, June/July 1980

[GRUM68] GRUMBACH, R. et al., Application of modern control theory to digital control of HBWR, Proc. ENEA Symposium on Applications of on line computers to nuclear reactors, 1968

[GULK81] GULKO, L., Set-theoretic tracking, Theory and applications, MS Thesis, MIT, 1981

[KEME79] KEMENY, J. et al., Final Report of the President's Commission on the accident at Three Mile Island, 1979

[LEHT81] LEHTOMAKI, N., Practical Robustness Measures in Multivariable Control System Analysis, PhD Thesis, MIT, 1981

[LIVI80] LIVINGSTON, W., The impact of TMI on process control computer technology, Control Engineering, May 1980

[LONG77] LONG, A. et al., Justification for future trends in nuclear plant control and protection systems, 1977

[McMO79] McMORRAN, P., Multivariable control in nuclear power stations, survey of design methods, AECL-6583, December 1979

[MACR77] MACRO CO., Documentation of utility experience with process computers in power plant applications, EPRI-NP-254, MAY 1977

[MOLI81] MOLINA, J., Analysis of the feedwater system for a PWR, MS Thesis, MIT, 1981

[NAKA81] NAKAGAMA, G., Linearized model real time reactor simulation, MS thesis, MIT, 1981

[NICH80] NICHOLSON, P. and LANNING, D., Requirements and concepts for a nuclear plant surveillance and diagnostics system, IAEA Specialists Meeting on Distributed Computer Systems for Nuclear Plants, Chalk River, Ontario, Canada, 1980

[NUCL80,1] NUCLEAR REGULATORY COMMISSION, NRC Action plan developed as result of the TMI-II accident, NUREG-0660, May 1980

[ORNE81] ORNEDO, R., Development and assesment of a PWR core model for real-time diagnostics, MS Thesis, MIT, 1981

[ROSE69] ROSENBROCK H., Design of multivariable control systems using the Inverse Nyquist Array method, Proceedings of the IEEE, November 1969

[ROUS81] ROUSE, W., Human-computer interaction in the control of dynamic systems, _Computing surveys_, March 1981

[SAFO77] SAFONOV, M., Robustness and stability aspects of stochastic multivariable feedback system design, PhD Thesis, MIT, 1980

[SPIT80] SPITZNER, K., The North Eastern Utilities Generic Plant Computer System, Hartford, CN, 1980

[TZAF80] TZAFESTAS, S., Control of nuclear power plants, _Autom. Control_, _8_, 2, 1980

INDEX

CHAPTER 2

A PRELIMINARY DESIGN

In this Chapter, a methodology for the design of a distributed computer system will be presented and concepts regarding the necessary tradeoffs, to satisfy the requirements, will be introduced. This procedure will be applied to the monitoring of processes in a nuclear power plant.

The term 'monitoring' has been used in the Thesis according to one of the definitions provided by The Webster's Dictionary [WEBS77]: "to keep track of, regulate, or control the operation of (as a machine or process)". The purpose of using this particular definition is that the concepts of <u>surveillance</u> and <u>control</u> are both included in the concept of 'monitoring'.

2.1 ON GOALS, OBJECTIVES, REQUIREMENTS AND SPECIFICATIONS

In Chapter 1, reference has bee made to a 'computer system' thus implying that a computer system is the correct solution. In this Chapter, a justification for that assumption is sought.

The design of the current generation of computer systems is a complex process evolved from ad-hoc rules and the desire to incorporate more functions at the processing level. Early systems were not always completely satisfactory for the following reasons:

- the final objective of the system (or the goal) was not clearly specified.
- the system was designed with one purpose in mind, and that purpose was given by the designer of the system, not by the prospective user.
- it was difficult to modify the system to extend its applicability.
- maintenance procedures were inadequate resulting in reduced availability of both the computer system and system being controlled.
- reliability goals were not explicitly addressed in the design.

The feedback from this experience has enabled a more systematic design process to be evolved which includes defined phases and iteration checkpoints. Although there are many methodologies, they all begin with the definition of the objectives to be accomplished by the proposed system.

Under the new philosophy, every design is the consequence of a collective decision process undertaken by three groups: the user, a supervisor selected by him and a design team. Each one has a different aim in the design procedure. The user needs a means of accomplishing its goal; in conjunction with the supervisor he defines the set of requirements to be obeyed by the system. Both check that the design really works by validating the expected system performance against predefined specifications. The designer is in charge of translating these requirements into specifications with his knowledge of the technology.

The use of the following concepts, used later in this Chapter, require a precise definition: goal, objective, requirement and specification.

<u>GOAL:</u> The goal is the (highest) purpose to be accomplished by the system, as defined by the user.

This definition does not match with the one provided by [GOLD79] where he defined goal as 'a statement of some system condition that is desired without a prescription of the means of achieving that condition'. This statement really corresponds to the definition of requirement provided below.

In this thesis the highest purpose is to provide a means of assuring the continuous safe operation of a nuclear power plant.

<u>OBJECTIVE:</u> while the goal is defined by the user with respect to the purpose of the system , the objective is defined by the supervisor and designer. For simplicity it will be assumed that the objectives of these two groups are the same although an extension to consider that such is not the case is straightfordward. The goal is an abstract definition communicated to the designer and the supervisor group by the user.

Differences in the terminology stress that only in very few cases will both goal and objective completely agree. In most of the situations the user will try to use the system for purposes not contemplated in the design and the supervisor and designer will also make efforts to incorporate ideas of their own. This process has been commonly compared to that of a source (the user) sending information to a receiver (the other groups) through a noisy channel, a paralellism with concepts of information theory. According to this theory some information is lost in the channel (the user cannot communicate all of his ideas) and some noise is taken by the supervisor and designer as valid information (the functions they add to the system to 'satisfy' the user). The only solution for making this discrepancy small is to provide feedback at different phases of the life-cycle of the system design evolution.

In this thesis the design objective is assumed to be that _'the system assures the safe operation of the plant and that, in addition and without being in conflict with that goal, improves its productivity'_.

It can be thought that this objective was worked out with the user because the experienced the designer tells the user that it is possible to include some concepts and functions in the system's design without adversely affecting the user's goal, facilitating system's update and maintenance and easing the ground for future regulatory allowances (or needs).

Note that there are many ways of improving the safety of a plant and its productivity. For instance it is possible to perform reliability evaluation programs and identify deficiencies by correcting them; to use of simulators in the training of operators; to write better documentation; to improve the design of control rooms, etc. It is assumed that these alternate methods of achieving the user's goal have been considered by the user or the supervisor already.

REQUIREMENT: global properties that all operations at a particular interface must satisfy. The requirements do not have to referenced to any implementation because the most effective way of accomplishing a given requirement may be by different functions at different levels (in a hierarchical design). In fact, by defining the requirements at the interfaces (the higher one being the operators or other computers and the lower one the sensors) the principle of no-implementation is invoked.

Procedures and objectives to write down the requirements were detailed in [HENI80]. His approach to requirements documentation can be condensed in three principles:

- formulate questions before answering them. In that way distinct 'concerns' will be identified,
- separate these concerns assigning different items to different teams so that if a change has to be made to a certain item, other teams would not have to revise their designs,
- be as formal as possible.

On the other hand the concerns should be identified so that the requirements formulation would accomplish the following:

- _Specify external behavior only;_ do not imply a particular implementation.
- _Describe the interfaces so that the design teams know the constraints._ In this thesis the higher level of the proposed system will interface with operators and other computers; the lower level

will interface with the sensors already installed in the plant. Both define clear constraints to the designers. The higher level constraint will indicate the approach to man-machine interface that should be used by the software team and how to organize the data bases so that they can easily accessed (with minimum delay and high reliability), and the hardware team will know what protocols must be used with other computers, where and how to control the displays and how much memory they will need for that purpose.

- Design to allow changes to be made easily. Modifications in requirements will imply changes in design and implementation. The more independent they are from the beginning, the easier changes will be made later in the design.
- Document precisely to provide good reference material. The purpose of the documentation is not to be a tutorial, rather it must serve other designers to understand quickly what was intended.
- Record forethought about the life-cycle of the system. For example, in this Thesis it was envisioned that in the future there is a possibility to include more sensors, microprocessors or actuators to implement control of some of the plant subsystems. Consequently, the design must accommodate at least two scenarios: an

initial one with a minimum number of signals going from the sensors to the computer system and an extended one where there will be additional control signals going from the computer system to the actuators.

- Characterize acceptable responses to undesired events. Examples of undesired events have been considered in this case to be sensor, hardware and software failures and plant transients.

SPECIFICATION: A specification becomes the definition of the expected performance for a particular operation or module of a system. Each specification, or group of specifications, has to satisfy a requirement.

It is useful to break the specifications down into functional (or logical) and performance specifications. Functional specifications define the operation or module functions (function + constraints + accuracy + relationship with other modules) and the performance specifications define the resources needed by the operations or modules (processors, memory, time, sensors, data links) to meet their functional specifications.

The following is a brief analysis of two requirements and specifications which will be treated extensively in Chapters 4 and 5. Both examples correspond to the definition of functional specifications. Given that the determination (by measurement or estimation) of state variables is necessary to perform most of the functions of the system, a Kalman Filter implementation is discussed in Chapter 4. The Kalman Filter is chosen in preference to other filtering solutions to minimize the mean square error between the estimated and the actual variables. Once the variables are determined they may be used for other purposes: for example, to determine if a parameter is out of range due to system or sensor malfunctions. An algorithm (the Sequential t-Test) is suggested in Chapter 5 to detect if the computed mean of a series of measurements is 'representative' of the true mean of the population. This method is used to satisfy another kind of performance specifications, in this case time specifications because the algorithm assures that the detection is made in a minimum number of observations.

2.2 DESIGN METHODOLOGY

Many studies cover different aspects of computer design: ie fault tolerance in distributed processing [GOLD79], reliabilility design for computer control [ROSE81], software maintainability [MUNS81], software validation [DEUT81] and [LEVI81] and overall design [TURN77] and [WEIT80]. The last two provide guidelines which can be succesfully integrated into a single one more powerful methodology.

According to reference [TURN77] it is known that there are several myths related to the development of computer systems, namely:

- Stating valid and complete requirements for a new computer system is a relatively simple task.
- Hardware and software can be purchased or developed separately and fit together later, and subsequently adapted to the administrative and procedural environment.
- Software is different from hardware and must be managed differently.
- Management review mechanisms used in hardware development are superflous for software (or are impossible to conduct).
- Many hardware inadequacies can easily be offset by simple software changes.
- Acquisition of software can be treated as a production-like process, similar to procurement of standard hardware.
- Software, once developed, hardly ever needs to be changed again.
- Support of software in operational systems is essentially the same as maintenance of the hardware.

These inadequate concepts have produced the frustations mentioned in Section 2.1 due to shifted schedules, unsatisfactory performance, dilution of responsibilities, costs exceeding estimates and unmaterialized benefits.

The methodology proposed in this Section is based in those introduced by Turn [TURN77] and by Weitzman [WEIT80], although some modifications are introduced. FIG.2.1 indicates some of its characteristics: phases, feedbacks, checkpoints and responsibilities.

As indicated in Section 2.1, three kinds of organizations are assumed to be responsible for the whole computer system development. They are the user, the supervisor and the designer. The relationship among them should be made evident by constructive criticisms, efforts to accomplish the goal and well defined authority areas. FIG.2.1 illustrates some of these concepts. The definition of the goal and the selection of the supervisor is necessarily left to the user; the definition of the requirements is a primary responsibility of the user, but it is advisable to receive suggestions from the supervisor. On the other hand, the Specifications definition phase is a primary responsibility of the supervisor and now the user must simply be an authorized observer. In other phases of the methodology it will be important that these three organizations are constantly informed of the system design progress. Whenever 2nd or 3rd priorities of supervision are indicated, it is understood that they mean that the organizations should be informed of the results of that phase, and detailed information should be available only on request.

Feedbacks have been incorporated into the model indicating tradeoffs, modifications, redefinitions, etc. From the beginning, checkpoints have to be defined stating how the progress of the design will be measured (ie parameters used to evaluate the design may be cost, total number of people recruited for the project, number of lines of source code written (in the Development Phase), man-hour figures, etc)

A brief explanation of the Phases of the methodology will now be presented.

<u>REQUIREMENTS DEFINITION</u>: during the requirements definition phase the organizations in charge of defining them (the user and the supervisor) will attempt to assure 1) that the definition of the requirements really represents what the goal intends to encompass, 2) that that systems already installed in the user's organization cannot satisfy the requirements to be a viable alternative to the proposed system and 3) that these existing systems cannot be upgraded to satisfy the totallity of the requirements. In summary, it is intended to know if a new system is the best solution for the organization.

The concept of 'system' in the previous paragraphs do not refer exclusively to computer systems or control systems, to give some examples. There, the concept of system is much broad, and it is related to show that procedures and functions that are performed by the user's organization at the moment of the design of the new computer system cannot fulfill the totallity of requirements.

The list of requirements for the system proposed in this Thesis is shown in FIG.2.2. Note that the necessity of having a computer system is a <u>consequence</u> of requirements number 2 (real-time processing), 5 (keeping large records and retrieving data at request) and 6 (interfacing with other computers).

<u>SPECIFICATIONS DEFINITION</u>: during the specifications definition phase, the validation of the technical and economic aspects of the requirements is made. Users and developers are confronted to ascertain the technical feasibility of implementing a system to satisfy the stated requirements. Tradeoffs are expected to be made and the feedback with the previous phase (Requirements definition) will be intense. The following questions must be answered (adapted from reference [TURN77]): What degree of technical risk or uncertainty is involved in meeting the requirements? Is any research indicated? For the problem considered in this Thesis detailed algorithms and models of the plant should be available or they would have to be

developed. Is the projected schedule appropriate compared to the schedule of the construction of the plant or the planned outages of the station? Are there any particular technical aspects that require extraordinary management measures?

After completion of the technical review and several iterations of reinterpreting requirements and modifying specifications, a detailed group of specifications is given. This set is outlined below in Section 2.4.

The Specifications Design phase should be performed mainly by the supervisor's group, because it has experience in such a work. The supervisor's group would work very closely with(in) the user's organization in order to understand how his ideas can be succesfully materialized in that environment.

DESIGN: the design phase should be performed by an organization independent of the user and the supervisor organizations.

The first part of the design phase (the Preliminary design) may be based on a set of techniques which can be partly automated. The idea behind the use of these techniques is that the functions that the system has to perform can be decomposed into so-called 'abstract processes' and interactions between them.*

For example, if the idea is to design a monitoring system some of the abstract processes that can be identified at the lowest level of the system (as defined in Chapter 1) are:

- a timing method (clock) to initiate activities
- acquisition of signals from sensors
- conversion of signals into meaningful (engineering) units ('signals' are transformed into 'data')
- filtering of data
- tables containing (upper and lower) limits for the sampled data
- overall control of the acquisition process
- modelling of the process of the plant being sampled
- comparison between sensor data and previous data (filtered) or analytically derived (analytic redundancy)
- indication of alarms
- detection of failed sensors
- self-tests for the unit

---

* An abstract process is informally defined as the activity resulting from the execution of a program with its data by a sequential processor. See reference [WOLF77] for a rephrase of this concept in terms of concepts from state machine theory.

Some of the interactions among processes are:

- clock signals to initiate processes
- numerical values from the filtering and modelling processes to be compared in another process
- alarms sent from the comparison process to the indicator of alarms.

Once the processes have been defined and their interaction perfectly stated, an implementation is sought. The implementation is made by grouping processes into sets so that the interaction (exchange of data) <u>between</u> sets should be much less than the interaction <u>inside</u> each set. An example of this implementation may be that variables which are needed for processing in one section of the plant should be located in non-volatile memories near the process, and only those variables that are used for calculations involving more than one process should be transmitted to another physical location. With this concept of 'local processing' the amount of data transferred in the net is kept very low. This is the approach followed in this Thesis for the design presented in Section 2.4.

Upon completion of a preliminary design, the detailed design begins. An iterative process is to be expected between these two phases. New information that is available may modify some design constraints considered earlier in the design; ie, new technological improvements, budget modifications, etc.

Finally, the development of the system includes an implementation of the detailed design in such a form that can be tested and, in the case of hardware, passed into production. For software, a number of methodologies are available which should be considered as leading to neat designs, good interaction with the hardware design and useful documentation. It must be remembered that, although the 'structural form' of hardware (equipment) and software (lines of instructions) differ, some requirements are applicable to both of them (testability, documentation, maintenance). Software design teams should consider these as requirements also applicable to their final product, even though such requirements have not been strictly specified (simply because requirements must be developed independently of implementation).

The following three phases have not been modified and they were originally defined by Turn et al. in the referenced material [TURN77].

<u>TESTING</u>: the testing phase comprises three distinct steps (8 and 9 in the notation of FIG.2.1.) involving distinct organizations (the user and the supervisor). The first step is not necessarily totally separate from the development of the system (the last step in the design phase). In fact, it is assumed that the designer will test

the system against a limited set of parameters before the date specified to proceed to the next phase (where the supervisor will test it more thoroughly). Testing of the system performance aganinst the design specifications (indicated as 8) must be performed by a technologically competent group independent of the system developer (the supervisor). The principal question is whether the system performs as specified in the detailed design specifications produced in step (4).

The second step (9) must be performed by the eventual users to test the systems's functional performance and to see if it satisfies the user's needs. It may turn out that although the original specifications are satisfied, the user finds that the real implementation of that set of specifications does not completely satisfy his requirements. New iterations will then be required.

Finally, when the system works as it is required, a very important phase is performed: the documentation corresponding to the tests performed and the actual responses from the system in the factory environment. This phase, frequently overlooked by the user (but not by the designer), will provide benchmark results for future releases of the system.

PRODUCTION AND INSTALLATION: in hardware systems, completion of the test step is followed by conventional factory production. For software, the production phase is minimal: error-free copies of the original programs must be made and distributed to the user. The absence of a software production phase is one major distinction between hardware and software, specially in terms of funding. Nearly all of the software acquisition cost consumes development funds, putting additional strains on a traditionally tight budget category.

OPERATIONAL USE AND MAINTENANCE: the final test of the system's ability to satisfy the need defined by the user comes in the actual operational use of the system in the user's power plant. At this point, the user has formally accepted the system from the designer, and the responsibility for support is transferred to the user, to another supporting organization or it could remain with the designer itself.

Maintenance guarantees that the system remains operational despite the natural deterioration of the system's hardware. Since there is no intrinsic wear-out of software, software support mainly involves product improvement (increasing efficiency, reliability or supportability) and correcting design oversights and errors.

MODIFICATIONS: this phase can initiate profound changes in the system that they may force a complete repetition of the preceding steps. The most 'limited' changes will occur in the case of activities initiated once the system has arrived at its final destination and is installed (ie maintenance). Modifications in the installation step could 'simply' imply a repetition of the design phase. However, most of other modifications (new user goals, new set of requirements, product enhancement through incorporation of new technologies), could in general imply a reversion to even earlier phases.

2.3 DISTRIBUTED FAULT TOLERANT COMPUTER SYSTEMS

As an introduction to Section 2.4, where the specifications for the proposed system are given, the concepts of distributed computer systems (DS) and fault tolerance (FT) need precise definition. In fact, many systems already in the market have these characteristics, implemented in one way or another, although the concepts on which they are based are not found clarified anywhere in the design process. For example, some systems, although physically decentralized, are really centralized with respect to the management and use of information, an attribute that makes them really centralized systems (ie plain remote multiplexing). With respect to FT many systems already in the market apply it in some, but not all, design levels; the malfunctions for which the design is fault tolerant are not characterized; reliability calculations are nonexistent or extremely simplified and validation of the design has never been made.

The concepts of DS and FT need to be integrated as a necessity of achieving higher survivability than in centralized systems in environments where large processing of data is required. Two commercial applications are banking and industrial control systems. However, in these applications there is often no guarantee that the system performs with any level of reliability or availability. The vendor may guarantee maintenance or the provision of spare parts by contract, but not any level of performance.

By contrast, in highly critical situations (aircraft control systems, power plants, communication systems, etc) there may be a need to assure and test for a desired level of survivability. In most of the situations the way in which the survivability of the system can be validated is by making conservative assumptions and using judiciously chosen analytical or simulation methods. Real demostrations to check some parameters (ie mean time between failures) may take a very long time, use resources unprofitably and have a very high standard deviation (because of the small samples considered). Such actual demonstrations should only be used when there a fairly significant number of componentes to test and when the material and labor costs of repeating the experiments is not substantial. In this category the most complex components that can be tested are boards or, in a few cases, groups of boards (and connectors and cooling fans and power supplies) forming a working unit. However, the threats, malfunctions, consequences and means of coping with them will differ according to the component or group of components that it is tested.

Two uses of distributed computer systems in the nuclear plant environment are those directly related with monitoring and control of the processes. For such a purpose the most common structure has a <u>hierarchical configuration</u> consisting of a) a central computer, b) groups of other (typically less powerful) computers distributed either functionally or physically over the plant and c) links that connect these two groups. The central computer has the roles of supervision, system reconfiguration, communications control, display control, long term central memory, communication with other computers and interaction with operators. The distributed computers are somewhat autonomous, dedicated to special functions, easily expandable and usually replicated for reliability. These two elements communicate by means of links to form an organized system. If CRTs and 'smart' sensors (sensors coupled with microcomputers to perform a limited set of functions) are included in the design, CRTs are usually controlled by the central computer and the sensors communicate directly with the distributed computers.

Definitions of distributed systems abound (in reference [DOBR81] twenty-one definitions are given) but most of them are based in the three components described above.

Advantages and disadvantages of DS have been described elsewhere [AKOK78]. The advantages applicable to the case presented in this Thesis are:

- rapid response to local needs
- objective oriented
- reduced overall system complexity
- low start-up costs
- low incremental expansion costs
- high performance/cost ratio (operations per second per total cost)
- high benefits/cost ratio (total benefits per total cost)
- prolonged life
- easier maintenance
- easily expandable to more hardware and functions
- forces modular programming
- smaller programs
- esier to debug and maintain
- less complexity in documentation
- less specialized support
- components (also software) can be provided by many vendors

- easier testing and self diagnostics (at low levels)
- newer hardware technology on the average
- enhanced signal integrity (digital signals less corrupted by noise and

  temperature effects)

However, some disadvantages should also be noted:

- idle resources
- complexity of the reconfiguration process
- complex routing of data
- difficult validation

In the last years, efforts have been made to integrate the idea of fault tolerance within the framework of systematic design to imply a conceptual reverse of the traditional 'ad hoc' methods for the design of distributed computer systems. According to traditional methodologies designs are made by thinking of the functions the system has to perform during more than 99% of its life, when it is working in its primary dedicated functions (analyzing measurements, controlling actuators, etc). Instead, the FT approach is based on a systematic way of designing (that does not weight more heavily those primary dedicated functions) but it concentrates efforts on 'robustifying' characteristics that will allow the system to spend even more time in its primary functions. This can be accomplished by defining threats and malfunctions, creating ways of detecting them, identifying the failed units and reconfigurying the system or allowing a 'graceful' degradation to a state with less computational requirements. These functions have not been appropriately considered in the past until it was too late and modifications were almost prohibitive.

In one way or another, some FT methods have been applied since the first computers were built in the 1950's. These early FT methods essentially consisted of providing redundant hardware. Later, more reliable components were used, diagnostic tests were employed, quality assurance and system tests were introduced at the manufacturer's level and finally structured programming, methodologies for requirements, specifications and validation made the important contributions of 'division of the problem into manageable parts' and 'systematization of the design'.

In order to understand the idea of FT the concepts of threat, malfunction and consequence are introduced, (based on [HOPK80]): a <u>threat</u> is a stress producing an anomalous condition or <u>malfunction</u>, which has as a <u>consequence</u>: a modification in the normal behavior of the system.

A system may be designed to prevent threats from producing malfunctions, and/or it may cope directly with the malfunctions themselves. The objective is that the consequences to the jobs being performed by the system are to be minimal. Threats and malfunctions depend on the implementation.

Three categories of threats can be identified:

- limited environmental stress (ie aging) causing components to lose their original characteristics over a long period of time.
- abnormal environmental stress (high radiation dose rate) that disrupt components in a short time, and
- design errors in the system.

Now the definitions of failure, error and fault will be introduced. They all are malfunctions, which can be divided into four levels:

- physical malfunctions (component <u>failures</u>)
- signal-level malfunctions (<u>faults</u>)
- data-level malfunctions (<u>errors</u>)
- system-level malfunctions (catastrophic system failures)

The idea behind a FT design is that (reference [HOPK80]):

```
* some faults can be tolerated all the time or all the *
* faults can be tolerated some of the time, but not all *
* the faults all the time.                              *
```

Summarizing, threat induced malfunctions and their consequences are to be contained by FT features.

FT methods can be broadly divided into two complementary branches: fault-intolerance and fault-tolerant methods [AVIZ71]. The total resources allocated to achieve the required reliability must be divided between these two methods.

Fault-intolerance is an attempt to eliminate the causes of unreliability before the system starts the computing process in its actual application. This is accomplished by

- using reliable components (low failure rate)
- providing good packaging (low influence of the environment)
- testing components, boards and systems (during design and after installation)
- developing modular software
- handling correct and updated documentation
- devising maintenance procedures according to the requirements given by the user (ie procedures may be different for high or low reliability applications)

On the other hand, fault-tolerance is defined as the capability of the system to overcome malfunctions without human intervention by means of protective redundancy. This redundancy can be achieved by

- additional hardware or hardware redundancy
- additional software or software redundancy
- repeating operations or time redundancy

Hardware redundancy can be divided in two categories: static and dynamic redundancy. Static redundancy techniques, also called fault-masking or parallel-redundancy employ multiple, identical components (processors, memories, buses, switches, etc), operating simultaneously in parallel.

The advantages of this approach are that

- it is simple,
- detects a high proportion of transient and permanent faults (at the signal level),
- provides instantaneous fault masking containing the propagation of its effects to higher levels,
- it is easy to design (just include standard voting circuitry) and
- is becoming less expensive due to reductions in hardware costs.

However, the technique is not free of disadvantages. They include

- increasing number of interconnections and addition of extra hardware (and/or gates to provide voting) that could effectively increase the failure rate of the unit,

- non applicability to the highest levels of the design (too costly to replicate everything),
- not useful for common-mode failures in:
  - software
  - power
  - clocking
  - synchronization
  - buses
  - protocols
  - design errors
  - manufacturing process
  - uncharacterized environmental stresses.

In the dynamic redundancy approach, after a threat occurs, a) a malfunction is detected, b) a real-time recovery process begins to identify the failure and/or failed unit, and, if necessary, c) replace it by a spare or d) degrade the system by cancelling processing of incorrect or suspected functions.

The advantages of dynamic redundancy are that:

- there is a good modularization of the design,
- the system survives as long as there are enough spares,

- the effect of transient errors are eliminated,
- adjustability of the number of spares to the particular application can be made in the case that modifications are required,
- diagnostic programs can check spare parts and assure that they are error free when they are needed.

The main disadvantages with these more sophisticated forms of redundancy are that they are more difficult to design and to validate.

The use of <u>software redundancy</u> is made by including programs that provide fault-detection or recovery at various levels in the computer system. Three major forms are

- multiple storage of critical programs and data,
- test and diagnostic programs at various levels in the hierarchy of the design and at the various phases of the systems' life-cycle and
- restarts of the executive.

The advantages of this feature are that

- it can be provided after the hardware has been designed if it is seen that more protection is required and

- it can be easily modified.

The main disadvantages are, as always, the difficulties of assuring completeness and the validation of the design. However, any effort made in this direction is useful.

Time redundancy is based in repeating or acknowledging programs after a malfunction has been detected. Two techniques are

- repeated execution or acknowledgment of a module of a program (microinstruction, macro, subroutine) and
- restart of a program with an uncontaminated set of variables after reconfiguration has been made (rollback).

This category of redundancy is employed with the other two types previously discussed. Real-time constraints tend to limit the application of time redundancy.

In a FT design, a combination of hardware, software and time redundancy are employed.

2.4 A PRELIMINARY DESIGN

Based on the requirements detailed in FIG.2.2 and knowing the characteristics of the power plant in which the system is to be installed, a preliminary design can be drawn. The final preliminary design is shown in FIG.2.3.

The simultaneous analysis of the requirements will provide a preliminary set of the characteristics of the system that will be designed.

The first characteristic, as explained in Section 2.1, is derived from the requirements 2, 5 and 6 and it is the necessity of a _computer system_. An analysis of the rest of the requirements will provide more characteristics of that system.

The second characteristic, obtained from requirements 1, 2 and 10 (namely that the tasks performed by the system have to be in real-time and that the total cost of the system must not be very high) suggest that the candidate architecture should be _distributed_. Otherwise, a centralized system with such real-time capabilities could be very expensive (ie array processors)** for this application. On the other hand, a distributed system based on powerful but not expensive processors could be configured in a

---

** The use of array processors could probably be justified for particular applications: for example, on-line spectrum analysis. However, one of the drawbacks of array processors is that their reliability is, in general, extremely low (the other important drawback being its high cost). Some facilities that use array processors have reported mean time between failures of the order of a day.

process-oriented structure, taking advantage of the modularization. In that way, requirement 7 (dealing with changeability and expandability) could also be satisfied. One way of achieving that process-oriented modularization would be to use <u>off-the-shelf</u> components.

The third important characteristic at the system level is implied by an analysis of the survivability requirements 8 and 9. According to Section 2.3 a careful consideration to two <u>fault</u> <u>tolerant</u> techniques must be given: fault intolerance and fault tolerance. Fault intolerance tries to prevent threats from originating malfunctions. On the other hand, fault tolerance is related to the concept of malfunction containment: the design is oriented to detect and contain malfunctions and consequences within the levels in which they were originated; ie, avoiding propagation of the consequences of a malfunction. Both of these concepts (fault intolerance and fault tolerance) mean extra cost on any design by increasing the amount and quality of hardware, software, design and test, effort, maintenance, etc.

The first tradeoff appears when confronting the results from the previous paragraphs: off-the-shelf components with stringent survivability characteristics. Normally the acceptance of one solution will preclude the other because most of high level off-the-shelf components (boards, processors, systems) cannot satisfy fault tolerance by themselves (ie as provided by the manufacturer) and considerable effort should be dedicated by the designer, first, to develop tests and procedures to check some parameters that are loosely specified by the manufacturer and, second, to incorporate some fault tolerance in the design. In some ocassions, it may be advisable to use original designs for special parts of the system.

If the system is divided into levels, there are a number of techniques applicable to each particular level (those underlined were used in this preliminary design).

Computerized control systems can be divided into four levels (component, module, subsystem and system level), connected to two environments (sensors and operators or other computers). The sensors interface with the computer system at the source layer* and operators interface with the system by means of the central computer (central node*) or computers connected to it.

Decisions related to the sensors involve the number of

---

* Note the terminology that it is used in this and in the following Chapter (FIG.3.3): sensors are connected to multiplexers that are at the 'source layer', the outputs of multiplexers are sent to concentrators that are at the 'concentrator layer' and concentrators are connected among themselves (creating an effective network) and to a central computer located in the 'central node'.

sensors that have to be connected (ie redundant or not) and the quality of the signals they will be sending to the multiplexers. If the computer system has to be installed in a plant that is in the process of being built, these points can be discussed by the designer; if the computer system has to be integrated into an existing plant, the amount and quality of the original sensors will almost be untouched. However, an opportunity to interface additional sensors to the computer is rarely presented.

With respect to the computer system, at the lowest level (gates, registers, connectors) redundancy of components may become very awkward due to many interconnections, and very costly because there are many units. At this level, two approaches seem useful: use of high quality components to reduce failure rates and coding (in its multiple versions of error detection (ED) or error detection and correction (EDC)). However, the use of high quality components increases cost and, with respect to coding, the more bits are used the more complicated and slower the uncoding schemes become and more useful bits (of data) are used. References [PETE72] and [PRAD80] treat this topic at length. In the design presented <u>double-error-detection</u> <u>and</u> <u>single-error-correction</u> are employed, because it handles transient errors, it is not as slow as more complicated schemes and provides automatic correction of single failures.

At the next level, the components are interconnected to form boards, power supplies, etc. A list of the errors and schemes to cope with them at this level can be seen in the IBM Fault Tolerant Network Study [IBMF80]. The typical fault tolerant scheme for high reliability short-life applications has been to replicate components and to provide a voter (2 out of 3, 2 out of 4) to compare their outputs and reject those that fail. Although at first sight this seems correct, a closer examination will prove that the scheme is only useful when each pattern of failures is expected not more than once in the lifetime of the system (or at least until the failed element is replaced) or these failures do not occur symultaneously in more than one component. Otherwise, (ie in a 2 out of 3 structure), the outputs of two failed units would be preferred instead of the healthy one. Another problem, constantly overlooked, is the failure rate of the voter itself, which may preclude (at least) three healthy and more expensive components of delivering their outputs when needed. The risk in using a voting structure is very high. A solution to this problem is to include <u>reconfiguration and recovery</u>.

Next, at the subsystem level (concentrators, peripherals), the idea is to use available (<u>off-the-shelf</u>) components with critical design parameters well characterized (reliability, maintainability, spares availability, compatibility of equipment) and with some kind of redundancy and implemented <u>malfunction detection, identification and reconfiguration</u>. The kind of redundancy will depend on the application. For modest applications requiring low reliability, one unit working and one spare (powered or not) or two units working in parallel with a voter may be enough. In these designs, upon detection of a malfunction, a signal is sent to the operator's console and/or a light is activated on the board itself. Sometime afterwards, the board is manually replaced by another board which is supposed to work as soon as it is connected. Many criticisms can be made of structures that use that kind of redundancy from the standpoint of high reliability applications and they are:

- costly and ineffective use of the hardware (ie in the double board configuration, whenever a difference in the outputs of a voter are noted, both boards are replaced, even when, more often than not, only one of them would be affected).
- these designs depend on the performance of a hardwired voter, having the same drawbacks that were noted before.
- it is usually assumed that the component that replaces the failed one is in perfect operating condition although there usually is no way of knowing if that is the case without performing some tests on it. It is interesting to note that every testing scheme applied by the manufacturer can only

find a percentage of the total number of failures the board has, in the same way that commercial software is commonly released with 'bugs'. Therefore, the assumption of perfect spares may not always be correct.

- replacement of the failed unit by a good one is not made in a short time (less than one second) but the delay may leave the system with unadequate protection for long periods of time (hours).

One solution to the problem of redundancy at this level is to employ more (than two) <u>parallel redundancy with voter implemented in software and reconfiguration</u>. The minimum number of working units has to be three to detect <u>single</u> failures unequivocally. This configuration (three processors working simultaneously with the same inputs and producing the same outputs) is called a triad. The selection of the actual number of units (three, four or more) will be done in Chapter 6, where the reliability and availability of the units is modelled and analyzed.

However, the use of techniques that have a certain amount of decision (switching spares on) must assure that those decisions are taken by a non-failed unit. The use of <u>testing</u> will guarantee, within certain limits, that the units work correctly (at least for those tests). In reference [HAYE80], testing methods are divided into two broad categories: concurrent (or implicit) and explicit. Concurrent approaches include mostly coding. Explicit methods can be applied by a processor to itself (_self-tests_) or to another unit (_external testing_). A more complete and updated list of testing methods is given by Williams et al in reference [WILL82].

Note that even with these techniques: coding, self-tests, external tests, redundancy and reconfiguration the spectre of some common-caused malfunctions still exists* (ie software errors, design errors). The problem of synchronization has been solved by Daly et al in [DALY73] with their design of a fault tolerant clock.

The use of redundant software has been traditionally rejected for two reasons: cost and voting only on the outputs. Using only one _validated_ version of the software and voting it every few instructions or every instruction, bit by bit, may allow error discovery at the instruction (or subroutine) level. Voting only the final results of a program does not provide information about its structure, in case of problems. See reference [HOPK80].

Finally, at the network level, two major components remain to be addressed: links and duplication of central computers. With respect to the links, at least one design exists that considered the fault tolerance of a network with 'intelligent' nodes [SMIT75]. In this way, reconfiguration and recovery at the highest level is accomplished by rerouting data. Each node is interconnected to at least three other nodes (this will be specified in the network layout design, Chapter 3). Also, each node contains the necessary circuitry to be connected to other nodes. Thus, some links and nodes can remain idle and, in the event of a reconfiguration, they could be used. Note that this concept differs from standard designs with modest reliability requirements where the links are 'hardwired' in the design (ie there is no possibility of changing the established routing of data). One approach that may be used to build these nodes can be extracted from LEVASSEUR's design [LEVA79] of a serial and parallel interface circuitry, programmable and reconfigurable.

The double central computer architecture is not considered if the _central computer is already fault tolerant_ (and by that it is meant a computer designed to guarantee a minimum of reliability). The reasons behind the rejection

----

* The term common-caused is preferred to common-mode because what is common is the nature of the malfunction and not the way its consequences are propagated. Common-mode malfunctions can be detected but their origin can not be diagnosed easily. Common-caused malfunctions can not be reliably detected and their diagnosis is even more difficult.

of an architecture with double central computer is that they are useful only for limited reliability requirements. Control systems with these computers have not been connected to safety systems and requirement 7 indicates the need to provide control without precluding these feature. An alternate approach to the acquisition of a central fault tolerant computer is to build one with rather standard components, following a systematic method of design, testing and validation. However, the magnitude of the effort will be comparable to that of designing the rest of the network and, consequently, this approach is not recommended.

The requirements that remain to be analized, 3 and 4, characterize the functions that the system will have to perform with respect to the lower layers (sources and concentrators) and the final destination of the outputs produced by the system.

With signals acquired from the sensors, a first and limited amount of processing will be required. The signals will be compared to upper and lower limits or other kinds of thresholds as indicated in Chapter 5. At the same time other checking will be made with signals coming from redundant sensors (if they exist) measuring the same quantity, or from data derived from dissimilar sensors combined in appropriate fashion. One more test can be made by using analytic models to calculate signals similar to those of the processes that are being sensed, providing a rather independent way of verifying the measurements. This procedure is called _analytic redundancy_.

The first elemental processing: comparison to limits and another sensors or data, can be made with microcomputers installed in the same cabinets where multiplexers are located. Sensors are hardwired to multiplexers, multiplexers write data in memories and processors take data from the memory, make the calculations and send the results (fewer data than sending every sensed variable) through fiber optics cables to the concentrators, located in nodes distributed througout the plant. In the concentrator nodes, the main computations will be related to the modelling of the processes (Chapter 4), Hypothesis Testing algorithms (Chapter 5), normal management of data (to another nodes and to the central computer for keeping records) and abnormal procedures (alarms created as a result of a mismatch in the comparisons made at the source and concentrator layers).

The central computer is in charge of providing the means for operators and other computers to access the files and the rest of the system. Files have to be accessed by the minicomputers connected with the central computer to generate the displays required by the operators. Computers can be linked with the system, for example, to provide immediate maintenance (this possibility may be offered by the designer of the system) or to provide or request data routinely (process computer, other user's computer systems).

A summary of the processing of signals and their conversion to data in the system says that the lower layers will filter, smooth and synthesize signals so that they can provide reliable data to be used higher in the hierarchy of the system. System routines running in the central computer and in the utility computer will be used to assess the safety and productivity of the plant as intended in the original user's goal. As an example of the use of this overall management of information see [MOTO77].

Finally, a review of the distributed system must be made to look for the overall reliability. If certain multiplexers and links are found to be weak points in the design, redundant multiplexers and links have to be provided and, due to the same considerations that dictated triplicated processors at the concentrator layer, they will also have to be triplicated. Note, however, that links that connect concentrators among themselves do not need replication because of the existance of the fault tolerant network.

2.5 CONCLUSIONS

In this Chapter, a methodology for the design of computer systems has been introduced and it has been applied to the preliminary design of a computer system to monitor a nuclear power plant.

The idea behind the method is to present a list of requirements: the characteristics of the system can be extracted from the requirements, and not viceversa. The process is iterative and involves three organizations: the user, a supervisor and the designer, each one with different purposes and structures.

Definitions of different concepts used throughout the Thesis have been given: objective, goal, requirement, specification, distributed system, fault tolerance, threat, malfuction, failure, fault and error.

The proposed system is a distributed computer system, which makes use of fault tolerant techniques to achieve the desired degree of reliability (which consists in ellimination of single failures) and which does not rely in brut force redundancy (which was shown that it can not cope with every malfunction). The structure of the system and the functions to be performed have also been delineated as a consequence of the analysis of the requirements.

BIBLIOGRAPHY FOR CHAPTER 2

[AKOK78] AKOKA, J., Design issues in distributed management information systems, PhD Thesis, MIT, 1978

[AVIZ71] AVIZIENIS, A., Architecture of fault tolerant computing systems, Digest of papers, 1975 International Symposium on FT Computing, Paris, June 1975

[DALY73] DALY, W. et al, A fault tolerant digital clocking system, IEEE Computer Society, Dig. 2nd Int. Symp. on Fault Tolerant Computing, June 1973

[DEUT81] DEUTCH, M., Software project verification and validation, Computer, April 1981

[DOBR81] DOBROWOLSKI, M., Guide to selecting distributed management information systems, InTech, June 1981

[GOLD79] GOLDBERG, J., Formal techniques for fault-tolerance in distributed data processing, AD-A071030, April 1979

[HAYE80] HAYES, J. et al, Testability considerations in microprocessor-based design, Computer, March 1980

[HENI80] HENINGERG, A., Specifying software requirements for complex systems: new techniques and their application, IEEE Trans. on Software Engineering, January 1980

[HOPK80] HOPKINS, A., Fault-tolerant system design: broad brush and fine print, Computer, March 1980

[IBMF80] IBM Federal Systems Division, Fault Tolerant Computer Network Study, Progress Report 1979-1980, AD-A088066, April 1980

[LEVE79] LEVASSEUR, D., Simplify IEEE-488 implementation with a multifunction interface, Electronic Design News, March 5, 1979

[LEVI81] LEVITT, K., Software validation and verification techniques, SRI International, Menlo Park, CA, 1981

[MOTO77] MOTODA, H. et al, Feasibility studies of core management system by data communication for BWRs, Nuclear Technology, Mid December 1977

[MUNS81] MUNSON, J., Software maintainability: a practical concern for life-cycle costs, Computer, November 1981

[PETE72] PETERSON, W., Error correcting codes, MIT Press, Cambridge, MA, 1972

[PRAD80] PRADHAN, D., Error correcting codes and self-checking circuits, Computer, March 1980

[ROSE81] ROSE, C. et al, Systematic reliability design for computer control systems, Case Western Reserve University, 1981

[SMIT75] SMITH, T., A damage and fault tolerant I/O network, IEEE Trans. Comp., May, 1975

[TURN77] TURN, R. et al, A management approach to the development of computer-based systems, The Rand Corporation, 1977

[WEBS77] WEBSTER'S New Collegiate Dictionary, G. and C. Merriam Publishing Co., Springfield, MA, 1977

[WEIT80] WEITZMAN, C., Distributed micro-mini computer systems, Prentice-Hall, 1980

[WILL82] WILLIAMS, T. et al, Design for testability-A survey, IEEE Trans. on Computers, January 1982

[WOLF77] WOLF, B. de et al, Methodology for requirements specification and preliminary design of real time systems, Charles Stark Draper Laboratory, Report C-4923, 1977

PHASE	ACTIVITIES	FEEDBACK	PRIORITY OF SUPERVISION		
*****	**********	********	USER	SUPERVISOR	DESIGNER
Requirements definition	(1) GOAL DEFINITION		1st		
	(2) REQUIREMENTS DEFINITION		1st	2nd	
Specifications definition	(3) OBJECTIVE'S DEFINITION	(2)	2nd	1st	
	(4) SPECIFICATIONS	(2)	2nd	1st	
Design	(5) PRELIMINARY DESIGN	(3)		2nd	1st
	OBJECTIVE'S DEFINITION				1st
	PROBLEM DEFINITION				1st
	PROCESS INTERACTION				1st
	(6) DETAILED DESIGN	(5)	3rd	2nd	1st
	(7) DEVELOPMENT	(3, 5)			1st
Testing	(8) SUPERVISOR GROUP TESTING	(7)		1st	
	(9) USER TESTING	(7)	1st		
Production and Installation	(10) PRODUCTION				1st
	(11) INSTALLATION	(7)		2nd	1st
Operational use and Maintenance	(12) USE	(7)	1st		
	(13) MAINTENANCE	(7)			1st
Modifications	(14) NEW GOALS	(1)	1st		
	(15) NEW REQUIREMENTS	(2)	1st		
	(16) NEW TECHNOLOGIES	(3)	1st		

FIG.2.1

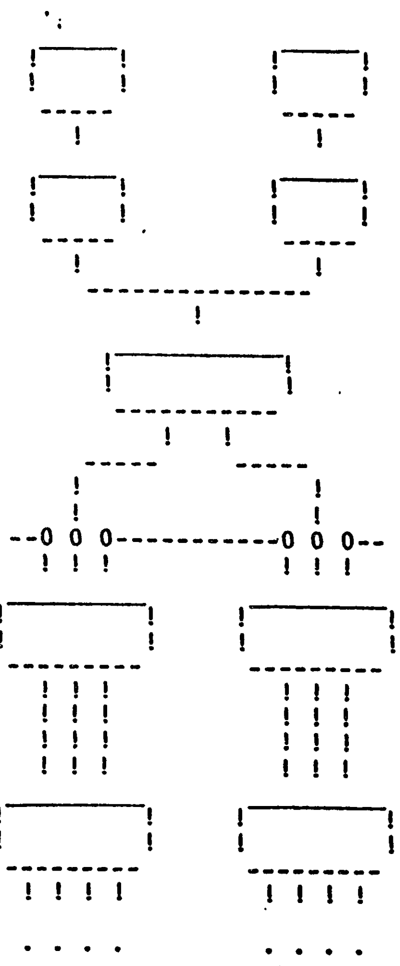
OBJECTIVE AND REQUIREMENTS

APPLICABLE TO THE MONITORING SYSTEM

OBJECTIVE	A nuclear power plant monitoring system
	REQUIREMENTS
1	A monitoring system is required for a nuclear power plant.
2	It has to interface with sensors distributed throughout the plant and sample, process, store and display data.
3	Some variables have to be compared with predetermined limits and/or with models of some of the subsystems of the plant.
4	Given anomalies in the comparison (item 3), some sensor data may be rejected or the operators may be alerted, or both.
5	Operators may ask for past, present or future values of some variables, design or operational data from the plant, monitoring system status (what elements are operational) and present maintenance data of the plant.
6	The system has to interface with computer systems (process computers, utility computer)
7	It must allow expansions to incorporate more sensors, models and functions, (including control of some of the subsystems by means of actuators).
8	Single failures must neither disable all of system's functions nor impact adversely the operation of the plant.
9	It must be readily modifiable to allow the incorporation of new technologies and improved algorithms.
10	It must be easily maintainable (except in the case of catastrophic incidents).

FIG.2.2

STRUCTURE OF THE INTEGRATED CONTROL SYSTEM
****



ELEMENTS ********	FUNCTIONS and CHARACTERISTICS ***************
OPERATORS	Operate the plant
CRTs OTHER COMPUTERS	Gather and present information
FAULT TOLERANT CENTRAL COMPUTER	Supervision System reconfiguration Communications control Display control· Central memory manag.
FAULT TOLERANT NETWORK	Fiber optics Reconfigurable network
CONCENTRATOR LAYER	Powerful microcomputers Analytic redundancy Alarms Control algorithms
LINKS	Fiber optics Triplicated
SOURCE LAYER	Microprocessors Signal validation Hypothesis Testing
SENSORS	Provide signals

FIG.2.3

CHAPTER 3

THE NETWORK DESIGN

In this Chapter one of the basic problems in the system's preliminary design will be addressed: the connection of signals distributed throughout the plant to a central computer. This question is also referred to as the layout problem.

The problem will be discussed, formulated and solved for the two scenarios postulated in Chapter 2: an initial one when the main function of the computer system will be those of surveillance and a second one, ocurring later in the life of the system, when some control will be allowed.

The objective is to minimize the installation cost of the network.

3.1 THE DISTRIBUTION AND CHARACTERISTICS OF SIGNALS IN THE PLANT

Given that the arrangement of signals in a plant changes from station to station (and from year to year), it is necessary to use an idealized plant to show how to find the optimal solution to the layout problem. For that reason a typical Westinghouse PWR - plant will be used, using the data provided in reference [UNIT76]. The arrangement can be seen in FIG.3.1.

The plant consists of a reactor building (RB), turbine building (TB), emergency feedwater pump building (EFPB), primary auxiliary building (PAB), waste treatment building (WTB), fuel handling building (FHB), cooling tower area (CTA), intake structure (IS) and yard area (YA).

The RB includes the reactor containment with 4 primary loops and their steam generators, reactor cooling pumps, pressurizer, piping and the safety injection system.

The TB houses the turbine generator, condensers, pumps and feedwater heaters.

The EFPB contains the auxiliary feedwater pumps, auxiliary feedwater control valves and the demineralized water pumps.

The PAB houses most of the auxiliary systems for the reactor coolant system, which are the chemical and volume, low pressure safety injection, residual heat removal and containment spray systems.

The WTB contains the liquid and gas waste processing, boron recovery and solid waste systems.

The FHB houses the underwater fuel storage facilities.

With respect to the signals themselves they are divided into four main categories:

. building	
. scenario:	initial
	growth
. function:	non-safety
	safety
	monitoring (non-control)
	control
. electrical characteristics:	analog
	digital

A summary of the tabulated data in [UNIT76] and applicable to our case is presented in FIG.3.2. Note the following:

- the non-control signals are used for indication, alarm and recording (IRA).

- signals classified into the 'initial scenario' subcategory are those that can be found in the non-safety and safety non-control subcategories; more explicitly, every signal except the ones used for control. The difference between the 'initial' and 'growth' scenarios is the inclusion of the control signals.
- the control signals can be thought to be composed of two groups: those that go from the field to the computers and the commands of the computers responding to signals received from the plant.
- to make the problem understandable two important simplifications have been made. First, a distinction is not made for 'electrical characteristics', although, in real life, the I/O cards of every multiplexer have a fixed amount of analog and digital lines, condition that could be incorporated as one more set of constraints in the following Section. Second, no distinction has been made in signals that go from the field to the computer and viceversa, although, again, there are input/output limitations in doing that.
- an extra 20% of signals per building is included in the last column of FIG.3.2. That is used in Section 3.2 to account for limited expansions.

3.2 THE CONNECTION PROBLEM

Given a multiplicity of sensors, computers and other data sources spread over a certain area and some measures that characterize the network and the traffic expected between the various sources, the desire is to optimize a function based on those measures. The most general solution (if it exists) will provide the optimal network.

Some of these measures could be delay times (considering or not the possibility of reconfiguration in the network), buffer sizes, installation costs, link capacity, connectivity, number of sources connected to one concentrator, reliability, etc. The optimization could imply the minimization or maximization of one, some or all of the measures considered.

The difficulty in deriving a solution will depend on the size of the network (number of nodes or links), the number of the measures to be optimized and the constraints imposed on the system. The complexity of the problem will determine whether an optimal or suboptimal solution is more appropriate and will also suggest an analytical, Monte Carlo or heuristic method. For complex networks the optimization procedure is iterative and more than one method is used.

Surveys of methods for centralized and distributed networks design are given in [FRAN72] and [SCHW77].

In this section the formulation of the connection problem will be presented. The structure of the network will consist of sources distributed throughout the plant and connected to concentrators, which in turn are connected among themselves and to a central computer. The sensors are assumed to be hardwired to the source layer (ie the multiplexers). Their location in the plant does not concern to the problem at hand. Their inclusion would provide unnecessary detail at a very low level. If changes would be made at that low level, no modifications would be noted at higher levels of the design. The sources are assumed to consist primarily of multiplexers and those processors that perform the data validation. The concentrators consist of more powerful microcomputers in charge of higher level functions (with respect to the sources) such as modelling, Kalman Filtering, comparisons between the filtered signals from the sources and the variables calculated in this layer, etc.

The sources and concentrators are located in nodes, which will be called distributed nodes, to distinguish them from the central computer, located in the central node. This same structure will be used in Chapter 6 for the analysis of the reliability of the network.

Summarizing the distribution and nomenclature of the elements in the plant, it can be said that the objective of this layout problem is to minimize the capital cost of a distributed system consisting of sources of signals, connected to distributed nodes which are connected among themselves and to a central node.

Assume that the sources, distributed nodes and central node are located in concentric layers, displayed in FIG.3.3. The outer layer contains m sources of signals (groups of sensors representing the number of signals per building as displayed in FIG.3.2, not isolated sensors) indexed with the letter i, $(1 < i < m)$; in the following layer, the n groups of concentrators are indexed with the letter j, $(1 < j < n)$; the innermost layer contains the central node.

The links' symbology designate the connection of layers with letters and the connection of nodes with subindices. According to FIG.3.1 the nomenclature for the links is the following:

- wij = link between a source located at node i and a concentrator located at node j,
- xio = link between a source located at node i and the central node,
- yjl = link between a concentrator located at node j and another concentrator located at node 1,

- zjo = link between a concentrator located at node j and the central node.

The set of variables can only adopt two numerical values: 0 (there is no link) or 1 (a link exists).

The capacity of a multiplexer located at node i will be indicated with the symbol Pi. That capacity represents the maximum number of signals that can be connected to the multiplexer. Similarly, the maximum number of signals accepted by an analog to digital converter (ADC) located at node i will be indicated with Ai and the maximum number of digital signals accepted in a digital to analog converter (DAC) also located at node i is Di. For the problem treated in this Thesis, only one kind of multiplexers and converters will be used.

The number of signals produced per source is indicated with Si and the numerical values that this variable adopts were shown in the previous Section, under the heading 'Scenario' in FIG.3.2.

The objective function to be minimized is the total installation cost of the network. This includes the design, equipment and labor cost of the installed hardware. The cost of the software is not considered because it is assumed to be a constant for a distributed system *.

The variables to be found are the number of links and the number of concentrators.

The costs that weigh these variables have the same subindex nomenclature used for the nodes and the links. They are:

- cij = initial cost for link wij
- cio = initial cost for link xio
- cjl = initial cost for link yjl
- cjo = initial cost for link zjo

Consequently, the objective function to be minimized is

$$F = \min \left\{ \sum_{i} \sum_{j} cij * wij + \sum_{i} cio * xio + \sum \sum cjl * yjl + \sum cjo * yjo \right\} \qquad (3.1)$$

---

* Actually the software costs are quantized, in time and in size of the system. In TIME because with changes in the available technologies some functions may not be made anymore in software and they are provided installed in hardware (called firmware). In SIZE because systems are designed as to manage resources between defined limits. Above these limits, some parts of the software must be rewritten to allow the expansion, implying modifications in routing tables, reconfiguration procedures, priorities, protection, etc.

i j j

This objective function will be minimized subject to a set of constraints. They are:

- a minimum number of converters and multiplexers must be available to provide the interface with sensors. These constraints may be incorporated as one more set of constraints when they are solved by the computer program or they can be calculated by hand when the network is small enough. In the present problem they will be calculated by the hand method.

The constraints can be written as

$SAi' - Ak2 * NAi < 0$ for all i

$SDi' - Dk3 * NDi < 0$ for all i

$Si' - Pk1 * NMi < 0$ for all i

where

SAi' = (number of analog signals at node i) * 1.2

SDi' = (number of digital signals at node i) * 1.2

NAi = number of ADC boards at node i

NDi = number of DAC boards at node i

NMi = number of multiplexers at node i

- communication lines from the sources to the central node must exist. This constraint is indicated by

$$\sum_j w_{ij} + x_{io} = 1 \qquad \text{for all } i \qquad (3.2)$$

The assumption has been made in (3.2) that the source i will communicate with either a distributed node (link wij) or with the central node (link xio), but not both. Other constraints of a similar kind that could be enforced could be that the source i has to have two links to different concentrators and that can be indicated as

$$\sum_j w_{ij} = 2 \qquad \text{for all } i$$

or that the source i has to be linked with two or three concentrators and a maximum of one link to the central node, which can be written as

$$\sum_j w_{ij} + x_{io} = 3 \qquad \text{for all } i$$

In this Thesis the constraint given by equations (3.2) will be used.

- if there is no input to a node j, there should be no output to the central node. That is indicated by

$$\sum_{i} wij - zjo > 0 \qquad \text{for all } j \qquad (3.3)$$

Here, if there is no input to the node j, the sum of all wij's for all i's (links arriving to j) will be zero, and this will be the upper bound of the numerical value of the variable zjo (link from the node j to the central node). Given that the lower bound for all variables considered is zero, zjo is forced to be equal to zero.

- limits on the number of links between concentrators can be stated. A parameter TESTC will be used for that purpose. The maximum of TESTC occurs in the case of a fully connected network, where

$$TESTC = \sum_{i=1}^{n-1} i$$

and n is the index that corresponds to the last group of concentrators. For example, if n=4, TESTC=6; if n=5, TESTC=10, and so on. Then, the constraint can be written as

$$\sum_{j} \sum_{l} yjl = TESTC \qquad (3.4)$$

- there must be a minimum number of links from the concentrator layer to the central node. This constraint is related to reliability requirements. A very small number of links is not desirable because of the possibility of saturating the bandwidth of the link in case of reconfiguration (ie too many signals through one link), although the ever increasing capacity of the fiber-optic links almost removes this constraint for small networks. The constraint can be written:

$$\sum_{j} zjo = TESTD \qquad (3.5)$$

There must be a correspondence between the number of nodes in the concentrator layer (TESTC) and the maximum number of links between nodes (TESTD). For example, if TESTC=4, then TESTD=6; if TESTC=5, then TESTD=10, and so on.

Actually, TESTC is used as a variable parameter in the design of the network and it can be thought as a special case of the more general formulation

$$TESTC1 < \sum_{j} \sum_{l} yjl < TESTC2$$

where TESTC=TESTC1=TESTC2, and TESTC1 and TESTC2 can be understood as being the minimum and maximum number of nodes that can be assumed for the network, respectively.

Equation (3.5) is a special case of the more general case where the equal sign is replaced by a less than or equal sign.

- concentrators have a limited processing capability. This set of constraints, also related to reliability considerations, effectively limits the maximum number of sources to which a concentrator could be connected in order to prevent saturation of 'healthy' nodes in the case of reconfiguration. If such a maximum is denoted by Sj, then

$$\sum_{i} Si' * wij < Sj \qquad \text{for all } j \qquad (3.6)$$

- the capacity of direct hook-up of the sources to the central node is limited. Process computers only accept a very limited number of sources directly connected to its input cards. This constraint is expressed as

$$\sum_{i} Si * xio < So \qquad (3.7)$$

As it is formulated, the problem at hand is one that requires an integer programming solution. All of the variables are of the 0-1 type.

The constraints can be written in the following way:

GROUP A = $\sum_{j} wij + xio = 1$
****

$$(i=1) \quad w11 + w12 + \dots + w1n + x1o = 1$$

$$(i=2) \quad w21 + w22 + \dots + w2n + x2o = 1$$

$$\vdots$$

$$(i=m) \quad wm1 + wm2 + \dots + wmn + xmo = 1$$

GROUP B = $\sum_{i} wij - zjo > 0$
****

$$(j=1) \quad w11 + w21 + \dots + wm1 - z1o > 0$$

$$(j=2) \quad w12 + w22 + \dots + wm2 - z2o > 0$$

$$\vdots$$

(j=n) $w1n + w2n + \ldots + wmn - zno > 0$

GROUP C $= \sum_j \sum_l yjl = TESTC$

****

$$y12 + y13 + y14 + \ldots + y1n + \\ + y23 + y24 + \ldots + y2n + \\ + y34 + \ldots + y3n + \\ \ldots + y(n-1)n = TESTC$$

GROUP D $= \sum_j zjo = TESTD$

****

$$z1o + z2o + \ldots + zno = TESTD$$

GROUP E $= \sum_i Si' \, wij < Sj$

****

(j=1) $S1' \, w11 + S2' \, w21 + \ldots + Sm' \, wm1 < S^1$

(j=2) $S1' \, w12 + S2' \, w22 + \ldots + Sm' \, wm2 < S^2$

. . .

$$\begin{array}{ccc} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{array}$$

$$(j=n) \quad S1' \, w1n + S2' \, w2n + \ldots + Sm' \, wmn < S$$

GROUP F $= \sum_{i} xio < So$
****

$$S1 \, x1o + S2 \, x2o + \ldots + Sm \, xmo < So$$

Having introduced the objective function (3.1) and the constraints (3.2) to (3.7), the next step is to apply the equations to our particular problem.

3.3 COSTS INCLUDED IN THIS PROBLEM

In this Section the installation problem will be addressed. Their costs will be applicable to the distributed nodes (source and concentrator nodes) and all the links in the plant. It will be assumed that the cost of the central computer does not depend on the particular distribution that the other two layers adopt.

The costs will include design, equipment and labor.

At the lower level the sensors are interfaced with the use of converters (ADC and DAC) and multiplexers. For the purpose of finding an approximate total cost it is assumed that converters accept 32 signals per board and multiplexers accept 64 lines per board. With these capacities at hand, the detail of units per building and the total number of these interfacing units for the whole plant are given in FIG.3.4. Total costs can be derived by using these figures. Assume that the average cost of these ADC, DAC and multiplexers is $ 1000 dollars per unit (triplicated all of them in the m nodes of the source layer), that they have to be packaged into a chassis (approximately $ 5000 dollars per chassis with triplicated power supplies) and that there will be m of those chassis, the total cost of the hardware in the source layer becomes

3 * m * 1000 * (total number of ADCs, DACs and multiplexers as given by FIG.3.4) + + 5000*m [dollars]

The total hardware cost at the concentrator layer will be the cost one of the replicated (at least three) processors times the number of nodes (which has been called n). Taking as a reference the cost of a triplicated AUGUST SYSTEMS** control computer to be $ 50000 dollars (including expanded non-volatile memory and triplicated power supplies), the total cost of the hardware of the concentrator layer will be 50000*n [dollars].

For the cases at hand and applying the above equations, the hardware cost of the source + concentrator layer is $650000 for the INITIAL scenario and $927000 for the GROWTH scenario. Assuming that the cost of integration and test is 15% of the total hardware cost, the partial cost of hardware, with their installation and testing, becomes around $750000 and $1070000 dollars, for the INITIAL and GROWTH scenarios respectively.

With respect to links, it is assumed that

- links between the sources and the multiplexers (not optimized in this problem) are twisted pair cables installed at a total cost of

  $$536.383 + .552 * dab \quad \left[\frac{\text{dollars}}{\text{signal}} \right]$$

  where dab = distance [feet] from a to b. These costs are taken from reference [UNIT76], and include the cable, tray, soldering of terminals, labor and testing.
- The rest of the links (wij, xio, yjl, zjo) are fiber optic links. The cost (transmitter, fiber, receiver, installation and testing, from reference

---

* AUGUST SYSTEMS is a registered trademark of AUGUST SYSTEMS INC.

[IFOC79]) for a single link is

$$1770.00 + 3.255 * dab \quad [\text{ dollars }]$$

Note that, although the cost per signal is larger for a fiber optic link, it soon becomes cost effective when more than a few signals are sent through the same fiber, for a fixed distance dab.

In Chapter 2 it was shown that redundancy was to be used to replicate some components, so that the requirement of not allowing a single failure to isolate some nodes could be satisfied. Consequently, links corresponding to the variables wij and xio are triplicated. The rest (yjl and zjo) is not replicated due to the redundancy already created by using the variable TESTC.

In the following Sections, the cost of the configuration adopted by the links will be calculated and that cost will be added to the previous costs corresponding to the source and concentrator layers, to find the installation cost of the system (without considering the central computer and applications software).

3.4 THE SOLUTION TO THE CONNECTION PROBLEM

In a network with m sources and n concentrators, the total number of variables can be derived by noting that there are

$m \times n$	wij variables,
m	xio variables,
$n(n+1)/2$	yjl variables and
n	zjo variables.

Consequently, the total number of variables is

$$(n + 1) * \left(\frac{n}{2} + m \right)$$

With respect to the constraints, Group A provides m equations, Groups B and E give n equations each one, and the rest (Groups C, D and F), 1 equation per group. The total number of constraints then becomes

$$2n + m + 3$$

In a real life situation a number of possible locations for the concentrators would be identified by the user's and the designer's team and the main purpose of the integer programming solution would be to pick the subset of nodes and links that minimize the cost of the network subject to the corresponding set of constraints.

For example, the PWR considered in FIG.3.1 had at least 54 possible locations where sources of signals could be located (m=54). Considering that there could be 20 places where to install the concentrators (n=20), the number of variables goes up to 1344 and the number of constraints to 97. For some integer programming codes such a number of variables is prohibitive.

In the problem at hand and introduced in Section 3.1, the plant is divided into 9 areas (m=9). The possible concentrator locations are 5 (n=5) and consequently, TESTC=10 and TESTD=5 for a fully connected network of concentrators. Note that the more general network that includes the central node is also fully connected because TESTD has been assumed to be equal to the maximum possible value, n, indicating that there has to be one link going from the central node to every concentrator location. From the standpoint of redundancy of links at the concentrator layer, every node communicates with the other four, providing adequate redundancy at first sight.

From the preceding equations the total number of variables for this problem is 69 and the number of constraints is 22.

The distances between the different elements in which the plant has been decomposed are shown in FIG.3.5. Based on these distances and the considerations introduced in Section 3.2, the costs that weight the variables are calculated and shown in FIG.D.1 and the constraints are displayed in FIG.D.2 (Appendix A).

3.5 OPTIMAL AND SUBOPTIMAL SOLUTIONS TO THE CONNECTION PROBLEM

Computerized 0-1 integer programming (IP) solutions initially look for a linear programming (LP) solution. The LP solution, if it exists, is optimal in the sense that it minimizes (in our problem) an objective function and no other combination of variables will provide a lower numerical value for the objective function. The set of variables given by the LP solution is a set of real numbers. The final IP solution begins with the optimal solution produced by the LP procedure and it gives a (possibly) new solution. This IP solution, in general, will be suboptimal in the LP frame.

In the case presented in this Chapter, the procedure indicated above has been followed using the computer program SESAME, offered in the Information Processing Services (IPS) at MIT.

SESAME reads an input file of data specifying the problem and writes the results in output files. The input file and the output files are given in Appendix A. The results are discussed in this Chapter. The actual input file used in this Section is shown in FIG.A.3 and the outputs provided by SESAME are shown in FIG.A.4 to FIG.A.7. A pictorial summary is presented in FIG.3.6. The output file in FIG.A.6 corresponds to the input file FIG.A.3.

As recalled from Section 3.2, two situations have been envisioned in the life of the system: an initial scenario and a growth scenario, called 'INITIAL' and 'GROWTH', respectively, in the SESAME environment. These two situations have been combined with two more cases. Given that there are five possible locations for concentrators, the first case considers that only four of them will be linked to the central computer and the second case takes in account all of them. Consequently, these considerations give a total of four possibilities:

4 links, initial scenario ('INITIAL4' ---> FIG.A.4)
4 links, growth scenario ('GROWTH4' ---> FIG.A.5)
5 links, initial scenario ('INITIAL' ---> FIG.A.3 and FIG.A.6)
5 links, growth scenario ('GROWTH' ---> FIG.A.7)

The following are notes applicable to the results obtained (note that they are the LP solutions, not the IP solutions):

4 LINKS, INITIAL SCENARIO:

- total connection cost is calculated to be $ 104778.94 dollars (see below for a modification to this cost). The cost is given by SESAME once the SOLUTION procedure is called, and appears in the output in the column ..NAME..., row FUNCTIONAL. The numerical value is in the column ..ACTIVITY...

(see Appendix A).

- The variables appear in the section called COLUMNS SECTION. It can be seen that they all are equal to 0 or 1; then, given that they are integers already, there is no need to call the IP solution procedure.
- the total number of signals connected to a particular concentrator is obtained from the ROWS SECTION in rows corresponding to the Group E constraints (ie the total number of signals connected at concentrator number 2 is 3844, 41 of them coming from building 4 and 3803 from the reactor building) and the number of lines available for connection appears in the next column ...SLACK ACTIVITY....
- the link from concentrator 1 to concentrator 2, y12, has been considered a single fiber optic cable in Section 3.3. For reliability reasons, as explained in Chapter 2, it is convenient to replicate it. Note that this procedure does not have to be repeated for other concentrators because there are at least 3 links going to the rest of the concentrators or to the central node. When 3 links, instead of one, are considered, the cost of that path increases from $ 3235 to $ 9705, raising the total connection cost to $ 111248.94 dollars.

- the final cost assumed for the network will be the sum of the costs considered in Section 3.3 for the sources and the concentrators ($750000 dollars) and the cost found in this Section for the links ($111250 dollars), which totalize $861250 dollars.

4 LINKS, GROWTH SCENARIO:

- the total connection cost is the same as before ($ 111248.94 dollars). Remember that these are connection costs and the 'GROWTH' solution assumes that there will be no need to make any change in the original layout found in the 'INITIAL' solution if the capacity of the concentrators is raised, as planned, from 4096 lines to 5276 lines.
- however, now the LP solution has two values which are not integers. The reason for that being than the 5180 signals from the Reactor Building (building 5) cannot be totally connected to one concentrator. Instead, 97.14% of them are sent to concentrator 2 and the rest are connected to concentrator 3. Given that the difference is so small (only 148 signals out of 5180) and that the costs of reconnection (changing from the initial to the growth layout) will be higher than upgrading concentrator 2 to accept that difference, this last solution (ie upgrading concentrator 2) is preferred.

- the solution is suboptimal and arises from technological considerations.
- the modifications to the original solution are then a) the link y12 has to be triplicated and b) the concentrator 2 has to allow the connection of 148 more sensors.
- the total installation cost of this configuration is the sum of the costs corresponding to the source and concentrator layers (obtained from Section 3.3) $1070000 dollars and the cost of the links obtained in this Section, $111250 dollars, which make a total of $<u>1181250</u> dollars.

5 LINKS, BOTH SCENARIOS:

- both scenarios produce optimal LP solutions which also are optimal IP solutions.
- again, there is no difference in the total cost of connections, that total being $ 127627.00 dollars.
- in the 'INITIAL' scenario, building 9 is connected with the concentrator 4 (ie w94 = 1.0). In the 'GROWTH' scenario, the output of SESAME indicates that building 9 has to be connected to concentrator 3 (ie w94 = 0.0 and w95 = 1.0). However, given that the costs of connection between building 9 and each of the two connectors is the same, it is seen that the difference has been given by the procedure

followed by SESAME in the search for the optimal solution and, given also that the allowed total number of signals connected to a concentrator (4096 in the 'INITIAL' scenario and 5276 IN the 'GROWTH' scenario) is much higher than the total number of signals actually connected, it is concluded that the difference is irrelevant, there is no need to change the link w94 to the new position indicated (w95) and both layouts can be used interchangeably.

- the total installation cost for the network, as said before, will be the sum of the costs corresponding to the source and concentrator layers and the cost of the links. For the 'INITIAL' configuration the cost becomes 877630 dollars and the 'GROWTH' configuration will cost $1197630 dollars.

3.6 CONCLUSIONS

In this Chapter, a typical nuclear power plant has been chosen to exemplify how the connection of multiplexers (sources of signals to which the sensors are interfaced) can be accomplished through concentrators and how these elements can be interconnected to provide a network whose purpose is to communicate the concentrators to a central computer. This problem is referred to as the layout problem.

It was shown that the formulation of the layout problem can be posed as a 0-1 integer programming problem in which the total connection cost is minimized. That cost has been called 'connection cost' and is a component of the total 'installation cost', which includes also the cost of the hardware installed with the multiplewers (analog-to-digital and digital-to-analog converters, power supplies, chassis) and concentrators (which were considered stand-alone components).

Typical data has been used to provide an applicable solution in four cases and the results were presented discussing their significance. Modifications had to be made to the solution given by the computer program used to solve the problem to accomodate the solutions to the cases at hand. The four cases include a combination of the following two situations: a) 4 or 5 links must be connected to the central computer and b) two scenarios are considered for the life-cycle of the system. The first scenario ('INITIAL') considers only the surveillance of sensors of the plant, while an extented scenario ('GROWTH') takes in account the possibility of exercising control over some of the subsytems of the plant.

The total installation costs have been calculated to be

4 links, INITIAL scenario ---> $ 861250 dollars

4 links, GROWTH scenario ---> $ 1181250 dollars

5 links, INITIAL scenario ---> $ 877630 dollars

5 links, GROWTH scenario ---> $ 1197630 dollars

without considering the cost of the central computer and software costs, which are assumed to be the same for the 4 and 5 links configurations, although different between the 'INITIAL' and the 'GROWTH' scenarios.

A pictorial summary of the resulting configurations and the installation costs is given in FIG.3.6.

BIBLIOGRAPHY FOR CHAPTER 3

[FRAN72] FRANK, H. et al, Topological optimization of computer networks, Proc.IEEE, November 1972

[IFOC79] INTERNATIONAL FIBER OPTICS AND COMMUNICATION, Handbook and buyer's guide 1979-1980, Information Gatekeepers Inc., 167 Corey Rd. Suit 111, Brookline, Ma, 02146

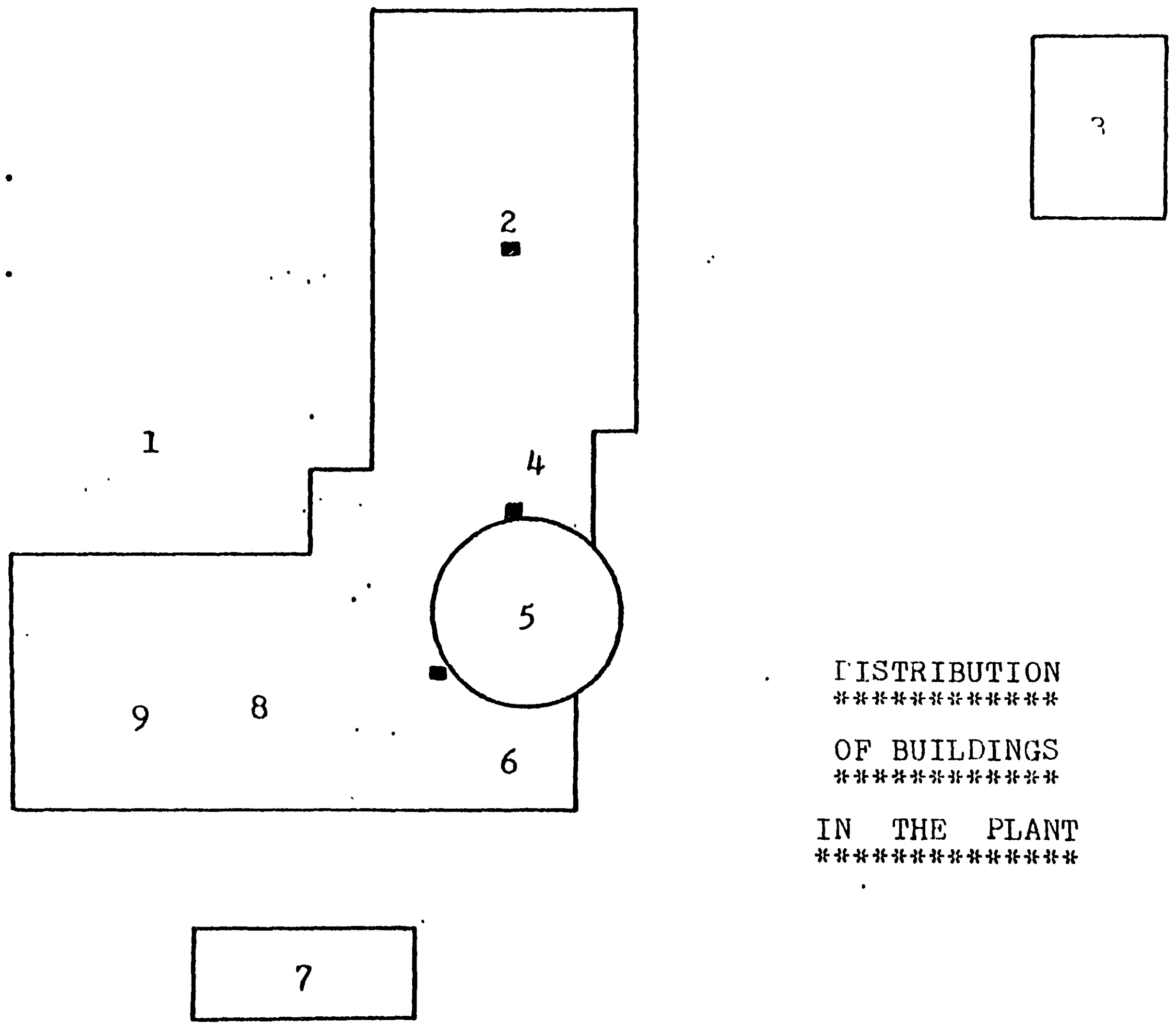
[SCHW77] SCHWARTZ, M., Computer communication networks design and analysis, Prentice-Hall, 1977

[SESA80] SESAME PRIMER, Center for Computational Research, MIT, Room 38-200, Cambridge, MA, 02139

[UNIT76] UNITED ENGINEERS AND CONSTRUCTORS, INC., Study of remote multiplexing for power plant applications, EPRI-NP-254, July 1976

List of Buildings

1	YA	Yard Area
2	TB	Turbine Bdng.
3	IS	Intake Structure
4	EFPB	Em. Feed Pump Bdng.
5	RB	Reactor Bdng.
6	FHB	Fuel Handling Bdng.
7	CT	Cooling Tower
8	PAB	Prim. Aux. Bdng.
9	WTB	Waste Treat. Bdng.



DISTRIBUTION
****

OF BUILDINGS
****

IN THE PLANT
****

FIG.3.1

SUMMARY OF SIGNALS
****

	Building	Non Safety				Safety				Scenario		Effective Signals	
		IRA		Control		IRA		Control					
		A	D	A	D	A	D	A	D	Init.	Grow.	Init.	Grow.
1	YA	63	81	-	14	4	-	-	14	148	176	178	212
2	TB	292	160	62	448	-	-	4	1	452	967	543	1161
3	IS	63	3	12	-	72	33	-	-	171	183	206	220
4	EFPB	23	6	10	18	1	4	14	127	34	203	41	244
5	RB(*)	184	2931	25	621	38	16	119	382	3169	4316	3803	5180
6	FHB	2	6	-	4	-	2	-	11	10	25	12	30
7	CT	76	13	38	122	4	-	-	62	93	305	112	306
8	PAB	108	30	8	205	14	13	52	553	165	983	198	1180
9	WTB	-	65	-	28	-	-	8	14	65	115	78	138

(*) Includes 2814 control rod signals in the IRA (D) subcategory.

Legend: A = analog, D = digital, IRA = instrumentation, alarm and recording
YA = Yard Area, TB = Turbine Building, IS = Intake Structure,
EFPB = Emergency Feed Pump Building, RB = Reactor Building,
FHB = Fuel Handling Building, CT = Cooling Tower,
PAB = Primary Auxiliary Building, WTB = Waste Treatment Building

FIG.3.[illegible]

DISTRIBUTION OF ICS ELEMENTS IN CONCENTRIC LAYERS
****

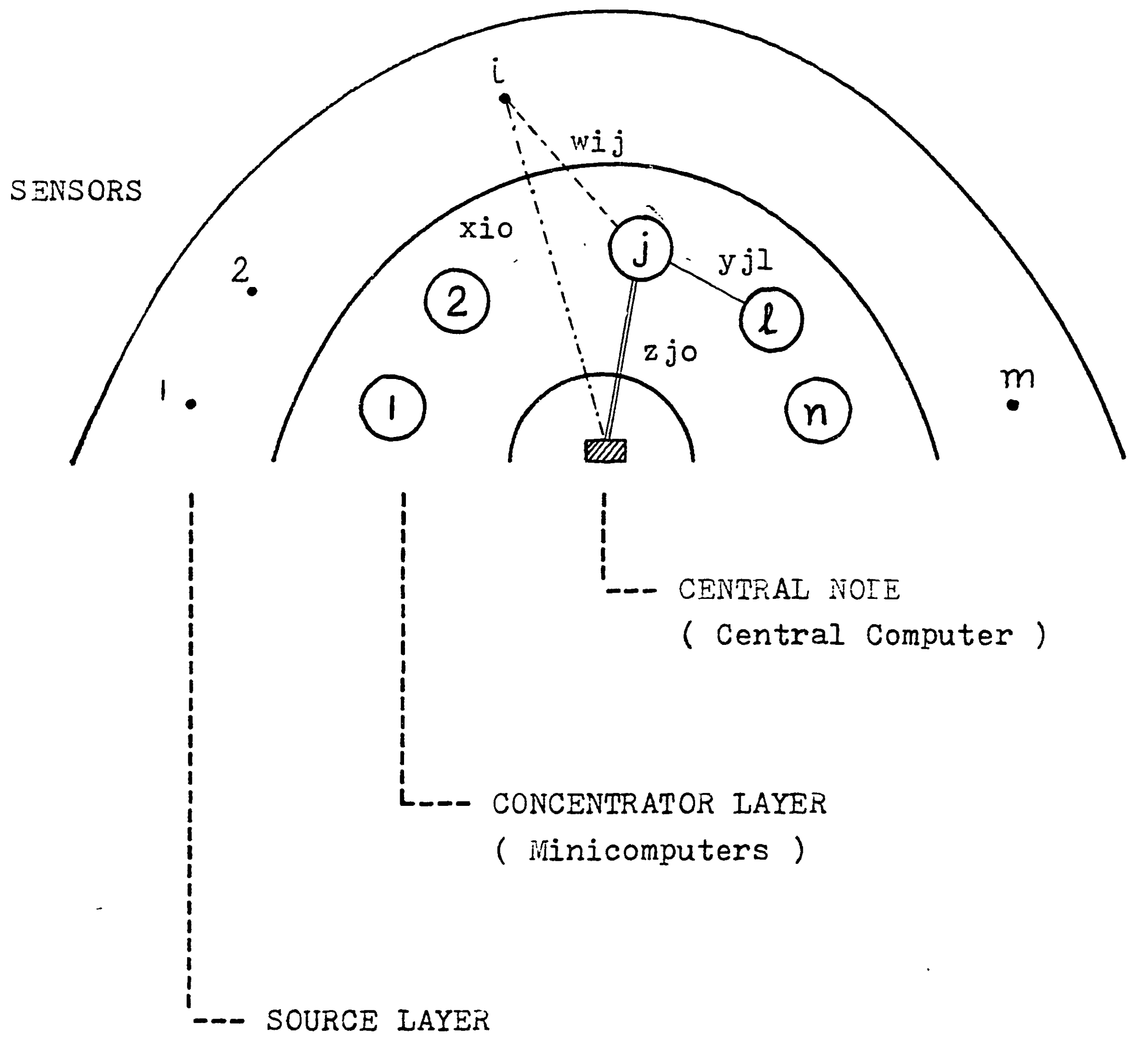


FIG.3.3

TOTAL NUMBER OF BOARDS
****

Building	Converters				Multiplexers	
	ADC		DAC			
	Init.	Grow.	Init.	Grow.	Init.	Grow.
1	3	3	-	2	3	4
2	11	14	-	17	9	19
3	6	6	-	-	4	4
4	1	2	-	6	1	4
5	9	14	-	38	60	81
6	1	1	-	1	1	1
7	3	5	-	7	2	6
8	5	7	-	29	4	17
9	-	1	-	2	2	3
Total	39	53	-	102	86	139

FIG.3.4

DISTANCES
****

From Building	To Concentrator o	1	2	3	4	5
1	320	450	700	800	1000	700
2	500	0	450	900	1100	600
3	1945	1500	1600	2100	2400	2200
4	300	400	50	400	700	500
5	400	900	250	250	600	500
6	440	1200	400	150	300	400
7	620	1500	300	400	300	500
8	400	1000	600	150	150	0
9	475	1700	700	400	200	200

From Concentrator	o	1	To 2	3	4	5
1	500	0	450	1000	1300	1000
2	350		0	300	500	400
3	420			0	200	200
4	450	(yj1 = y1j)			0	200
5	400					0

o = Central Node

Distances in feet

FIG.3.5

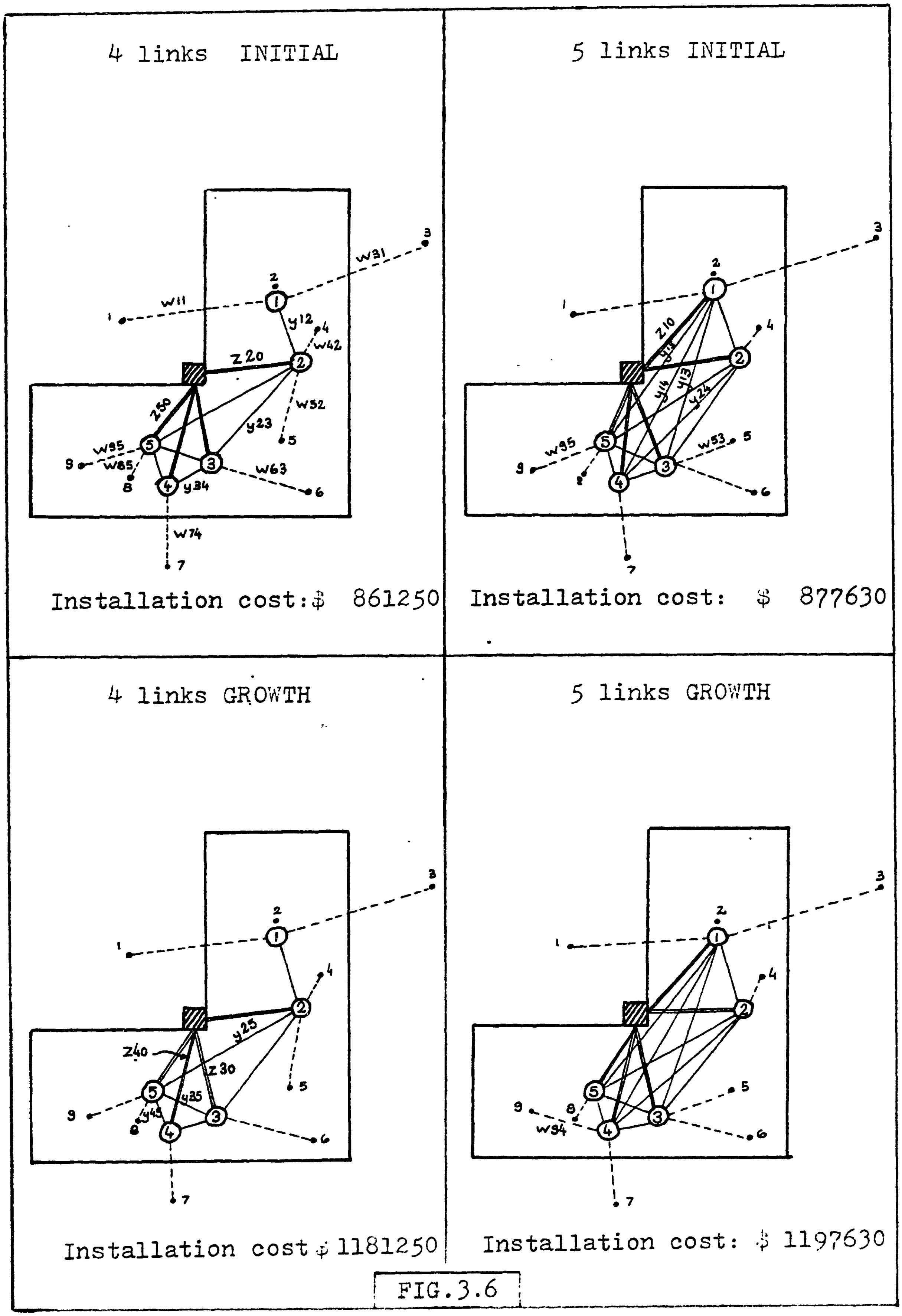

4 links INITIAL
w31
w11
y12
w42
z20
z50
w52
y23
w95
w85
w63
y34
w74
Installation cost:$ 861250
5 links INITIAL
z10
y12
y14
y13
y24
w95
w53
Installation cost: $ 877630
4 links GROWTH
y25
z40
z30
y35
y45
Installation cost $ 1181250
5 links GROWTH
w94
Installation cost: $ 1197630

FIG.3.6

CHAPTER 4

IMPLEMENTATION OF A KALMAN FILTER

In this chapter a useful technique for estimation purposes -Kalman filtering- (KF) will be introduced. Although this concept of filtering is not new, it has not been widely applied in some areas where the constraints of on-line implementation made it difficult. The wide availability of general purpose and inexpensive microprocessors has removed this limitation and has facilitated its implementation.

4.1 INTRODUCTION

In the following paragraphs of this introductory section the need for estimation will be addressed, techniques that can be used for that purpose will be mentioned and the advantages of the KF over these methods will be considered.

It will be useful to begin by describing the structure of a typical control system.

A control system that can be succesfully implemented in a limited number of cases is one consisting of a plant excited by inputs driven by a human operator. That operator 'observes' the behavior of the controlled plant (primarily from the measurements he has available), compares that information with the expected behavior he assumes for the plant, evaluates the differences and acts accordingly, graduating the inputs, to maintain these differences small. In this way, a closed-loop structure is obtained.

This structure, shown in FIG.4.1, is used whenever the time constants of the plant are long enough to allow the operator observe the responses, predict what the future values of the important parameters will be and take a decision on whether the controls should be positioned to correct, if necessary, the behavior of the plant.

In processes where

- changes of the observed parameters are very fast,
- interactions between loops make the prediction of the observed variables difficult or
- hazardous situations preclude the use of a human operator,

an automatic control system is needed. This control system will have to take decisions based on available observations (measurements) or inferred data (estimates).

These estimates have to exhibit some desirable characteristics. They have to be unbiased, so that their expected value is the same as that of the parameter being estimated. They have to be minimum variance, so that the error covariance is less than or equal to any other unbiased estimate. They also have to be consistent, so that the estimated parameter converges to its true value as the number of measurements increases.

Some of the techniques available to obtain estimates with these characteristics are least-squares, weighted-least squares, maximum likelihood and Bayesian estimation. A description of these methods can be found in reference [GELB77]. For gaussian random variables, identical results are obtained in each case. In order to find the estimates linear operations on the measurements are made. Consequently, they are called linear estimators. It can be proved that, under the very mild condition that the noises are gaussian, a nonlinear filter cannot do better than a linear filter to find the estimates. If by optimal estimator it is meant one which minimizes the mean square error, the KF is the optimal linear estimator. Its derivation can be seen in references [KALM64], [JAZW70] or [GELB77].

Then, the KF is an algorithm used to provide estimates of selected parameters that define the behavior of the plant. Recall, from FIG.4.1, that the 'plant' has been defined as a combination of the original plant, the sensors and the actuators. The incorporation of a KF to the plant can be seen in FIG.4.2. The addition of the KF will help the operator in the decision process because he now has available estimates of those state variables that were not directly measured. With respect to FIG.4.1 this scheme represents an improvement and it is the one considered as the first possible scenario in the life of the computer system proposed for the plant.

An even better situation would be one where the control functions of the operator are replaced by a similar group of functions of the computer system, and this is the second scenario in the design proposed in Chapter 2. The structure is shown in FIG.4.3. In this case the inputs to the plant are given by an optimal control law easily applied on-line, feature that makes the method attractive for implementation. For the derivation of this control law see reference [KWAK72].

Summarizing what has been explained, the KF provides a simple means of estimating unmeasurable parameters of a plant, and these estimates are optimal in the sense that the variance of the error between the estimated and the real parameter is a minimum which cannot be improved with any other filter. These estimates can be used to improve the knowledge of the plant and to aid in its control, either by an operator or by a computerized control system.

Previous treatments of this topic can be seen in references [GURA71], [MEND71] and [SCHM70], to name a few. However, they have been limited to the calculation of the number of additions and multiplications necessary to produce the updated estimates and the time consumed by any other operation was considered explicitly only for particular cases.

In the following sections the model of the filter is shown, the time and memory requirements between updates are calculated and these quantities are used with the concept of 'instruction mixes' to calculate the time and storage required for its on-line implementation.

In this Chapter a calculation will be made to find the total time and storage needed for the implementation of the filter under the rather general condition of time-varying error covariance matrix Qk (see Notation, Section 4.2). Operations that were only partially considered before or applicable to only one kind of processor will now be included.

4.2 THE MODEL OF THE STANDARD KALMAN FILTER

The particular KF to be used will depend on characteristics of the process to be 'observed'. The selection of the Standard Kalman Filter as the algorithm to be treated in this chapter has been made to apply the procedure to estimate the computational complexity to a technique that is gaining popularity in the nuclear field and at the same time to have a frame of reference to estimate the order of the linearized models that can be used in real-time calculations. FIG.4.2 shows the calculations that take place in the execution of the algorithm.

The equations necessary to produce the updated estimates xk+1,k+1 are [GURA71], [GELB77]:

System model

$$xk+1,k = \Phi * xk,k + \Gamma * wk \qquad (4.1)$$

Measurement model

$$zk+1,k = H * xk+1,k + vk \qquad (4.2)$$

Error covariance extrapolation

$$Pk+1,k = \Phi * Pk,k * \nabla^{\dagger} + Qk \qquad (4.3)$$

Kalman gain

$$Kk+1 = Pk+1,k * H^{\dagger} * [H * Pk+1,k * H^{\dagger} + R]^{-1} \qquad (4.4)$$

Error covariance update

$$P_{k+1,k+1} = P_{k+1,k} - K_{k+1} * H * P_{k+1,k} \quad (4.5)$$

State estimates update

$$\nu_{k+1} = z_{k+1,k+1} - z_{k+1,k} \quad (4.6)$$

$$x_{k+1,k+1} = x_{k+1,k} + K_{k+1} * \nu_{k+1} \quad (4.7)$$

Initial conditions

$$E\{x(0)\} = x0, \; E\{(x(0)-x(0))(x(0)-x(0)^{\dagger})\} = P\dot{0}$$

Notation

n	=	number of state variables = order of the system
m	=	number of measurements
x	=	state estimate vector = dim(nx1)
$x_{k,k}$	=	state estimate @ tk given measurements were made @ tk
Φ	=	state transition matrix = dim(nxn)
Γ	=	system noise distribution matrix = dim(nx1)
w_k	=	system noise @ tk
H	=	measurement matrix = dim(mxn)
v_k	=	measurement noise @ tk
$P_{k+1,k}$	=	extrapolated error covariance matrix = dim(nxn)
$P_{k+1,k+1}$	=	updated error covariance matrix

Qk	=	system noise covariance matrix = dim(nxn)
Rk	=	measurement noise covariance matrix = dim(mxn)
Kk+1	=	Kalman gain @ tk+1
νk+1	=	innovation @ tk+1
(.)†	=	transpose of (.)
E{.}	=	expected value of {.}

Assumptions

The matrices B, Γ, H and R are constant.

The matrices P, Q and R are symmetric.

Noises: .system noise w is Gaussian (normal), zero mean, with covariance matrix Q, and that will be denoted as N(0,Q).

.measurement noise v is also N(0,R).

.noises are uncorrelated, that will be indicated by E{ wj vk† } = 0 for all j,k

Figure 4.4 (modified from [GELB77] will prove useful in understanding how the KF works and introduces the concepts of extrapolation and update. Assume that a set of measurements are taken at time tk. These measurements, zk,k, are used to find the innovations νk which will update the estimated states variables xk,k-1 extrapolated from tk-1. The Kalman gains Kk are obtained from the covariance matrix Pk,k-1 and they are used to update the estimated state variables and the covariance matrix P. With this set of updated quantities and the use of the state transition matrix and the new measurement covariance matrix Qk, a new extrapolation is performed. The acquisition of a new set of measurements at tK+1 will initiate the cycle once more with a new update.

For generallity, the update of the covariance matrix Q was considered. A procedure to accomplish this and its proof were presented in [D'APP66]. Figure 4.5 is a flowchart that shows how the algorithm works. At every sampling instant ti (where i = ...,k,k+1,...) an nxn matrix, D', is initialized and the parameter r that determines the number of iterations is reset to zero. The matrices D' and F' are calculated by doing

$$D'(r+1) = \frac{\Delta T}{r+1} * (F'(r) * G * Qk * G - D'(r) * F)$$

where

$$F'(r+1) = \frac{\Delta T}{r+1} * (F * F'(r))$$

with $D'(0) = 0$ and $F'(0) = I$.

Note that the matrix $F'(r+1)$ is calculated just once, when the model of the subsytem is loaded in microprocessor's memory. After that it is a constant.

The matrices D'' and Φ are defined by

$$D'' = \sum_{i=0}^{\infty} D'(i)$$

and

$$\Phi = \sum_{i=0}^{\infty} F'(i)$$

Having calculated the state transition matrix Φ, the matrix Qk is given by

$$Qk = D'' * \Phi$$

If the Qk matrix converges to the desired predetermined accuracy or the final number of iterations has been reached the algorithm terminates; otherwise,it continues.

The characteristics of the method are that "it converges rapidly ($10 < r < 22$) to five or six place accuracy in single precision" with " ΔT significantly less than the dominant time constant of the system under analysis".

4.3 OPERATIONS AND STORAGE REQUIRED BETWEEN UPDATES

Some of the particularities considered in this implementation of the KF are that

- If it is decided that it is not necessary to update the error covariance matrix Qk at every sample, the parameter r that controls the number of iterations must be set to zero.
- the matrix inversion algorithm. In this work it will be assumed, following reference [GURA71] that the inversion requires ((m**2*(m+3)/2)+q*m) operations for an mxm matrix; the parameter q was set equal to 5 in [SCHM70], 7 in [GURA71] and 10.25 in [WEIT79].
- the processor and particular clock frequency used (in this case, a MOTOROLA Mc68000 microprocessor based system with an 8MHz clock). The processor defines the number of machine instructions executed per operation and the clock determines the time required to perform a machine instruction.
- a reduction in the number of operations and in the storage is possible if the symmetry of some matrices is considered and an efficient allocation of the matrices, vectors and intermediate results is performed.

The analysis of the number of operations and storage required in the KF algorithm can be divided into two stages: first the Standard KF itself and second, the update of the Qk matrix.

The sequence showing the operations performed in the first stage is presented in FIG.4.6. The update of the Qk matrix is considered separately but it follows the same guidelines. Divisions are not taken in consideration because there are very few of them and substractions are counted with additions because the number of machine instructions per operation is the same.

As a result of the counting made in FIG.4.6 the number of multiplications (NMUL) performed between updates (with q = 7) gives:

$$\begin{aligned} \text{NMUL} = {} & 1.5\ n{*}{*}3 + 1.5\ n{*}n + 2.5\ n{*}n{*}m + \\ & + 3\ n{*}m + 1.5\ n{*}m{*}m + \\ & + 7\ m + 1.5\ m{*}m + .5\ m{*}{*}3 \end{aligned} \tag{4.8}$$

and the number of add/substr. (NADD) is

$$\begin{aligned} \text{NADD} = {} & 2.5\ n{*}{*}3 - .5\ n{*}n + 2.5\ n{*}n{*}m + \\ & + .5\ n - .5\ n{*}m + \\ & +1.5\ n{*}m{*}mm + .5\ m{*}m + .5 \end{aligned} \tag{4.9}$$

The storage required at this stage depends on the fact that the dimension of the measurement vector z, which is equal to the number of outputs, m, may be smaller, equal or greater than the order of the system, n. Under these conditions, the storage of data becomes:

for $m < n$ $(.5n)(n+4m+5)+m$ (4.10)

for $m = n$ $.5\ (5\ n*n + 7\ n)$ (4.11)

for $m > n$ $(.5n)(n+2m+5)+m(m+1)$ (4.12)

Repeating the procedure to calculate the number of operations performed and the storage required for the update of the Qk matrix the following results are obtained (neglecting the 2(r+1) divisions):

multiplications	$(n**3)(2r+3) + n*n*(r+1)$	(4.13)
add/substr.	$(n**3)(2r+3) - n*n$	(4.14)
storage of data	$5.5\ n*n + .5\ n$	(4.15)

As a final element to calculate the total storage it is necessary to know the amount required by the applications programs. The Standard KF program size was shown in [MEND71] to be 488 words long (140 words of main program and 348 words of subroutines), or, equivalently, 996 bytes. The storage for the program shown in FIG.4.5 can be calculated with the techniques developed by [HALS76]. In that work it was stated that some parameters of a program could be calculated with a very acceptable accuracy just knowing the number of inputs plus outputs the program has and the language used to write it. Given that there are 5 inputs and 2 outputs, the program length in ASSEMBLER is 115 bytes. Then, the program storage becomes storage of program = 1111 bytes (4.16)

Now the elements to calculate the total number of multiplications is available from (4.8) and (4.13); for add/subst.: (4.9) and (4.14) and for storage: (4.10), (4.11), (4.12), (4.15) and (4.16).

Finally the totals become

multiplications

$$.5[(n**3)(4r+9)+(n*n)(2r+5+5m)+n(6m+3m*m)+ +m(14+3m+m*m)] \quad (4.17)$$

add/subst.

$$.5[(n**3)(4r+11)+(n*n)(5m-3)+n(1-m+3m*m)+ +m(2+m)+1] \quad (4.18)$$

storage [bytes]

for $m < n$	$12n*n+4nm+6n+2m$	$+ 1111$	(4.19)
for $m = n$	$16n*n+8n$	$+ 1111$	(4.20)
for $m > n$	$12n*n+2nm+6n+2m*m+2m$	$+ 1111$	(4.21)

4.4 INSTRUCTION MIXES

What has been calculated -arithmetic operations- is not enough to determine the time spent between measurement updates because three other classes of operations necessarily performed by the processing unit have not been taken into account. They are divided into logical, control and input/output (I/O). Every application program operates with these four categories of instructions and, according to the particular use(navigation, track and command, radar data processing, real time, control and display, etc) the percentage of instructions in the actual program execution varies [CORS70]. For example, in numerical applications, the highest percentages are likely to be found in two categories: arithmetic (multiplications, divisions, additions and substractions) and control (loading registers, requesting data from memory). Such a blend of instructions is called a 'instruction mix'. One example of it -for real time- can be seen in FIG.4.7, with the typical execution times that correspond to a MC68000. The individual percentages corresponding to multiplications and add/substractions, 5% and 16%, have been modified to adapt the percentages to this application, where it is known that they are in the approximate ratio (multiplications)÷(add/substractions) = 1.12. As their sum has to be 23% for this instruction mix, it is seen that the individual percentages should be 10.8 and 12.2 respectively.

Expressing the required time to perform the calculations in terms of the number of multiplications and add/subst. is a straightforward derivation:

$$TT = \text{Total Time} = 11.447 * NMUL + 3.197 * NADD \quad (4.22)$$

where NMUL is given by (4.17) and NADD by (4.18).

Given that NMUL ≈ 1.12 NADD, it can be seen from the above equation that including those operations that are not exclusively multiplications and additions increase the execution time by 51.61%.

Applying the equations to different values of n and m FIG.4.8 is obtained and doing a least squares fit to the data the results are the following (TT = Total time in seconds, S = storage in kilobytes):

for m = n/2 $TT = 3.641\ 10^{-4}\ n^{2.923}$

$S = 0.249\ n^{1.067}$

for m = n $TT = 3.932\ 10^{-4}\ n^{2.927}$

$S = 0.231\ n^{1.137}$

for m = 2n $TT = 5.372\ 10^{-4}\ n^{2.915}$

$S = 0.203\ n^{1.302}$

Plots corresponding to these equations can be seen in FIG.4.9 for the ranges of interest (4 to 20 state variables).

4.5 CONCLUSIONS

The conclusions on the real-time use of a KF can be summarized in the following statements:

- the time spent increases with the third power of the number of state variables (n). The importance of obtaining Reduced Order KFs is immediate: the model of the plant that is used should be limited or reduced to a manegeable number of state variables (14-15 seems to give a good update interval ≈ 1 second). If the reduction is a must, it could imply a loss of accuracy of dynamics modelling. Reducing the number of state variables without losing too much accuracy is an art in the control field and there are not well-proven methodologies to do that automatically. Therefore, an alternative approach that deserves attention is the real-time 'batch' processing suggested by [MEND71] where the number of measurements to be processed is divided into batches. However, as can be seen from FIG.4.9 the sensitivity of total time required to produce the updated variables (TT) to the number of measurements is second order with respect to the changes in the number of state variables.
- the storage increases with the number of state variables to a much lower power. For the ranges considered (at most 20 state variables), there are no limitations due to storage because

mini-microcomputer boards already come with 32 or 64 Kbytes of ROM (programs) and 64 to 128 Kbytes of RAM (data) and there are no problems in obtaining more memory in additional boards.

BIBLIOGRAPHY FOR CHAPTER 4

[CORS80] CORSIGLIA, J., Matching computers to the job, first step towards selection, _Data Processing Magazine_, December 1970

[D'APP66] D'APPOLITO, J., A simple algorithm for discretizing linear stationary continuous time systems, _Proc. of the IEEE_, December 1966

[GELB77] GELB, A.,editor, Applied optimal estimation, MIT Press, Cambridge, MA, 1977

[GURA71] GURA, I. et al, On computational efficiency of linear filtering algorithms, _Automatica_, _7_, 1971

[HALS76] HALSTEAD, _Elements of software science_, Prentice Hall, 1976

[KALM64] KALMAN, R., When is a Linear Control System Optimal, _J. Basic Eng._, 86, 1964

[KWAK72] KWAKERNAAK, H. et al, _Linear optimal control systems_, John Wiley and Sons, 1972

[JAZW70] JAZWINSKY, A., _Stochastic processes and filtering theory_, Academic Press, 1970

[MEND71] MENDEL, J., Computational requirements for a discrete KALMAN Filter, _IEEE Trans. Autom. Control_, December 1971

[SCHM70] SCHMIDT, S., Computational techniques in KALMAN Filtering, AGARDograph 139,, February 1970

IMPLEMENTATION OF CONTROL SYSTEMS

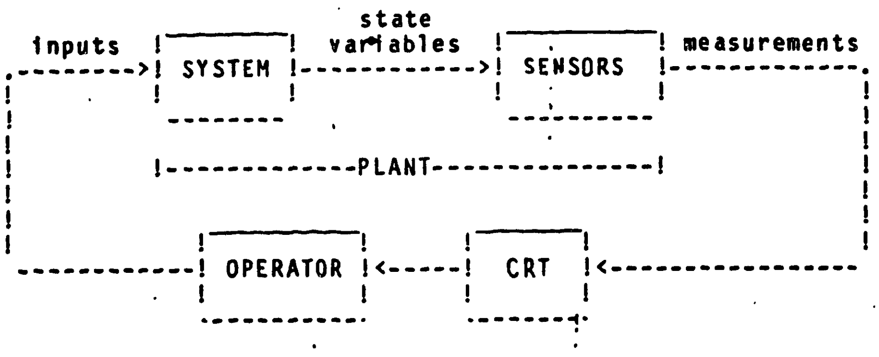


FIG.4.1

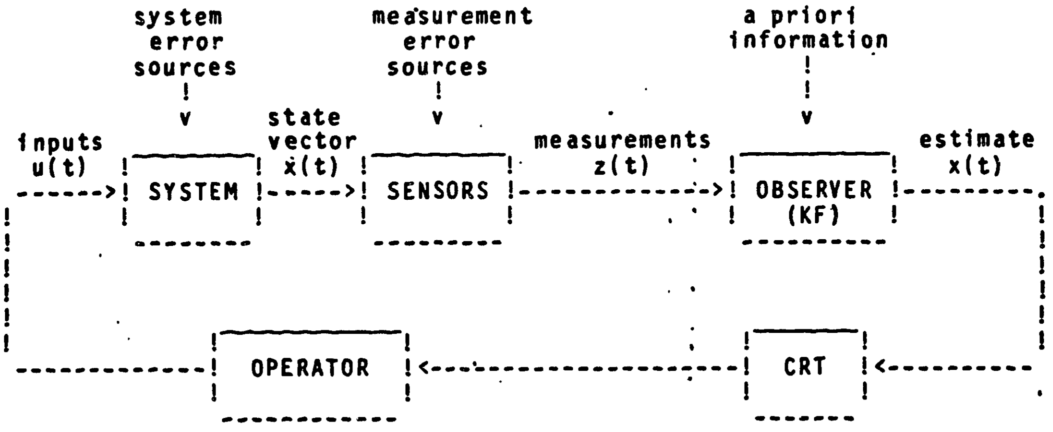


FIG.4.2

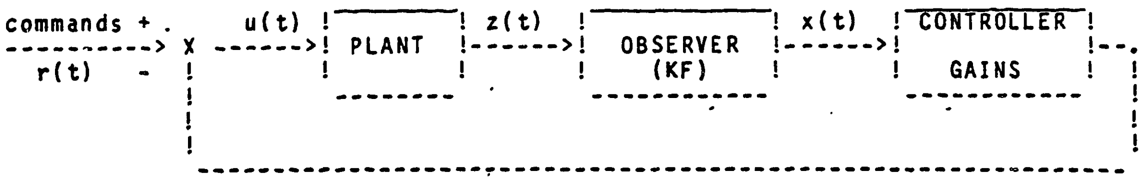


FIG.4.3

COMPUTATIONS PERFORMED IN THE KALMAN FILTER ALGORITHM
****

```
                        Φ----.
                        !    !
                        ! FIG.4.5
                        !    !
                        !    Qk
                        !    !
                        v    v
Pk,k-1-->(IV.5)-->Pk,k--0----0-->(IV.3)-->Pk+1,k-->(IV.5)-->Pk+1,k+1
   !        !           !                    !        !
   !        !           !                    !        !
   v        !           !                    v        !
(IV.4)----> K           !                 (IV.4)----> K
            !           !                             !
            !           !                             !
            v           v                             v
xk,k-1-->(IV.7)-->xk,k--0------->(IV.1)-->xk+1,k-->(IV.7)-->xk+1,k+1
   !        !                                !        !
   !        !                                !        !
   v        !                                v        !
 (IV.2)     !                             (IV.2)      !
   !        !                                !        !
   !        !                                !        !
   v        !                                v        !
zk,k-1-->(IV.6)                           zk+1,k-->(IV.6)
            !                                         !
            !                                         !
            !                                         !
          zk,k                                    zk+1,k+1

--------------------------------------------------------------------
           tk                                          tk+1       time

     !---update---! !------extrapolation------! !---update---!
```

FIG.4.4

UPDATE OF THE NOISE COVARIANCE MATRIX Qk
****

```
. Time = ti ; i=...,k,k+1              begin @ tk,tk+1,...
. D'(0)= 0.0 ; F'(0) = I               initialize D' and F'
. r = 0                                reset counter
. repeat until convergence of A and Qk or desired iteration
  .D'(r+1) = [F'(r).G.Qk.G  - D'(r).F]* T/(r+1)
  .D" = D'(r)+D'(r+1)                  D" = SUM(0,∞,D'(i))
  .D'(r) <--- D'(r+1)                  Load D'(r+1)-->D'(r)
  .F'(r+1) = [F.F'(r)]* T/(r+1)
  .Ak+1,k = F'(r) + F'(r+1)            Ak+1,k = SUM(0,∞,F'(i))
  .F'(r) <--- F'(r+1)                  Load F'(r+1)-->F'(r)
  .r = r+1
. else
  .Qk = D"*Ak+1,k
. end
```

FIG.4.5

OPERATIONS PERFORMED IN THE IMPLEMENTATION OF THE KALMAN FILTER ALGORITHM
****

EQUATION	OPERATION	ORDER	MULTIP.	ADD/SUBST.
4.1	I = Φ * xk,k	nxn . nx1	n**2	n(n-1)
	II = I + Φ * wk	nx1 + nx1	--	n
4.2	III = H * xk+1,k	mxn . mx1	nm	m(n-1)
	IV = III + vk	mx1 + mx1	--	m
4.3	V = Φ * Pk,k	nxn . nxn	n**3	n(n**2-1)
	VI = V * Φ	nxn . nxn	n**2(n+1)/2	n(n**2-1)/2
	VII = VI + Qk	nxn + nxn	--	n(n**2+1)/2
4.4	VIII = Pk+1,k * H	nxn . nxm	mn**2	mn(n-1)
	IX = H * VIII	mxn . nxm	mn(m+1)/2	m(m+1)(n-1)/2
	X = IX + Rk	mxm + mxm	--	m**2
	XI = X	mxm ->mxm	in the text	--
	XII = VIII * XI	nxm . mxm	nm**2	mn(m-1)
4.5	XIII = H * Pk+1,k	mxn . nxn	mn**2	mn(n-1)
	XIV = XII * XIII	nxm . mxn	mn(n+1)/2	n(n+1)(m-1)/2
	XV = Pk+1,k - XIV	nxn - nxn	--	n(n**2+1)/2
4.6	XVI = zk+1,k+1-zk+1,k	mx1 - mx1	--	m
4.7	XVII = XII * XVI	nxm . mx1	nm	n(m-1)
	XVIII = xk+1,k + XVII	nx1 + nx1	--	n

FIG.4.6

INSTRUCTION		PERCENTAGE	CLOCK PERIODS	EXECUTION TIME (1)
ARITHMETIC		(23.0)		
Multiplication	(MULS,Dn)	10.8	71	8.875
Addition	(ADD,Dn)	12.2	5	0.625
LOGICAL		(21.0)		
Compare	(CMP,Dn)	12.0	5	0.625
Shift	(ASR)	5.0	6	0.750
And/Or	(AND)	4.0	5	0.625
CONTROL		(52.0)		
Load	(LEA,Am@)	33.0	4	0.500
Conditional branch	(BCC)	10.0	10	1.250
Increm. & store	(LEA+An@+)	4.0	8	1.000
Move reg to reg	(MOVEM,An@)	5.0	13	1.625
I/O		(4.0)		
Programmed I/O	(MOVEP)	2.0	17	2.125
Buffered I/O		1.0	0	0.000
Int.response		1.0	0	0.000

(1) Clock periods * (1/frequency) = Clock periods * (.125) in microseconds

FIG.4.7

Tim (sec)/Storage(Kb)

Number of state variables n

[illegible]	5	1	14	15	20
0.5	[illegible] / [illegible]	[illegible]/ 2.56	[illegible].824/ 3.55	1.010/ 4.37	2.273/ 5.85
1.	[illegible]/ [illegible]	[illegible].329/ 2.73	[illegible].887/ 4.56	1.088/ 4.83	2.552/ 7.67
2.	[illegible]/ [illegible]	[illegible].456/ 3.61	1.17[illegible]/ 4.84	1.437/ 6.66	3.356/10.91

FIG.4.3

LIMITATIONS OF THE REAL-TIME IMPLEMENTATION OF A KALMAN FILTER

AS A FUNCTION OF THE NUMBER OF STATE VARIABLES

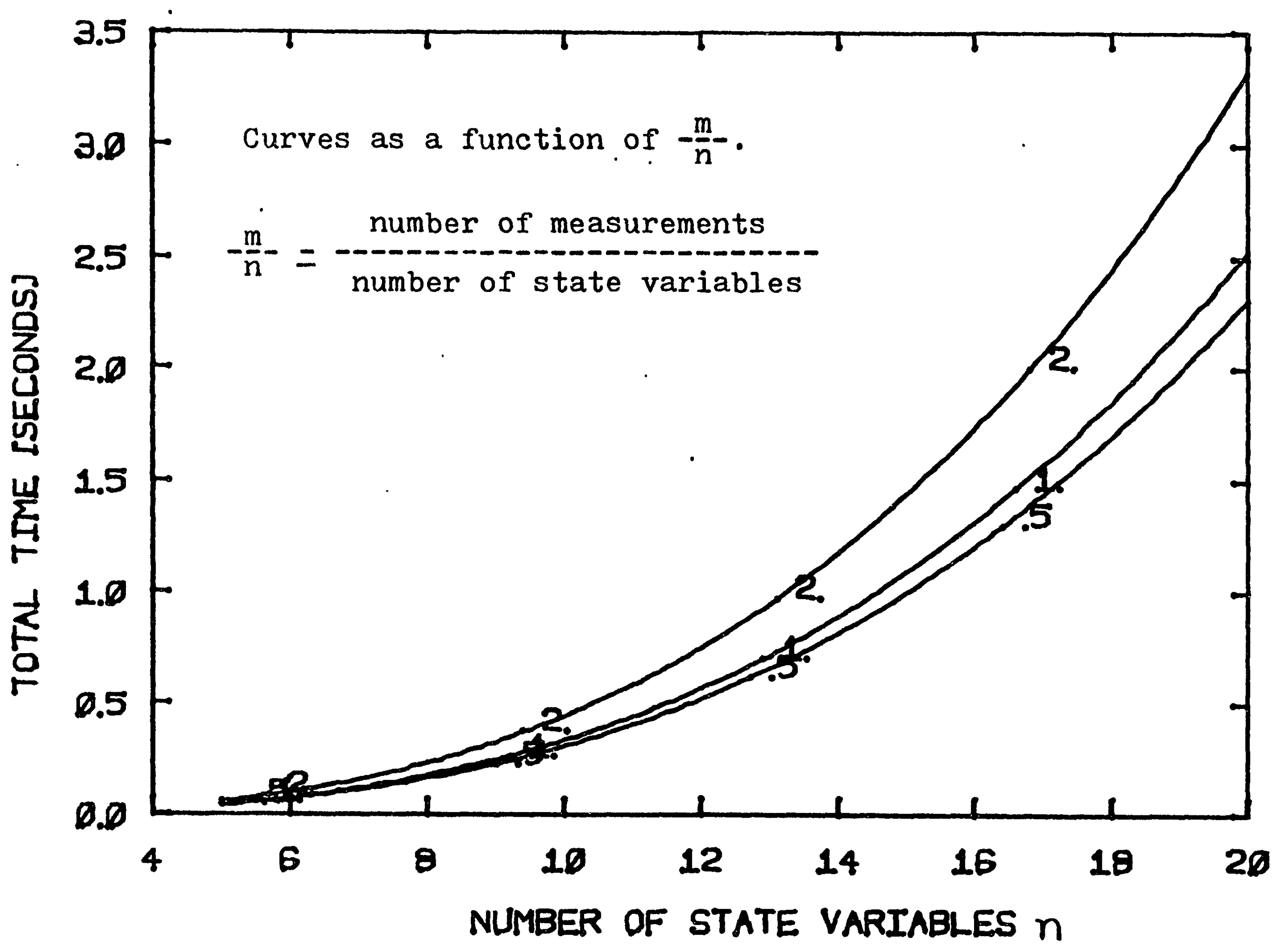


FIG.4.9.a

STORAGE LIMITATIONS IN THE IMPLEMENTATION OF A KALMAN FILTER

AS A FUNCTION OF THE NUMBER OF STATE VARIABLES

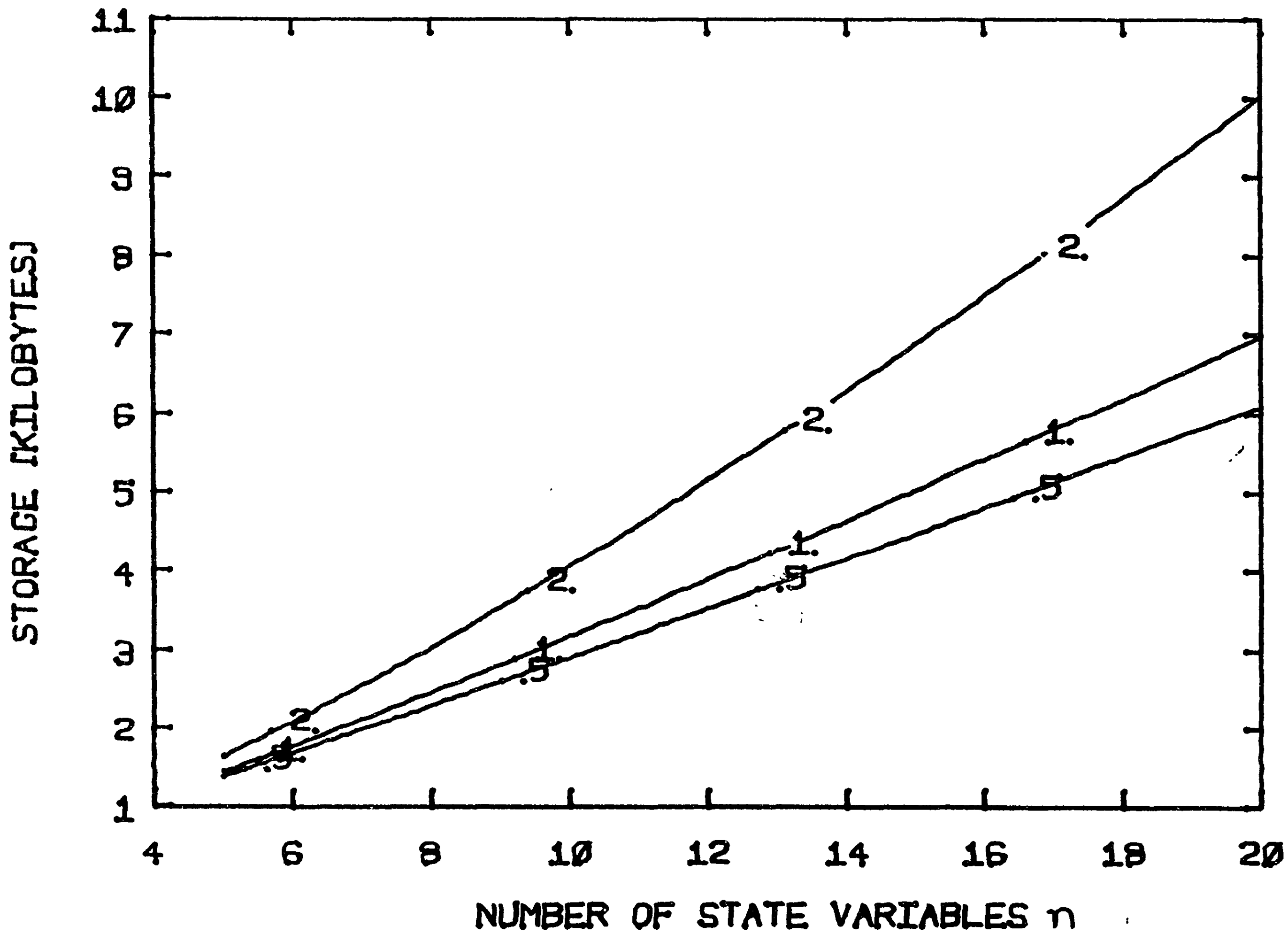


FIG.4.9.b

CHAPTER 5

A SEQUENTIAL DETECTION ALGORITHM

5.1 INTRODUCTION

There are a number of situations in which it is necessary to know (within a specified confidence level) if a certain parameter (ie estimated mean) is equal to a predetermined constant or stays within predetermined limits. If this is not the case it could be necessary, for example, a) to alert an operator that something is not functioning properly so that he could take the appropriate correcting actions or b) indicate that the plant is working about a new nominal operating point and, consequently, a new linearized model will represent it better.

In these cases the observed data is drawn from a population with an assumed (or to be calculated) probability density function and within that assumption some other parameters are calculated (mean, standard deviation, variance, standard deviation of the estimated mean, etc).

When only one parameter of the population is sought, that situation is called a _point estimation_. The _interval estimation_ finds the interval that includes the population parameter for a given confidence level. _Hypothesis testing_ decides if the population parameter is contained inside the given interval for specified type I and type II errors (defined below).

The objective of this chapter is to show how to estimate the time and storage required by a particular sequential detection algorithm (the Sequential t-Test or StT) with a similar methodology to that shown in Chapter 4, and oriented also to real-time applications. Two features that make the algorithm useful for on-line implementation are that

- it was designed so that the average number of observations to reach a decision is minimal and
- it does not need to calculate the variance of the observed parameter (if coupled with a Kalman Filter a minimum mean square error between the observed and the estimated parameter is assured).

The objective of the Chapter is _not_ to propose the actual implementation of the StT simply because in this work it has not been compared to other methods for sequential detection. That comparison would not only depend on time and storage requirements but on other conside·tions as well, for example accuracy and stability of the algorithm, which will depend on characteristics of the observed process.

5.2 HOW THE SEQUENTIAL DETECTION WORKS

In Hypothesis Testing the following steps are taken: a particular probability density function pdf (normal, binomial, Poisson, etc) is proposed as representing the observed population. Hypothesis regarding the relationship between the estimated data and some predetermined limits are stated, a numerical criterion on which to base the decision is calculated, samples of measurements are taken, data is analyzed, the decision is made and the sequence begins once more.

FIG.5.1 represents a normal pdf of the estimated parameter θ vs. θ itself. In the region labelled ACCEPT the decision is to accept the calculated value of the parameter θ as an estimate of the true mean of the population for the specified interval. This is the H0 or null hyphotesis. If the estimated θ falls outside the interval $\theta 1 < \theta < \theta 2$, the decision is to REJECT the H0 hypothesis, and this is called the H1 or alternative hypothesis.

The acceptance or rejection of hypothesis leads to two errors named:

type I : although $H0$ is true, the decision is to accept $H1$.

type II: although $H1$ is true, the decision is to accept $H0$.

The first kind of error happens when a correct numerical value of the estimated parameter is taken as incorrect. The second kind of error occurs when an estimate of that parameter, which falls outside the limits $\theta 1$, $\theta 2$, is taken as a valid representative. From the figure and from what has been expressed it is seen that the further apart are these limits the smaller the chance of making the errors. The price to be paid is an increase in the number of observations to reach a decision.

The probability of making a type I error is α and that of making a type II error is β. When the number of observations and α are fixed, β becomes a function of α (Neyman-Pearson theory [WALD47], Bayesian tests [HANC66]). However, when α and β are fixed the number of observations becomes a random variable. The average number of observations to arrive to a decision is the parameter minimized in the sequential test that is going to be used, making it useful for real-time applications.

WALD in [WALD47], proposed a series of detection methods based on a series of pdfs (normal, binomial, etc) and with other characteristics as well (simple and composite hypothesis, etc). Two of them were considered for evaluation in this thesis and in both the parent population was assumed normal: the first considers as known parameters the mean and standard deviation of the population and the second considers that the only known value is the mean. The first test is called Sequential Probability Ratio Test or SPRT. However, for generality, it was decided to use the second test, called the Sequential t-Test or StT.

Given that $\theta 1$, $\theta 2$, ...,θl are the unknown parameters of the population under test, a simple hypothesis is made when unique values are assigned to them. For example, if the proposed distribution of the population is normal it can be defined with two parameters: the mean and the standard deviation. If the mean is assumed to be, say $\theta 1 = 10.0$ and the standard deviation $\theta 2 = 2.5$, the hypothesis to be tested is simple. In this case the SPRT may be used. When one or more of the parameters θ is not uniquely determined, for example if $\theta 1 = 10.0$ but $\theta 2$ is $2.1 < \theta 2 < 2.7$, then the hypothesis is called composite, and this is our case.

5.3 THE SEQUENTIAL t-TEST

Assume that the mean of a population normally distributed is θ and that the estimated mean is θo. If the standard deviation of the population (unknown) is σ, it is possible to say that θo can be taken as the true mean whenever the error $|(\theta - \theta o)/\sigma|$ is less or equal than some specified value δ. According to this procedure it is possible to define that the error is a given percentage of the -unknown- standard deviation. The test accepts with probability $1-\alpha$ the hypothesis that θ^0 is the mean of the population and rejects it with probability β.

Given the sequence of observations $x1, x2, \ldots, xn$ drawn from the above population, the test, as originally proposed by [WALD47] is based on the comparison

$$\beta/(1-\alpha) < p1n/p0n < (1-\beta)/\alpha \qquad (5.1)$$

where α and β are the probabilities of making errors of the first and second kind respectively and

$$\frac{p1n}{pon} = \frac{\int_0^\infty \frac{1}{(\sigma^{**}n)} \exp\left(\frac{-1}{2\sigma^2}\right) \sum_{i=1}^{n} (xi-\theta o-\delta\sigma)^2 + \frac{1}{(\sigma^{**}n)} \exp\left(\frac{-1}{2\sigma^2}\right) \sum_{i=1}^{n} (xi-\theta o+\delta\sigma)^2]\, d\sigma}{2 \int_0^\infty \frac{1}{(\sigma^{**}n)} \exp\left(\frac{-1}{2\sigma^2}\right) \sum_{i=1}^{n} (xi-\theta o)^2 \, d\sigma} \qquad (5.2)$$

The test is initiated when a threshold is violated. That threshold could be prescribed in terms of upper or lower absolute limits for a measured variable or a difference between results obtained from, say, analytic redundancy calculations and the estimated parameter from a Kalman Filter for a non measurable variable. The smaller the difference between the upper and lower thresholds, the more frequent the StTs will be performed. The bigger the difference between them, the higher the probability of not detecting a change in the mean value of the estimated parameter. The tradeoff will depend on the characteristics of the measurements being sampled.

Additional observations are taken as long as p1n/p0n stays within the limits, the hypothesis is accepted as soon as

$$p1n/p0n < (1-\beta)/\alpha$$

and it is rejected as soon as

$$\beta/(1-\alpha) < p1n/p0n.$$

Although the test reaches a decision as soon as any of the two previous conditions are satisfied, the actual programs to perform the sequential detection routines should include a specified number of repetitions so as to eliminate almost completely the possibility of accepting erroneous decisions.

Perhaps the most important reason why the StT was not widely used is because its implementation required a set of Tables created only for that purpose [AMS749]. The contribution of this thesis is to provide an algorithm that can be used with microprocessors (see Appendix B) and avoids completely the use of any table.

5.4 TIME AND MEMORY REQUIREMENTS

Following the work done in Chapter 4, the algorithm will be decomposed in its constitutive operations. The total time spent in one iteration will be the sum of the time spent per operation times the number of operations per iteration. In this case it will not be necessary to use a prefabricated instruction mix to find the approximate percentage of the executed operations because the program has been implemented in a HP-67 programmable calculator and the actual number of operations will be taken directly from that program, described in Appendix A.

An additional complication, not considered in the instruction mix used in Chapter 4, is the need to use special purpose subroutines to calculate square-roots, logarithms and exponentials. The details can be seen in Appendix B.

The total number of operations, the time to perform the and the total storage are detailed in Appendix C. Note that the number of operations is different depending which iteration number is considered but it is a constant after the sixth sample has been taken (n=6). From this Appendix it is seen that the time required to perform one iteration (one per sample) is roughly equal to one milisecond and the total storage (program + data) is approximately 800 bytes.

The maximum number of signals (sensors) connected to a microprocessor or the sampling rate can be obtained as a consequence of these results.

Given that

- the total time of execution is roughly equal to one milisecond per iteration in the microprocessor considered and
- a percentage (PS) of the total number of sensors connected to the microprocessor is sampled every TS seconds,

it can be seen that the total number of sensors that can be monitored by the microprocessor is 100000*TS/PS. If 20% of the sensors (PS=20) are sampled every second (TS=1), the maximum number of sensors to be connected is 5000.

However, many other programs will be running in the same unit (routing and filtering algorithms, self-tests, etc) and that maximum number of sensors will be sensibly reduced by a factor RF, which can be defined as the percentage of the time that the microprocessor is expected to be busy with the sequential detection algorithm. Then, the maximum number of sensors becomes 1000*TS*RF/PS. If, as before, TS = 1, PS = 20 and now RF = 10, the maximum number of sensors to be connected is reduced to 500.

If the case that all sensors are not sampled at the same time, sensors that work with the same sampling rate can be grouped for the purposes of analysis. If the generic group is called i, the sampling rate applied will be called TSi and those sensors will represent a percentage PSi of the total number of sensors.

With this notation, the maximum number of sensors to be connected is

$$\frac{1000 * RF}{\displaystyle\sum_{i=1}^{ilast} \frac{PSi}{TSi}}$$

As an example and using the previous figures, if now half of the percentage of sensors considered before (PS1=10) are sampled every second (TS1=1) and the rest (PS2=10) is monitored every 5 seconds (TS2=5), the maximum number of sensors becomes 833.

Consequently, in order to be able to connect more sensors to a microprocessor the percentage of time that the processor will use for sequential detection (RF) must be increased, or the sampling must be made less frequently (higher TS). The parameter PS cannot be controlled by the designer: PS will be small (PS will tend to zero) if the plant is running smoothly and PS will be big (PS will tend to 100) if there has been a transient or abnormal operating conditions, because many thresholds that initiate the StT will be exceeded.

Conversely, given the number of sensors assigned to processors located in the nodes of the network (obtained from Chapter 3) the sampling rate can be calculated.

5.5 CONCLUSIONS

In this Chapter a sequential detection algorithm, the Sequential t-Test, has been introduced. This test decides if the parameter observed from the sequential measurements obtained from the sampling procedure is contained within a specified interval set a priori by the user or the designer. For example, it would determine if the mean of a certain temperature being measured in some subsystem of the plant is consistent (ie its values are inside an interval specified by the user for a certain probability of making a mistake: taking a valid measurement for a wrong one or viceversa).

Note that the concept of sequential detection involves the use of a probabilistic interval as opposed to a rigid fixed interval based on upper and lower limits.

A fixed interval validation procedure detects those instances on which the measured parameter is outside the limits indicated by the interval. The advantage of this procedure is that it is almost instantaneous (it only requires one comparison and a few logic steps) and gives the indication that some 'abnormality' has occurred. However, the big dissadvantage is the amount of false alarms generated because the parameter has only temporarily gone out of bounds. In this procedure, the seriousness of the abnormality is determined by the width of the interval. The usual procedure to compensate the amount of false alarms is to identify a series of zones in which the measured parameter can stay, for example, green, yellow and red zones. The parameter can stay freely inside the green zone, it may be a fixed number of times in the yellow zone (ie per minute) and, in case that this fixed number is exceeded or that it goes to the red zone, an alarm will sound and the operators will be alerted. However, a usual mechanism for the operators to bypass these false alarms is to increase the width of the intervals, effectively dimishing the validation procedure.

Another dissadvantage of this fixed interval procedure is that it does not make decisions, it can only detect a change and inform the operator that something has ocurred.

On the contrary, a probabilistic interval as set by Hypothesis Testing, determines the amount of false alarms from the very beginning and its primary purpose is to make decisions. In the case of false alarms, even when a parameter could be temporarily out of bounds, the only action to be taken is that a test will be initiated and, if it was the case of a very short (in time) excursion, the decision of the test will be that nothing went wrong and more measurements will be taken.

With respect to the decisions taken by the algorithm a procedure could be automatized to have meausurements identified as unreliable and disregarded after a series of checks are made (ie to repeat the test a number of times). In this case, the operator can be transparent to this automatized procedure if enough information on the process is available (redundant sensors or analytic redundancy calculations).

An implementable solution to the problem of signal validation may to incorporate both fixed and probabilistic interval methods into one algorithm. With this structure, a mild threshold can be specified (fixed interval) and, if the parameter exceeds its limits, a sequential procedure (probabilistic interval) can then proceed to identify if a permanent change has ocurred or not in the measured parameter.

A design that could incorporate both the Kalman Filter (Chapter 4) and the Sequential t-Test to provide a more powerful data validation is suggested in Chapter 7.

The Sequential t-Test algorithm was implemented in a programmable calculator and an example was given in Appendix B.

It was shown that the total time required for its implementation in a MC68000 microprocessor-based system is roughly one milisecond per iteration and that the total storage requires almost 800 bytes.

Finally, the maximum number of sensors to be connected to this unit and the rates at which they are sampled were calculated.

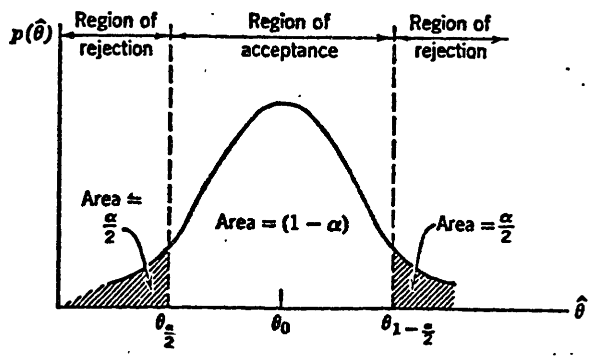
BIBLIOGRAPHY FOR CHAPTER 5

[AMS749] NATIONAL BUREAU OF STANDARDS, Applied Mathematics Series, Tables to facilitate the use of the Sequential t-Test, Number 7, 1949

[HANC66] HANCOCK, J., Sequential detection theory, 1966

[WALD47] WALD, A., Sequential analysis, John Wiley and Sons, 1947

PROBABILITY DISTRIBUTION FUNCTION OF AN ESTIMATE
****



From BAK, T., Mathematics for scientists, NY, 1966

FIG.5.1

CHAPTER 6

SURVIVABILITY MODELING

From Chapter 2 it may be recalled that one of the requirements of the system proposed to monitor the plant is that it has to tolerate single failures. As a consequence of this requirement some fault tolerant and fault intolerant techniques were introduced in the preliminary design shown in Section 2.4. Commercial designers often design without specifying either the threats that will act upon the system or the malfunctions that are to be prevented from propagating through the hierarchies of the system. Thus, in most of the commercial designs _validation_, the step after the design has been completed, becomes almost impossible (simply because there is nothing to validate against: if the failure modes are not characterized there is no possible verification of the behavior of the system).

In this Chapter an approach in the direction of validating the design will be taken by modeling survivability characteristics of the network presented in Chapters 2 and 3. Numerical results will be obtained.

Following the line of previous chapters it must be remarked that approximations about the true structure and characteristics of the system will be made and that the results obtained will be purposefully general. More detail will have to be included by the user of these methods to find results applicable to each case. For example, a single number to specify the failure rate of the units is not given, simply because that figure will depend on the technology at hand. Rather, typical numbers will be quoted and it will be possible to say that if a certain level of survivability is required, the failure rates of the individual units will have to be better than some values. These values can be found using the programs presented in Appendix G.

Finally and based on the results of this chapter, recommendations about the structure of the network will be given.

6.1 DESCRIPTION OF THE PROBLEM

The analysis of a system to be modeled requires the specification of

- the structure of the system,
- the appropriate measure of performance and
- the parameters that affect the measure of performance.

In the case treated in this Chapter two systems are modeled into three Models. The first system consists of replicated components located at the source and concentrator layers (multiplexers + processors at the source layer and concentrators at the concentrator layer, called distributed nodes in general). The second system corresponds to the whole network, including the previous two layers and the central computer.

Three different models are used because different measures of performance are required and different parameters influence these measures. Model I models the reliability, availability and other parameters of the distributed nodes by using a continuous-time discrete-state Markov model of the nodes. Model II models also reliability and availability but now it refers to the network as a whole, considering nodes, links and the central computer. A Markov model would have too many states to be considered for MODEL II, running times would be prohibitive for even small networks and, consequently, another approach is sought. By using a recursive algorithm the reliability and availability can be calculated efficiently. In Models I and II, the influencing parameters are the failure rate of the individual units, the repair rate of the repairman, the total number of units and the coverage (defined below). Model III, a queueing model, provides general results that treat in great detail the influence of repair in repairable components. The model can be applied to a system composed by units that exhibit the same failure rate and repair rates. The complete list of measures and parameters will be presented when the models are defined.

6.2 RELIABILITY AND AVAILABILITY MODELING AND THE CONCEPT OF COVERAGE

The term 'survivability' has been used in general to refer to the those characteristics of the system that will quantify the likelihood that the system's functions are performed as required. In particular, reliability and availability concepts will be used extensively in both MODEL I and MODEL II and other reliability measures will be introduced for MODEL III.

In this Thesis, <u>reliability</u> at time t will be used by defining it as the probability that the system is working at time t, given that it was working at time 0 with probability 1, and no repair has been made from a failed state to an operating state**. However, minor repairs (for operational states) are allowed. These minor repairs will be assumed to be executed by non-specialized personnel, and will typically require less than an hour (considering that the required person has to a) be available, b) look for a new spare unit, perhaps c) check some procedures, d) install the unit and e) check the behavior of the system afterwards.

The design team will have to consider provision for this kind of 'minor' maintenance from the beginning of the design. For this purpose, the (sometimes called) 'minor replaceable unit' or 'line replaceable unit' has to be designed. This will be the only unit that can be replaced by non-technical personnel, without affecting the normal work of the node. The maintenance to take the system off the failed state will be assumed to be provided by trained personnel, which may be split into two phases: during the first phase, maintenance personnel may check the node _from their facilities_, which are typically located far from the plant. For that reason appropriate links between the computers have to be provided. During the first phase, the malfunction and possibly the cause of the malfunction will

----

** This definition is applied to the nodes' reliability. For the whole network another reliability measure will be used, although the definition given above will still be applicable to the nodes modelled in the network.

be intented to be determined without going to the location of the system. Thus, appropriate measures could be taken or the required hardware to be used during phase 2 could be identified.

In case the problem can not be solved in the first phase, the second phase requires technical personnel to go the plant and solve the problem in situ. This kind of maintenance, which can be called 'major' maintenance, will take longer than the 'minor' maintenance.

Availability from time 0 to time t is defined as the probability that the system will be working in that interval of time, given that it was working at time 0 with probability 1, and repairs were allowed to take it from the failed state to an operating state.

These definitions of reliability and availability are somewhat standard. Some authors (Buzacott, for example, in [BUZA70]), have used the terms 'point availability' as an equivalent to reliability and 'interval availability' as synonymous to availability. Of course, as long as the concept is perfectly defined, either definition can be used.

In actual calculations (for example in the programs AVALNODE and AVALNET) it will be used the fact that the 'point availability' and 'interval availability' have the same numerical value as time tends to infinity. Thus, one integration will be avoided.

It can be seen from the definitions of reliability and availability that the conceptual difference is given by considering the 'major' repair or not. From Chapter 2, it may be recalled that the system has a 'minor' repair capability of reconfiguration (ie an operator or, more often, by a repairman which takes out a failed unit and replaces it with a new one), and that capability is <u>not</u> considered repair as it relates to the difference that exists between both concepts.

In the case treated in this Thesis, namely that of a control system installed in a power plant, major repairs will be possible and, consequently availability will be of interest. The reason for <u>recommending</u> the use of availability to quantify the performance of the system is because the influence of the repair rate is shown immediately in the results. Reliability may be used only in applications that do not have (or can not have) external, specialized repair. Note that reliability could also be defined not taking into account the minor repairs as well; that would be the situation of a computer in an unmanned spacecraft, for example.

Reliability results will be used to show the sensitivity of the design to certain parameters, because reliability values are more easily obtainable than availability values. However, the concepts presented in this Chapter are primarily intended to be applicable to the availability of the network.

After Bouricius et al, referenced in [BOUR71], the concept of <u>coverage</u> was used to quantify the complexity of the reconfiguration procedure. Before that, reliability models used combinatorial results or fault tree models that took into account the failure rate of the components that made up the system and, explicitly, the failure rate of the switching circuitry that switched spares on. However, when the function of reconfiguration was performed by software, it was necessary to use a new parameter to signify whether the reconfiguration procedure had been succesful or not, after a malfunction was detected in the system.

Formally, in reference [BOUR71], coverage is defined as the conditional probability that, given the existence of a failure in the operational system being modeled, the system is able to recover and continue information processing with no permanent loss of essential information. Consequently,

coverage = Prob[system recovers / detected malfunction]

To obtain the coverage that corresponds to a system: a) the designer has to specify tests that induce malfunctions to the system and write down different tests that detect those malfunctions, b) the designer has to create procedures that allow the system to identify its actual structure after the malfunction has ocurred (in terms of what elements are working and the amount of spares that are available), c) the system has to switch one non-failed spare on and change the status of the spare to the status of working unit. Hence reconfiguration is an involved process and,moreover, the functions performed by the system during the reconfiguration have to be transparent to the rest of the working 'healthy' processors.

6.3 MODEL I, MODELING OF THE NODES. POSSIBLE CONFIGURATIONS

In Appendix E, the modeling of systems by means of Markov models is treated. In this Section, the concepts described in Appendix E will be applied to model the nodes of the network. Some other parameters of the nodes will be found with Model III which uses a queueing model approach. Model II is dedicated to the network, and uses the results of Model I.

Many configurations can be analized, and they all will depend on which states are considered operational and which not, how many units are taken in account, from what initial state and to what terminal state transitions are made, etc. For example, in the case treated in this Thesis, the nodes are modeled as it is shown in FIG.E.1 (reliability) and FIG.E.4 (availability). Other authors have used similar architectures (see references [BEAU77] and [NG76]), although they did not explicitly treat the case of varying a) the number of operational states and b) the number of spares for different coverage values. These two cases will be considered numerically in the following two Sections. In the rest of this Section, the difference among the configurations will be explained.

The first difference that can be made involves the use of a different strategy for repair than the used in Appendix E. In Appendix E, it was assumed that the 'minor' repairs were made to take the system from a state with J operational units to another state with J+1 operational units. Another repair strategy that can be considered is restoring the node to its original state (all units working). This model was called 'Improved Model I'. However, the improvement is negligible and, consequently, unnoticiable in the performance of the system. Although it makes sense to replace all defective units at once, as suggested by the IM rather than one unit at a time, both models (Model I and Improved Model I) can be used interchangeably.

On the other hand, the terminal state of the 'major' repair is important, and the highest availability of the node will be obtained when the terminal state has as many operational units as possible. This last configuration can be seen in FIG.E.4.

Having settled the repair strategy to be used, the next topics to be treated involve an analysis of the number of operational units (working and spare units) and the states that are considered operational or failed.

Following the guidelines given in Chapter 2, it will be assumed that triplicated units will be the basic structure used for the nodes in the source and concentrator layers (see Chapters 2 and 3 about the nomenclature, also displayed in FIG.3.3). In addition, to these triplicated working units, a number of spares could be added so that, in case of failure of one of the working units, one of the spare units can be switched automatically by the reconfiguration algorithms running in the nodes themselves. None, one or two units can be considered to be spares (it will be shown that it is not necessary to consider more than 2 spares). Consequently, the number of operational units will be 3, 4 or 5.

What happens if three units are working, a malfunction ocurrs and there is no spare to switch on? Here is where a big difference (in terms of reliability and availability) can be made. The options that exist are, broadly, two. The first option assumes that the system may continue operation, although, perhaps, some funtions will not be performed anymore (for example, the testing strategy among units may change, although self tests can be performed as before). Due to this (possible) reduction in the performance of the node (that will have even less impact in the performance of the system), the situation will be called 'with degradation' (WD). It must be stressed that, depending on the functions that the node units has to perform, there may be no degradation at all. The second option assumes that the system becomes suddenly non operative because single failures, in the node, can not be detected anymore by comparing the malfunction syndrome among the three units. This case has been labelled <u>'without degradation'(WOD)</u>.

Combining the possible numbers of spares to be considered (three) with the degradation cases (two), gives six possible configurations to be studied.

6.4 SELECTION OF THE CONFIGURATION AND BASIS FOR COMPARISONS

The reliability of the six configurations mentioned in Section 6.3 will be analized in this Section by making use of the program RELCOMP, included in Appendix G. A sketch that shows the variables used in that program and the comparisons that were made can be seen in FIG.6.2. The models are drawn in FIG.6.3.

The program RELCOMP was used to make an initial investigation of the parameters that affected significantly the reliability of the model and, once those parameters were identified, to use more specific models for the analysis.

*   *
*   The first observation is that, allowing degradations, *
* always increases reliability, for the same failure rate, *
* repair rate and coverage.   *

*   *

This situation of having a better performance is not surprising. What is remarkable is the big difference that may occur by allowing degradations. For example, for a node with 3 operational units (3 working units and no spares), with a failure rate of 0.1 [failures/hour], a repair rate of 0.1 [repairs/hour] and a coverage of 0.99, the variable used to calculate the ratio between the reliabilities of the node WD and the node WOD is numbered 24 in FIG.6.2. Assuming that the system starts with all its units working at time 0, the reliability ratio is:

time [hours]	reliab.WD/reliab.WOD
0.9	1.3
10.0	16.0
20.0	200.0
30.0	2500.0
.	.
100.0	$1.0\ 10^{11}$
.	.
150.0	$2.8\ 10^{16}$

*   *

* The second observation is that the amount of spares *
* to be used depends on the mission time and the coverage, *

* for a fixed failure rate and repair rate. *

* *

That the amount of spares depends on the mission time can be verified by assuming two configurations (say one configuration with 4 operational units and the other configuration with 5 units) and varying the coverage to see when the reliability of both are equal. In the following table, the coverage will be specified with too many significant figures only to show that a small variation in the value of the coverage will imply a large change in the mission time (defined as the time that takes to both configurations reach the same reliability value). ** For units with a failure rate of 0.01 [failures/hour] and a repair rate of 0.1 [repairs/hour], the reliabilities have the same numerical value at

Time [hours]	with coverage	(see figure)
100.0	0.99277	FIG.6.4
500.0	0.991094	FIG.6.5

** Note that the mission time is very sensitive to changes in the coverage but, on the other hand, the reliability is almost insensitive to changes in the coverage. Thus, there is no purpose in specifying a mission time unless the coverage is known with great accuracy.

1000.0 0.99088 FIG.6.6

In the last two cases, the reliability of the system with 4 units is more reliable than the system with 5 units before the 'crossing' time (for example 500.0 hours for FIG.6.5). After the crossing time, the system with 4 units is less reliable. It must be noted again that to design or to measure the coverage of a system is not a simple task and it is doubtful that it will be possible to specify coverage to the levels of resolution that are indicated in the above list. Perhaps, it will not be possible to specify it better than 1%; doubtfully better than one per thousand. See reference [BAVU75] about the upper and lower bounds for coverage and methods of computing values for the coverage.

```
*                                                       *
*     The resolution of the procedure to specify the   *
* coverage would have to be very high to decide the exact *
* amount of spares to be used so that, for critical values *
* of the coverage, the reliability obtained from the   *
* system is guaranteed to be a maximun, for a specified *
* mission time.                                         *
*                                                       *
```

A closer analysis to the table presented above, shows that, for a fixed time, (say 500.0 hours) the node with 4 units will exhibit the same reliability that a node with 5 units for a certain coverage (0.991094), and the node (with

4 units) will be more reliable (than the node with 5 units) for <u>longer</u> missions <u>as long as the coverage decreases</u>. For <u>higher</u> values of the coverage, the mission time for a node with 4 units is <u>shortened</u> (with respect, always, to a node with 5 or more units). This behavior, as it may be expected, does not only occur with 4 and 5 units but it was verified with smaller number of units (3 and 4 units). An explanation is provided in the next Section.

From this observation, it can be concluded that

```
*                                                            *
*    the lower the coverage, the lower the number of units   *
* that can be operational to maximize the reliability of     *
* node.                                                      *
*                                                            *
```

If the minimum number of units to be initially working in a node is 3, as considered before, another way of stating this conclusion is that

```
*                                                            *
*    there are situations in which a node will be more       *
* reliable if it has less spares (or even no spares) than    *
* another node with more spares, and those situations will   *
* depend on the numerical value of the coverage.             *
*                                                            *
*    Thus, the presence of spares may decrease the           *
* reliability of the node.                   ********        *
*                                                            *
```

This result has been faintly mentioned in the literature [BOUR71] and no stress has been given to it, although the more costly aspect of indiscriminate replication of the number of components has been the basis of many designs since then.

Perhaps one of the reasons for not emphasizing this observation is that it was known since the early '70s that the coverage has to be very close to one to be able to talk of a very reliable system, and most of the authors directly set the coverage equal to one when doing their analysis. In other cases they made the analysis for only one set of units and did not consider varying simultaneously the number of units and the coverage.

A summary of the results obtained so far can be observed in figures FIG.6.7, FIG.6.8 and FIG.6.9. In those figures, the reliability of four configurations is presented. One of the configurations, labelled 0S (zero spares) in the figures, represents three units working in parallel, with no spares, and it also assumes that, if one of the units fail, the node will be considered failed ('no degradation'). This configuration corresponds to the upper left model in FIG.6.2. The rest of the configurations assume that degradations are allowed: the first model assumes also no spares, and is labelled 0W (no spares <u>w</u>ith degradation), the second model considers one spare and is called 1W and the last model works with two spares and is identified as 2W. These last three models correspond to the lower models shown in FIG.6.2. For the purposes of showing the desired effect, the nodes are assumed to work with a failure rate of 0.1 [failures/hour] and a repair rate of 0.1 [repairs/hour].

In the first figure, FIG.6.7, a low value of coverage is used to show that allowing degradations increases the reliability of the node (0W, 1W and 2W curves exhibit a higher reliability than the node represented by 0S, for all times after time 0), and that the _no_ _spares_ configuration is _preferrable_ to the one or two spares solution.

The next figure, FIG.6.8, shows that a much higher value of coverage, 0.99, reverses the situation, namely, for configurations with 2 (1) spares will exhibit higher reliability than configurations with one (none) spares. Note also that the reliability of the node is now much higher than before, and this is due to the higher coverage.

An intermediate value of coverage must exist where nodes with, say, two spares have the same reliability as nodes with one spare. This equality is found when the coverage equals 0.87 as illustrated in FIG.6.9. If the figure is carefully examined, it will be noted that the notation written near the top of the figure and between the time 1 hour and 2 hours indicates that the no spares configuration (0W) is more reliable than the other nodes with spares (1W and 2W). However, the notation located at coordinates (60 hours, 0.13 reliability) shows a condition for what two spares are preferred to 0 spares (always with degradations allowed).

It is not obvious that a node with no spares may be more reliable than the same node with spares. Therefore, the following discussion is provided to explain the reasons by analyzing the extreme values that the coverage may adopt, and its influence on the reliability of the nodes.

The configurations selected to explain this behavior are the center and right models in the upper line of FIG.6.2, namely those with 4 or 5 operational units (1 or 2 spares) and not allowing degradations, in order to make the explanation simple. The comparison is shown in FIG.6.10 for the four cases, ie the combination of having one or two spares and a coverage of 0 or 1.

For the lowest possible value of coverage (zero), consider the first case (upper-left), in which the node is composed of 4 units in series compared to the second case (upper-right), in which the node's behavior is that of 5 units in series. It is known that the more reliable system is the one with less units, thus, for the minimum coverage, the upper-left node structure is preferred. Therefore, when the coverage is zero, a configuration with n+1 units is <u>less</u>

reliable than a node with n units.

The concept of 'series' configuration is used in this Section as a valid analogy although the units will not be electrically connected in series. The term 'series equivalent model' will be used to indicate the analogy. Actually, in the operation of the system, the units will be using the same inputs and software voting will be provided to the outputs. These triplicated units, voting bit by bit (or every few bits) their outputs, are indicated with the letter 'W' (for Working) in FIG.6.10. The spares will be indicated with the letter 'S'. In the configurations with zero coverage, a malfunction will inhibit the reconfiguration procedure and, therefore, in the analogy it is assumed that the units are connected in series because, in a series system, if one unit fails, the system is declared failed.

In a configuration with a coverage of one, malfunctions <u>can not</u> inhibit the reconfiguration which is always assumed to be completed. A system with n operational units (3 units working and n-3 spares), after reconfiguration, will end up working with n-1 units (3 units working and n-4 spares). Therefore, it can be imagined that, during the life of the node, n-3 configurations (of coverage equal to <u>zero</u>) were used, although not simultaneously. An imaginary switch is included in the output of the 'series equivalent model, to show that, when the node begins to work, the switch is connected to the system that has n units and, after a malfunction occurs, the switch will be connected to the next configuration, which has one less unit. Therefore, instead of declaring the node failed when the malfunction occurs as it is made in the upper models of FIG.6.10, the node is allowed to work in the new configuration, effectively increasing the life of the node. In the following paragraphs, the case of coverage equal to one will be treated for nodes that initially have 4 and 5 units.

The highest value of coverage is one (ie, every malfunction is succesfully managed and there is only one transition from the operational states to the failed states) and two cases will be considered. The case shown in the lower-left corner corresponds to a node that can only make a transition to a new structure with 3 working units. In this case it is assumed that the system starts with four units working before any malfunction occurs. Thus this behavior can be thought to be similar to a node of 4 units working in series such that, in the event of a malfunction, the node can be reconfigured to a node of 3 units working in series. This case may be compared to a structure (lower-right) that allows 5 units (in series) be reconfigured to 4 units (in series) and then to 3 units (in series) before executing a transition to the failed state. Evidently, because of the extra mission time represented by having the initial 5 units on line, this configuration is more reliable than the node presented in the lower-left corner. Thus, when the coverage equals one, a system with n+1 units is <u>more</u> reliable than a system with n units.

Consequently, given that for a coverage of 0 one of the configurations is more reliable than the other and that the situation changes when the coverage equals one, there must be some intermediate value of the coverage in which both structures exhibit equal reliability. Clearly also from FIG.6.10 the increase in coverage will give an increase in reliability with the added expense for the capabilities of reconfiguration.

The analysis of this counterintuitive observation provides an even more interesting result. This result will permit, in a single graph, the integration of the parameters failure rate, repair rate, coverage and total number of operational units.

Doing the analysis of nodes with a different number of operational units (3, 4 and 5), it was noted that the <u>ratio</u> of the parameters failure rate and repair rate uniquely defined the coverage at which the reliability of a node with any number of units matched the reliability of a node with one more (or one less) unit. Figure FIG.6.11 indicates these results. For example, for a node with 3 operational units with a failure rate of 0.01 [failures/hour] and a repair rate of 0.1 [repairs/hour], if the coverage is less than 0.9828, the node, as it is, will be more reliable than the same node with 4 operational units (3 working and 1 spare).*

In figure FIG.6.12, it is shown that the amount of spares needed depends on a) the ratio of failure and repair rates and b) the coverage. The curve 34 indicates the coverages that make 3 and 4 operational units exhibit the same reliability at the times indicated in FIG.6.11. The curve 45 shows the same behavior for 4 and 5 operational units. The region below curve 34 corresponds to situations in which the node with 3 units is more reliable than the same node with more units. The region between curves 34 and 45 indicates the situation when it is more reliable to have 4 units. In the upper region, 5 units will be preferred.

It can be observed that, for values of the failure rate/repair rate applicable to individual units of a computer system (0.0001 < FR < 0.001 and RR = 1.0), <u>the coverage will have to be extremely high in order to justify</u>

---

* The reliability is calculated at (1/3*failure rate) hours, to establish a basis for comparison although it can be observed that, for some situations, the reliability will not change at all for any time considered (FIG.6.6) or it will not change appreciably in the time span of interest (FIG.6.9).

the inclusion of spares. In most of the cases, having three units working in a triad configuration (same inputs and software voting on the outputs) will be enough.

It can be seen that the assumption of having 3 links going from the multiplexers (located in the source layer) to the concentrators (in the concentrator layer) is reasonable and mathematically justifiable, because the failure rates of the cables are very low and their repairs (actually replacement) do not take too much time.

6.5 MODEL II, MODELING OF THE NETWORK

In Section 6.2 the reliability and availability of the nodes has been defined and their formulation and solutions were presented in Appendix E. With respect to the network another reliability measure will be introduced.

The preliminary design of the network presented in Chapter 3 will be used in this and the following sections. It must be recalled that the configuration consist of n nodes (see FIG.3.3) connected, by means of fiber optic links, between themselves and to a central computer which, for the purposes of this analysis, will be considered one more node. Thus, effectively, the new network will consist of n+1 nodes. It will be assumed that the configuration is fully connected (also called 'complete graph') as the examples of FIG.3.6 indicate. Consequently, there will be n(n+1)/2 (non-triplicated) links.

It will be said that a link is working if there exists at least one connection through it between the nodes on which the arc is incident, otherwise, it is 'failed'. The probability that the link incident on nodes j and l is working will be indicated p(jl) and the probability that it has failed will be q(jl)=1-p(jl). Link failures will be assumed independent. Note that the links considered are those that are called yjl and zjo in Chapter 3.

The term 'reliability measure' will be used to indicate a derived parameter, function of the input parameters failure rate and repair rate of the units, number of units, failure rate of the links and number of links. Many reliability measures could be used to quantify the survivability of a network, for example a) the probability that a specified number of nodes is not disconnected from the network, b) the average number of failed units or c) the average number of failed links.

The objective of this Section is to calculate the probability that the whole network will not be disconnected into 2 or more subnetworks by failures of the links or the nodes. This objective will be the 'reliability measure' of interest.

The following two definitions will explain what is understood by reliability and availability of the network in this Thesis.

When reliability (as defined in Section 6.2) is used for the nodes of the network and the links are not repaired if they fail, the probability that no node is disconnected from the network will be the reliability measure of interest. This measure will be called _network reliability_.

When the availability (as defined in Section 6.2) is used for the nodes of the network and the links admit repair, the probability that no node is disconnected from the network will be the reliability measure. This measure will be called _network availability_.

It will also be assumed that every failure rate and repair rate corresponding to the units that conform the nodes and the links are independent. However, the assumption that the repairs of different nodes are independent will be relaxed in the queueing model of the network (MODEL III).

One more assumption will be that links are identical. Thus, the same failure rate and repair rates will be applied to every link in the network.

The algorithms used and detailed in Appendix F will be based in a work made by Buzacott in [BUZA80]. The program to calculate the reliability and availability of the network are called RELNET and AVALNET, respectively, and they are included in Appendix G.

In order to show how parameters impact on the availability of the network, standard numerical values will be used. Results will be presented for a network with the following typical set of parameters:

NODES

Number of units per node	3
Failure rate of the units [fail./hour]	0.001
Repair rate (minor) [rep./hour]	1.0
Repair rate (major) [rep./hour]	1/24
Coverage	0.95
Number of nodes (including the central computer)	5

CENTRAL COMPUTER

Failure rate [fail./hour]	0.00001
Repair rate [rep./hour]	1/10

LINKS

Failure rate [fail./hour]	0.00001
Repair rate [rep./hour]	1/24

The parameters corresponding to the central computer have been selected so that a steady state availability of 0.9999 would be obtained for the central computer, and they have been obtained from FIG.F.1.

The results of the availability analysis for this example is shown in FIG.6.13 (output from AVALNET) and FIG.6.14 (plot from AVALNET).

It can be observed that the steady state results do not indicate extraordinarily high availability: 0.9963 for the nodes and 0.9951 for the network**.

The availability of the network can be upgraded if some of the parameters are modified. In particular, the following modifications were made (and the results are included):

** The characteristic that the reliability of the network decreases as the number of nodes increases has a simple explanation. The reliability measure for the network has been indicated to be the probability that the network is not disconnected. Thus, as the number of nodes increases, the probability that a node can fail is increased. Consequently, by definition, the probability that the network will fail is also increased. This phenomenon is similar to the decrease in reliability in a series system as the number of units increases. A simple proof of this assertion and the parameters from which it depends can be obtained from the equations given in Appendix F.

		Steady state availability		
Failure rate	Coverage	Node	Network	Plot
(0.001)	(0.95)	(0.9963)	(0.9951)	(FIG.6.14)
0.0001	0.95	0.9996	0.9985	FIG.6.15
0.001	0.99	0.9991	0.9965	FIG.6.16
0.0001	0.99	0.99993	0.9996	FIG.6.17

A failure rate of 0.0001 [fail./hour] and a coverage of 0.99 are parameters that indicate a high quality design. Even with these values the availability of the system is not higher than 0.999 unless both parameters are simultaneously considered. It will be shown immediately that the availability of the network is very sensitive to these parameters (small changes in the failure rate and the coverage imply significant changes in the network's availability). The specification of these parameters could be made stating that it is required, for example, a failure rate better than 0.0001 for individual units and that the coverage should be no less than some value representative of the state-of-the-art (see [BAVU75] for some of these values and how to calculate them - in 1975 -). Even so, the task of the designer will not be easy to assure that his design parameters comply with the specifications.

In order to know how the rest of the parameters affected the availability of the nodes and the network, those parameters that could be more critical to be specified were considered. They include the failure rate and the coverage (whose influence was noted in the last example), the (major) repair rate and the availability of the central node. The rest of the parameters do not impact significantly the availability of the design. The cases were considered with respect to a reference case, whose defining parameters are:

NODES

Number of units per node	3
Failure rate of the units [fail./hour]	1/3333
Repair rate (minor) [rep./hour]	1.0
Repair rate (major) [rep./hour]	0.5
Coverage	0.95
Number of nodes in the network	5

CENTRAL COMPUTER

Availability	0.9999

LINKS

Failure rate [fail./hour]	0.00001
Repair rate [rep./hour]	1/24

These parameters correspond to a 'better-than-standard' computer system with a modularization that allows not lengthy repairs. In particular, the central computer availability is exceptionally good for a commercial system.

The parameters that were modified are a) failure rate to 0.001 and 0.0001 [fail./hour], b) coverage to 0.9 and 0.99, c) (major) repair rate to 1 and 1/24 [rep./hour] and d) the central computer availability from 0.99 to 0.99999. The results are presented in FIG.6.18 where the availability of the nodes and the network are plotted.

From the observation of FIG.6.18 the following conclusions can be made:

- the network availability can be no better than the availability of the _least available_ node (distributed or central computer) for the ranges of the parameters considered.
- changes in some of the parameters may produce significant variations in the availability of the network _only_ when the availability of the central computer is very high. Thus, if the central computer is not very reliable, it is unnecessary to improve the availability of the individual nodes, because the network availability will not change appreciably.

- if the individual nodes are unreliable, there is no purpose in improving the central computer (although the cost of the configuration will increase dramatically). Thus, it represents no solution to use a fault-tolerant computer as central node if the components that are appended to that computer are unreliable (ie non replicated processors).

For the cases presented in Chapter 3, the use of the last set of variables give the following availabilities for both the INITIAL and the GROWTH scenarios:

	4 links	5 links
node	0.9999004	0.9999004
network	0.998602	0.998503

It can be observed that the difference between the steady state availabilities of the case with 4 links and 5 links will be unnoticiable in actual operations. However, those numbers can give an idea of the levels of survivability that can be expected unless not-standard provisions are considered. (Standard provisions are a single bit for parity checking but not error correction and detection, a duplicated component but not a triplicated unit with spares, redundant links but not a network of links, diagnostic software packages but not dynamic reconfiguration of the system, etc). If the requirements specifying the availability indicated a minimum of 0.99 for the network, the design accomplished its objectives with respect to this point (subject to validation). However, if the requirements indicate that the minimum availability should be 0.999 or higher, further work is required.

A minimum availability of 0.999 for the network will be attainable with a little effort, although that level will not be attained with standard components, ad-hoc design methods and a minimum of fault tolerance.

A minimum availability of 0.9999 will be much difficult to attain and to validate. Significant efforts should be undertaken to prove that a system with that level of survivability can be effectively designed.

6.6 MODEL III, QUEUEING MODELING

In previous Sections some reliability measures of the system proposed in Chapter 2 have been calculated. Other reliability measures are very difficult a calculation with the models presented (for example non-exponential distributions) or even to model (ie it has been assumed that the repairs in different nodes were independent, although with only one repairman that would hardly be the case). The model presented in this Section (MODEL III) will allow to calculate some more parameters that could be used to specify a computer system. For the purpose of modeling, results of queueing theory will be used.

That a computer system can be modeled by using queueing models is not new. However, one of the models needed to understand the behavior of the system invokes multiple repairmen. Also, the interrelation of the parameters make it difficult to understand how the modification of one parameter will affect the results (ie the sensitivity of the results to the modeling parameters). For the ranges of failure rates and repair rates that will be used in the computer system presented above there is a graphical way of presenting results so that tradeoffs can be made inmediately. Graphical results are common for systems with only one repairman, but the author is not aware of similar methods for multiple repairmen.

Another difficulty is that it is not always clear what parameters are input and which can be considered as output. For example, for the designer that designs the units from scratch, the specification for the variable failure rate of the units will be an output, because he will have to design a unit with a specified failure rate and the rest of the parameters will be rather fixed. However, if the designer works with off-the-shelf units, the failure rate of the units will be fixed, and other parameters will have to be variable so that tradeoffs can be made. The graphical procedure allows one to make iterations without any difficulty, and any parameter of the system can be input or output.

Since there may be situations when the system may require more than one person to repair failed units, the possibility of multiple repairmen is considered for the case of 'minor repairs' (see Section 6.2). However, one of the requirements presented in Chapter 2 was that the maintenance and repair of the system has to be simplified to a maximun and, thus, a minimum of personal for repair is required. For this reason, the ideal minimum is to have only one repairmen and situations will have to be identified so that this objective is accomplished.

The purpose of this Section is <u>not</u> to introduce the reader to the concepts of queueing theory which can be obtained by consulting many references such as [KLEI67] and [ALLE78]. Rather, the application of those concepts will be presented to model some parameters that are very difficult to analize analytically with other theories or that would require lengthy Monte Carlo simulations ** to produce similar results.

The principal concepts necessary to understand the queueing system presented in this Section can be obtained from FIG.6.19, which introduces the nomenclature to be used.

The computer program QWR (Queue With Repairs), included in Appendix G, was used to generate the results used a posteriori for the graphical solution.

The units that are modeled with this MODEL III are those units that can be under 'minor repairs' and one of the characteristics of those units is that they have to be subject to the same failure rate ALFA and the repairmen should be able to repair them in the same average time 1/AMU. Thus, physically different units could be modeled with this MODEL III. Consequently, if the characteristics of the replaceable units in the source layer and the concentrators in the concentrator layer are similar (see layer notation in Chapter 3), they all could be added up to be the 'units' by the model.

David Kendall (see reference [ALLE78]) developed a shorthand notation that will be followed to describe queueing systems. The Kendall notation has the form

---

**It is noted that a useful characteristic of Monte Carlo simulations is that they produce a myriad of results but at the expense of a multitude of runs. Even more results could been obtained with a Monte Carlo simulation than with queueing models, although not with the simplicity and the possibility of making quick tradeoffs as it is offered in this Section.

A/B/KC/K/m/Z. Here, A describes the interarrival time distribution to the 'queue' of FIG.6.19, B the repair time distribution, KC the number of repairmen, K the maximun number of units allowed in the queueing system (also called system capacity), m the number of units in the environment that provides the queueing system with a continuous flow and Z the queue discipline.

The time distribution used for the symbols A and B that is of interest to this work is the exponential distribution because of the Markov or 'memoryless' property of the distribution. Thus, if the distribution is exponential, the expected time remaining to complete the service of a unit (for example the repair of a unit) is independent of the service already provided. The use of the exponential distribution for the operation and the repair of the units is indicated by replacing A/B by M/M. The maximun number of units allowed in operation, in the queue and in repair, in the cases at hand, equals the number of units in the plant (which has units in operation and failed units waiting for repair) plus those units that are being repaired (with the repairmen in the plant making the replacements). If the spares are in 'hot standby' (electrically connected to the system to detect the failures and to minimize the time to reconfigure the node) the spares will be considered in operation too. Thus, the total number of units will be K, the maximun number of units allowed in the system. The queue discipline (the rule for selecting the units to be repaired) is assumed to be first-come-first-served and the omission of any symbol in the corresponding Kendall notation indicates that this type of discipline has been selected.

Consequently, the above notation indicates that the queueing system of interest to these discussions can be written M/M/KC/K/K.

The system works in the following way: units are working in nodes of the network and if a unit fails it enters into a 'waiting list' for repairs (queue). If there is a repairman available (not repairing another unit), the unit will initiate the repair procedure which will last an average of 1/AMU hours. If there is no repairman available, the unit will wait until the first repairman finishes with his previous repair. When a unit has been repaired, it is returned to be installed in a node. Note that there is no requirement that the <u>same</u> unit has to go back to the node; the only specification is that a unit has to wait (to be identified and replaced) and that the repair (replacement) has to last an average of 1/AMU hours.

The parameters discussed in the previous paragraphs are indicated in FIG.6.19. The model works with random variables that are defined by the distributions assumed in the model, in this case exponential distributions. The averages are the expected values of those random variables.

Reliability measures that are defined for the queue are ALQ (the average number of units to be repaired), WQ (the average waiting time in queue to be repaired considering those cases in which the units do not wait for repair and those cases in which the units have to wait for a repairman to be available), D (the probability that a unit has to wait to be repaired) and EQ (the average waiting time only for those units that <u>must</u> wait, ie, there is no repairman available). For the queueing system (queue and repair sections of FIG.6.19) the following reliability measures are used: AL (the average number of units down (ALQ + units in repair)), W (the total average waiting time (WQ + time in repair)) and ALAMBDA (the rate of incoming units to the queueing system).

Typical parameters were used to calculate the above reliability measures with the equations shown in FIG.6.20 (from reference [ALLE78]) and implemented in the program QWR (included in Appendix G). The results from these calculations were incorporated into one nomogram (FIG.6.21).

In the nomogram, the same scales are used for the variables W and WQ and for the variables L and LQ. An example will be useful to show how to work with it, the results obtained and the accuracy that it may be expected from those results.

EXAMPLE:

assume that a network has 7 nodes and 5 units per node (K=35), that the failure rate of the units is 0.001 [fail./hour] (ALFA=0.001), that there are 2 repairmen (KC=2) and that each one can repair one unit in an average time of 10 hours (AMU=1/10).

The following procedure has been used to calculate the parameters presented above in order to compare the results from the nomogram with the results calculated by using the QWR program.

Procedure:

- first, calculate the ratio ALFA/AMU (0.01);
- second, with ALFA/AMU, KC=2 and K=35 find ALQ in the scale named L(q) (it gives ALQ=0.009);
- third, on the same graph, but on the lower vertical scale to the right, find D (D=0.05);
- fourth, with ALFA/AMU and AMU (0.1) find the intermediate variable x (x=0.001) and then, with x and K=35, find ALAMBDA in the horizontal scale labelled λ (ALAMBDA=0.035);
- fifth, move the value of ALAMBDA to the vertical scale λ and, with the previous value of ALQ=0.009, find WQ (WQ=0.23);

- sixth, move the value of D=0.05 to the upper vertical scale D and, from that point and with the previous value of WQ=0.23, find EQ (EQ=4.7);
- seventh, as W=WQ+(1/AMU) the same scale that represents WQ, can accomodate W, because the only difference is the addition of the inverse of the repair rate. Thus, W=0.23+(1/0.1)=10.23, which is represented on the same scale;
- eighth, as done with W and WQ in terms of representing both parameters on the same scale, the same occurs with L and ALQ **, Then, both L and ALQ can be represented on the same scale (called for that reason L(q)) and, with ALAMBDA=0.035 and W=10.23, the graph gives AL (AL=0.4).
- ninth, and for the purpose of this analysis, the same parameters are used as input to the program QWR, and the previous graphical results are compared to those obtained from this step. The results are shown below:

** because of the relations (AL=ALAMBDA*W and ALQ=ALAMBDA*WQ) known as Little's formulas.

STEP	PARAMETER	GRAPH	QWR	ERROR(%)
1	ALFA/AMU	-	-	-
2	ALQ	0.009	0.00975	-7.7
3	D	0.05	0.05008	-0.001
4	ALAMBDA	0.035	0.0346	1.2
5	WQ	0.23	0.28	18.0
6	EQ	4.7	5.62	-16.0
7	W	10.23	10.28	-0.5
8	L	0.4	0.35	14.3

It can be seen that the error is not significant for some parameters although it may be as high as 20% for others. The graph should be used to estimate those parameters that can not be directly calculated from the available equations (for example trying to obtain the failure rate of the units or the number of repairmen as a function of the probabilities D and EQ). Once the parameters are identified, a more exact calculation can be made with the program QWR, but note that the QWR program only accepts as inputs ALFA, AMU, K and KC as inputs.

The parameters calculated by either of the two methods presented may be used to specify the characteristics of the computer system. For example, one of the most critical parameters of interest is the number of repairmen. The organization of the program QWR can be used to find this variable only by iteration. With the use of QWR, FIG.6.22 was calculated. In that figure, the inputs are the failure rate and the repair rate of the units (ALFA and AMU), and two significative parameters are found as a function of the number of repairmen (KC): the probability that a unit has to wait for repair (D) and the time that a unit who <u>must</u> wait has to stay in the queue to be repaired (EQ). If the repair time is small compared to the waiting time in the queue, the parameter EQ will be of interest; otherwise, if the waiting time in the queue is small compared to the repair time or if both times are comparable the total average waiting time (W) will be used. In the case at hand, the repair times are small with respect to the queueing times, then, EQ will be used.

The behavior of other parameters (ALQ, L, WQ and ALAMBDA) was also investigated, but the only parameters that appeared critical from the point of view of the survivability of the system were D and EQ, which could, thus, be used to specify the system.

To specify the system to the designer (see Chapter 2) limits for some of the parameters could only be given by engineering judgement. Since the units of a given computer system should not wait too long to be repaired, it seems reasonable to say that a) the probability that a unit has to wait to be repaired should not be larger than 1% (although perhaps saying less than 5% would be enough) and b) units should not wait for repair longer than the average time that takes to repair one of the units (EQ=1/AMU).

With these ideas in mind those parameters that could represent the desired system can be extracted from FIG.6.22. The figure was calculated for a total of 20 units in the system.

As an example, assume that D=0.01. If the average repair time is one hour (AMU=1.0 hour) then EQ is 1 hour. If the desire is to have only one repairman, from the figure, those parameters that should be required for the units and the repair service to comply with the above requirements are ALFA/AMU better than 0.001 and AMU better than 1.0 [repairs/hour]. Consequently, a failure rate higher than 0.001 with a repair service that takes more than 1 hour will imply that the probability that a unit has to wait will be much higher than 1% (for example a failure rate of 0.01 [failures/hour] and a repair rate of 0.1 [repairs/hour] will give a probability of waiting of almost 20%), although the expected time EQ will only be a little more than before (1.15 hours).

A set of conclusions can be extracted from the observation of FIG.6.21 and FIG.6.22.

- The performance of the computer system can be succesfully modeled as a function of the parameters that were used in Sections 6.4 and 6.5, in particular as a function of the ratio of the failure rate to the repair rate (see figures FIG.6.12 and FIG.6.21). Those parameters can be used to specify the system to the designer or, once the system is built, to make modifications in the existing system. The usefulness of modeling the system with these parameters is that they can be verified during actual operation by appropriate measurements.
- The parameters that define the behavior of the queueing system can be represented graphically in the range of interest of the input parameters, and that nomogram can be produce immediate results for non obvious tradeoffs.
- The probability that a unit has to wait for repair (D) and the expected time in queue for the units that must wait (EQ) appeared as relevant parameters to a) specify, b) design and c) validate the reliability characteristics of a computer system.

- As verified with the previous concepts of node and net availability, there are ranges in which the modification of some of the parameters do not impact the behavior of the queueing system significantly and there are ranges where the impact is important. Note that a reduction in the failure rate in FIG.6.22 will reduce the value of D significantly, although the variable EQ will almost be unchanged for a failure rate/repair rate ratio lower than 0.001.
- Since it has been considered that 'minor repairs' are those repairs modeled by the queueing system and it was shown that only one repairman could be enough to effect the repairs (which will consist mainly in replacement of the failed units), the repair service will not affect the structure and functions of the personnel dramatically. **

6.7 CONCLUSIONS

In this Chapter the concepts of Markov modeling, graph

** For the example shown in FIG.6.22, when ALFA/AMU equals 0.001, the rate at which the units arrive to the queue is 0.002 [units/hour], or an average of 1 unit per 500 hours of operation (1 unit every 20 days). These numbers do not seem to impose a real burden to the operators.

theory and queueing theory have been applied to the evaluation of survivability concepts of a computer system. The network presented in Chapters 2 and 3 has been analyzed with these models. The network is composed of links that connect nodes. These nodes are composed by a number of units, each at least triplicated.

In a broad sense, the term survivability has been used to indicate reliability and availability. In particular, the difference between reliability and availability has been assumed to be that the existence of repair from the failed state is included in the calculation of availability. The survivability for the network has been defined as the probability that no node is disconnected from the rest of the network. When reliability is used to model the nodes of the network, the survivability of the network is called 'network reliability'. When availability is used to model the nodes, the survivability will be named 'network availability'. Since in this Thesis it was assumed that the network would be implemented in a nuclear power plant, repair will always be available and consequently, availability will be of interest.

Three models have been studied: MODEL I, MODEL II and MODEL III.

MODEL I is a Markov model of the survivability of the nodes.

MODEL II solves the survivability problem of the network, applying the results from MODEL I for the nodes and modelling the network as a graph. The links and nodes of the graph are subject to failures.

MODEL III is a queueing model of the system, which considers the repair of all the units (that make up the nodes) that are in the network.

MODEL I showed that the reliability of a node was increased if, instead of declaring it failed when less than three units were working, the node was allowed to continue in operation, situation that was named 'degradation', because some functions that were routinely performed with three units in the node could not be accomplished anymore. The model also showed that the parameter 'coverage' , defined as the conditional probability that if a malfunction occurs the units will detect it and a spare will be switched on affecting a succesful reconfiguration, is the decisive parameter to determine the amount of spares that the node will be allowed to have ready for automatic recovery. In particular, the model showed that there are cases in which the reliability of the system decreases if the number of spares is increased, counterintuitive argument that was shown to have a simple explanation.

It was shown that varying the ratio of the failure rate to repair rate, along with the coverage of the units, allows determination of the amount of spares that are required to maximize the availability of the node.

The model also showed that the availability goes to a steady state. It will be assumed in the future that this steady state availability will be used when reference is made to availability. Note that another scheme to have a reliable system could be to 'reset the system to time zero' by changing all failed units in the nodes when the availability crosses a certain level. However, since the availability decays very fast in the first (roughly) 100 hours, a little delay in that maintenance procedure will prevent the unit from keeping within Consequently, such a maintenance procedure will not be recommended as viable.

MODEL II showed that the availability of the network was lower than the availability of the nodes because of the definition used for the network survivability and for the ranges of the parameters considered.

It was also found that the availability of the network is less than the availability of the least available component (node or central computer). The recommendation of designing with high quality components (for the nodes and the central computer) is obvious from this result.

Another result from MODEL II is that, if some of the components are unreliable, very little will be gained in the availability of the network by modification of some other parameters.

The failure rate of the units, the coverage and the availability of the central computer are the parameters whose changes most affect the availability of the network. The availability of the configuration proposed in Chapters 2 and 3 was shown to be 0.9985 ('5 links' case) and 0.9986 ('4 links' case, see nomenclature in Chapter 3). It is also found that:

- a network availability of 0.99 was readily obtainable,
- a network availability of 0.999 was attainable with efforts, and
- a network availability of 0.9999 was possible by concentrating significant efforts in the survivability of the system, and that such a design was difficult to validate (considering that the requirement of maintaining low costs is kept).

In order to be analyze the repair system (or 'service') MODEL III was used. The usual assumption of independence of the repair distribution among nodes can be relaxed with this model.

A graphical method for analyzing the system was developed and shown to be useful to perform tradeoffs.

It was observed, very much as found with MODEL I, that the ratio of failure rate to repair rate was a parameter that could be used to specify characteristics of the queueing system.

Given the failure rate, the repair rate and the number of units, other parameters that define the characteristics of the system can be found. For example, estimates can be made for the number of repairmen that are necessary to maintain certain variables below certain levels such as the probability of a unit waiting for repair (D) being less than a few percent and expected time in queue for those units that must wait (EQ) being less than the average time to repair one unit. The use of these variables 'D' and 'EQ' will allow the determination of the quality of the repair service. The result of the analysis on the number of repairmen showed that one repairman could suffice if the failure rate of the units is kept lower than 0.001 [failures/hour] and the repair rate higher than 1.0 [repair/hour]. This is a condition that could be accomplished with modularization of the components and consideration of the repair strategies from the beginning of the design.

Similarly to the findings of Section 6.5 with respect to the sensitivity of the network availability to changes in some parameters, it was found that some parameters were very sensitive to changes while others were not, depending on the magnitude of the variables failure rate, repair rate and their ratio.

BIBLIOGRAPHY FOR CHAPTER 6

[BAVU75] BAVUSO, S., Impact of coverage on the reliability of a fault tolerant computer, Report NASA TN-D-7938, September 1975

[BEAU77] BEAUDRY, D., Performance related reliability measures of computing systems, IEEE Intern. Conf. Fault Tolerant Computing, 1977

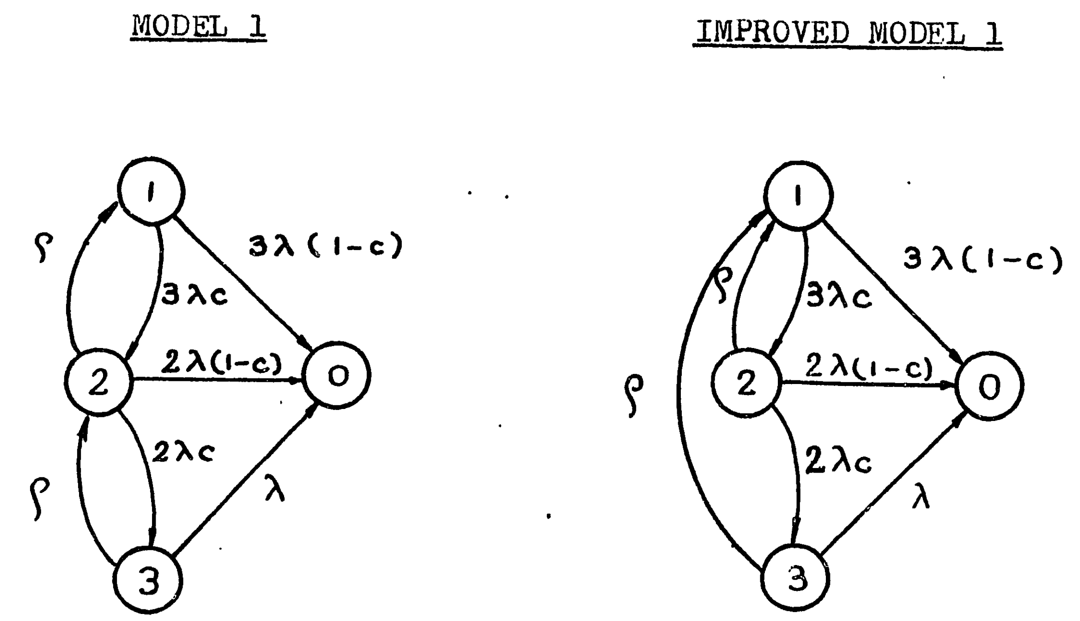
[BOUR71] BOURICIUS, W. et al, Reliability modeling for fault tolerant computers, IEEE Trans. on Computers, November 1971

[BUZA70] BUZACOTT, J., Markov approach to finding failure times of repairable systems, IEEE Trans. on Reliability, November 1970

[BUZA80] BUZACOTT, J., A recursive algorithm for finding reliability measures related to the connection of nodes in a graph, Networks, Vol 10, pp 311-327, 1980

[KLEI67] KLEINROCK, L., Queueing systems, Vol. I and II, Wiley, New York, 1976

[NG76] NG, Y., Reliability modeling and analysis for fault tolerant computers, UCLA-ENG-7698, September 1976



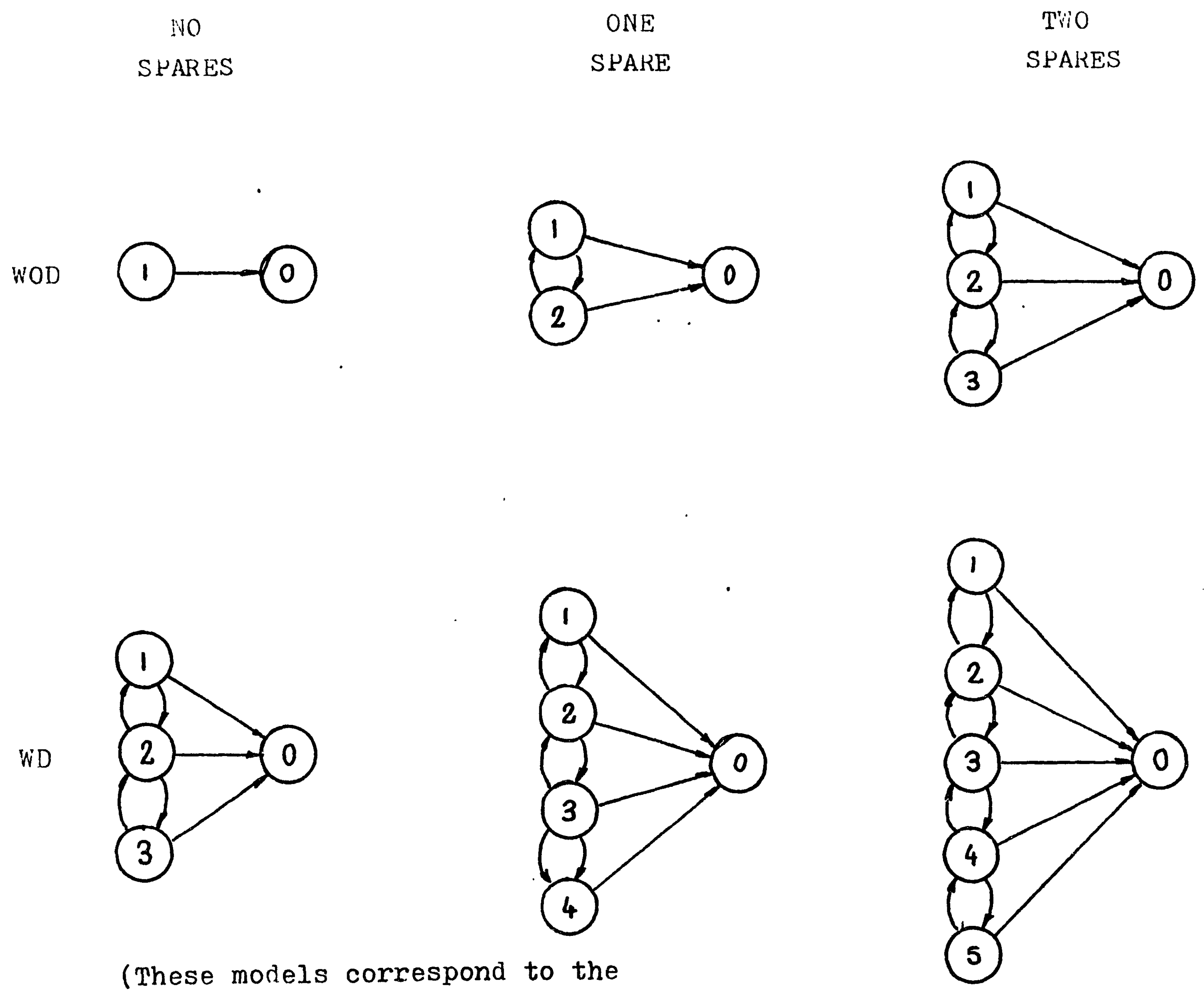
See Notation in FIG.E.1

FIG.6.1

LABELLING OF THE VARIABLES USED IN THE ANALYSIS OF THE CONFIGURATIONS
****

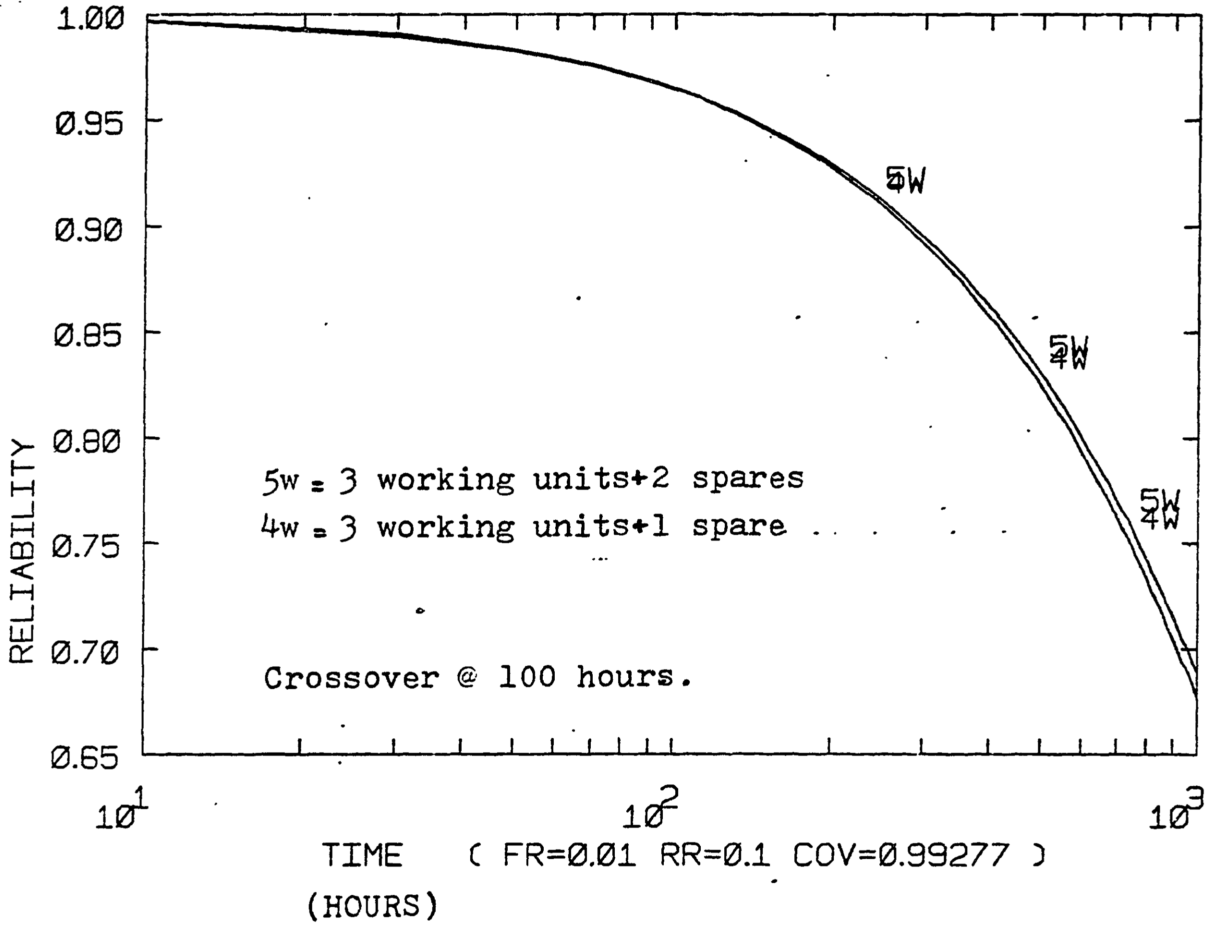
$$\begin{tikzcd}
 & \text{NO SPARES} & \text{ONE SPARE} & \text{TWO SPARES} \\
\text{WITHOUT DEGRADATION (WOD)} & \circled{15} & \circled{19} \arrow[l, "27"'] & \circled{20} \arrow[l, "28"'] \arrow[ll, bend right, "29"'] \\
\text{WITH DEGRADATION (WD)} & \circled{21} \arrow[u] & \circled{22} \arrow[l, "36"'] \arrow[u] & \circled{23} \arrow[l, "31"'] \arrow[u] \arrow[ll, bend left, "32"]
\end{tikzcd}$$

FIG.6.2



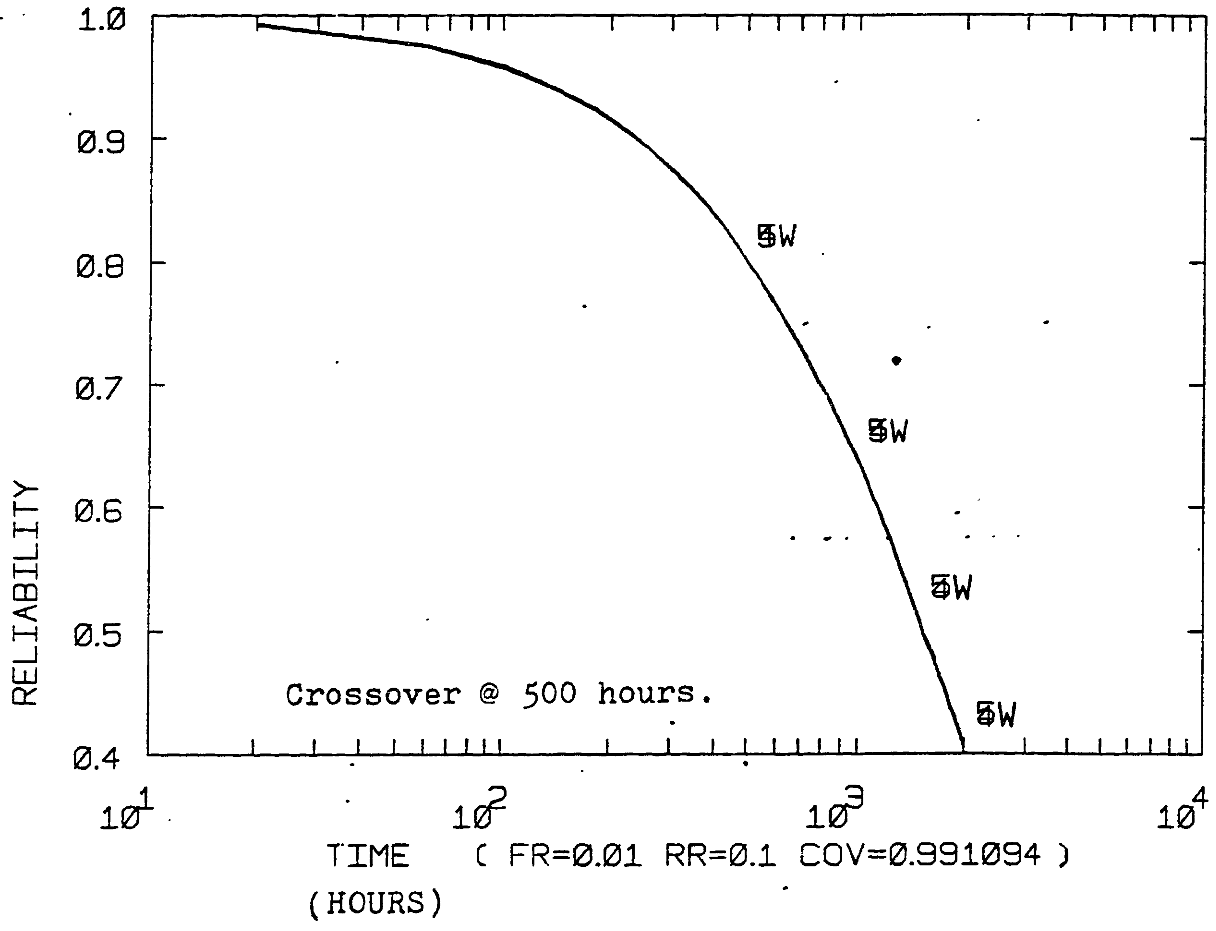
(These models correspond to the configurations shown in FIG.6.2)

FIG.6.3



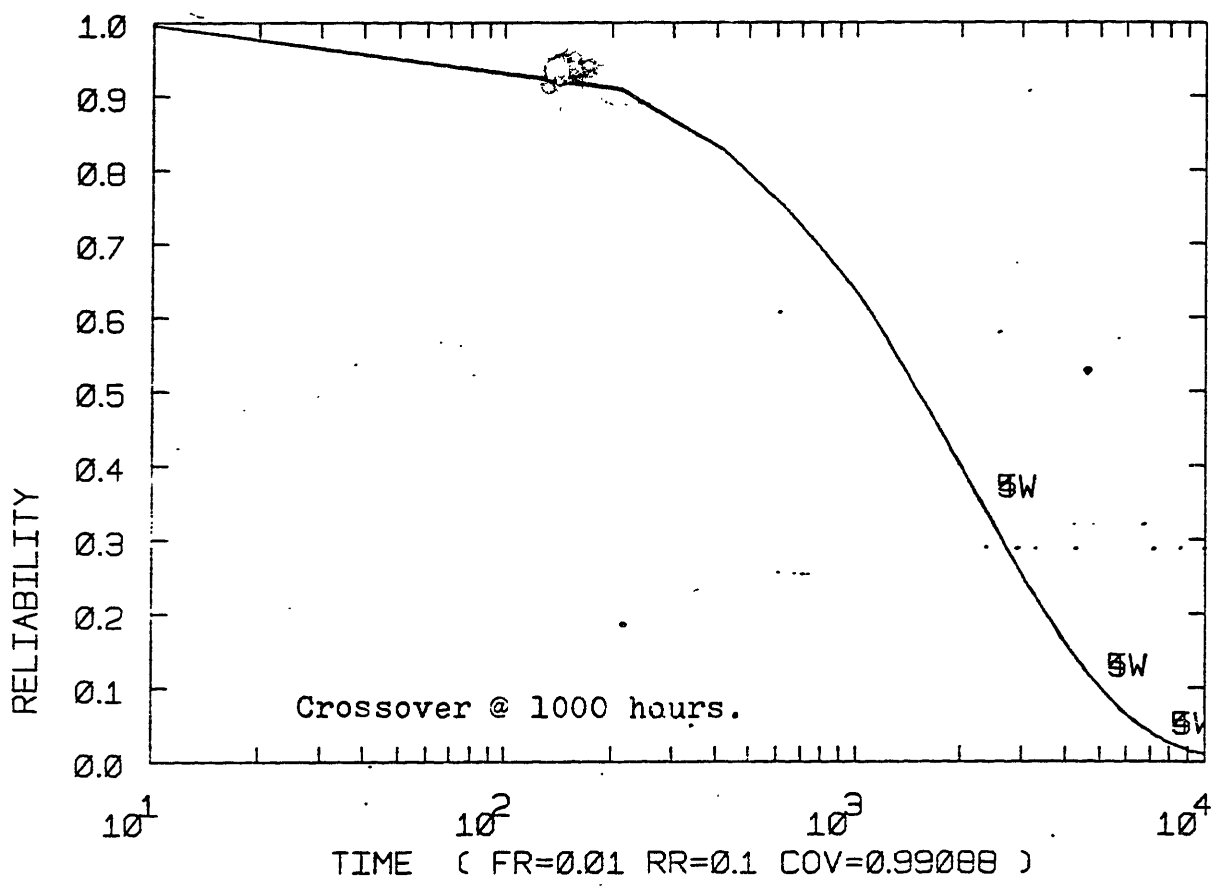
CROSSOVER AT 100 HOURS
****

FIG.6.4



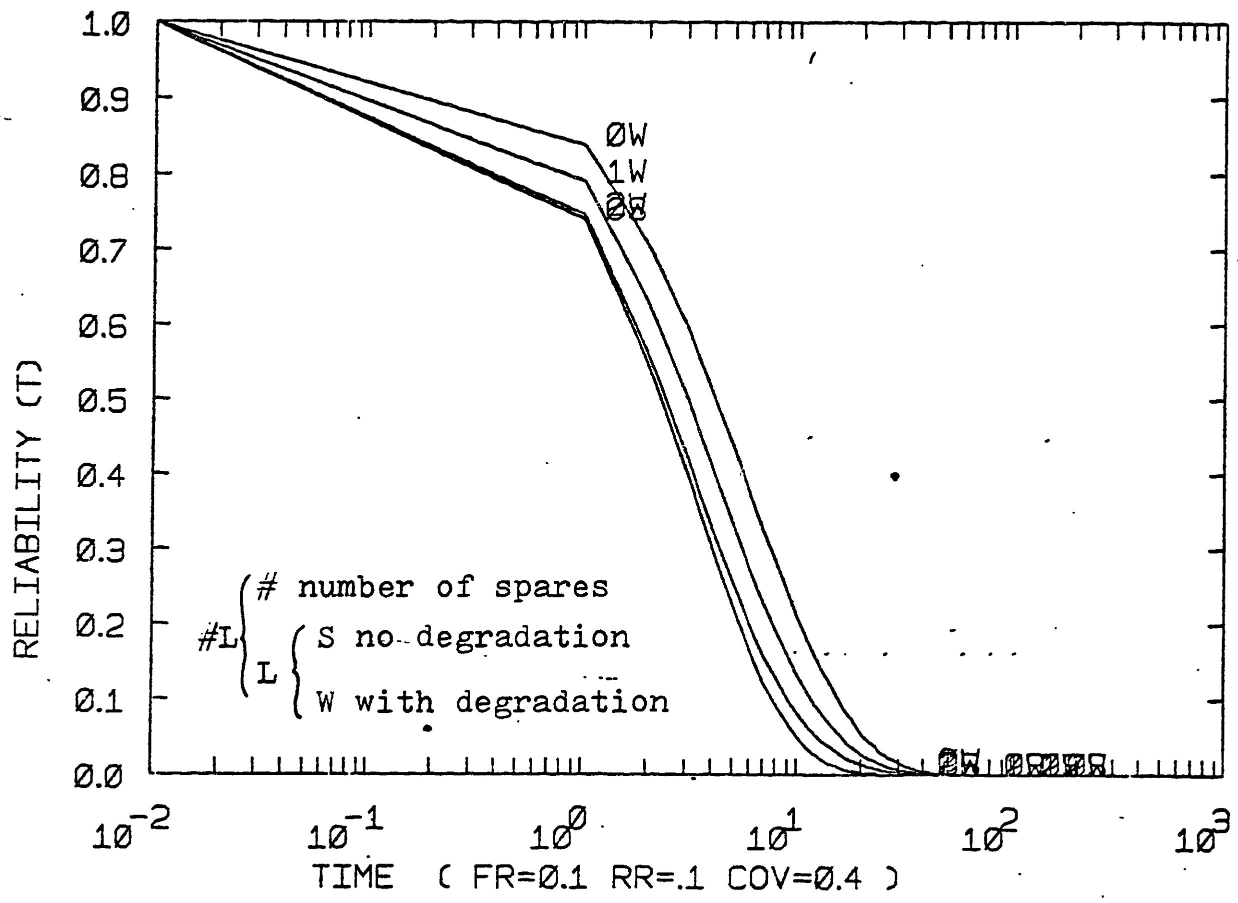
CROSSOVER AT 500 HOURS
****

FIG.6.5



CROSSOVER AT 1000 HOURS
****

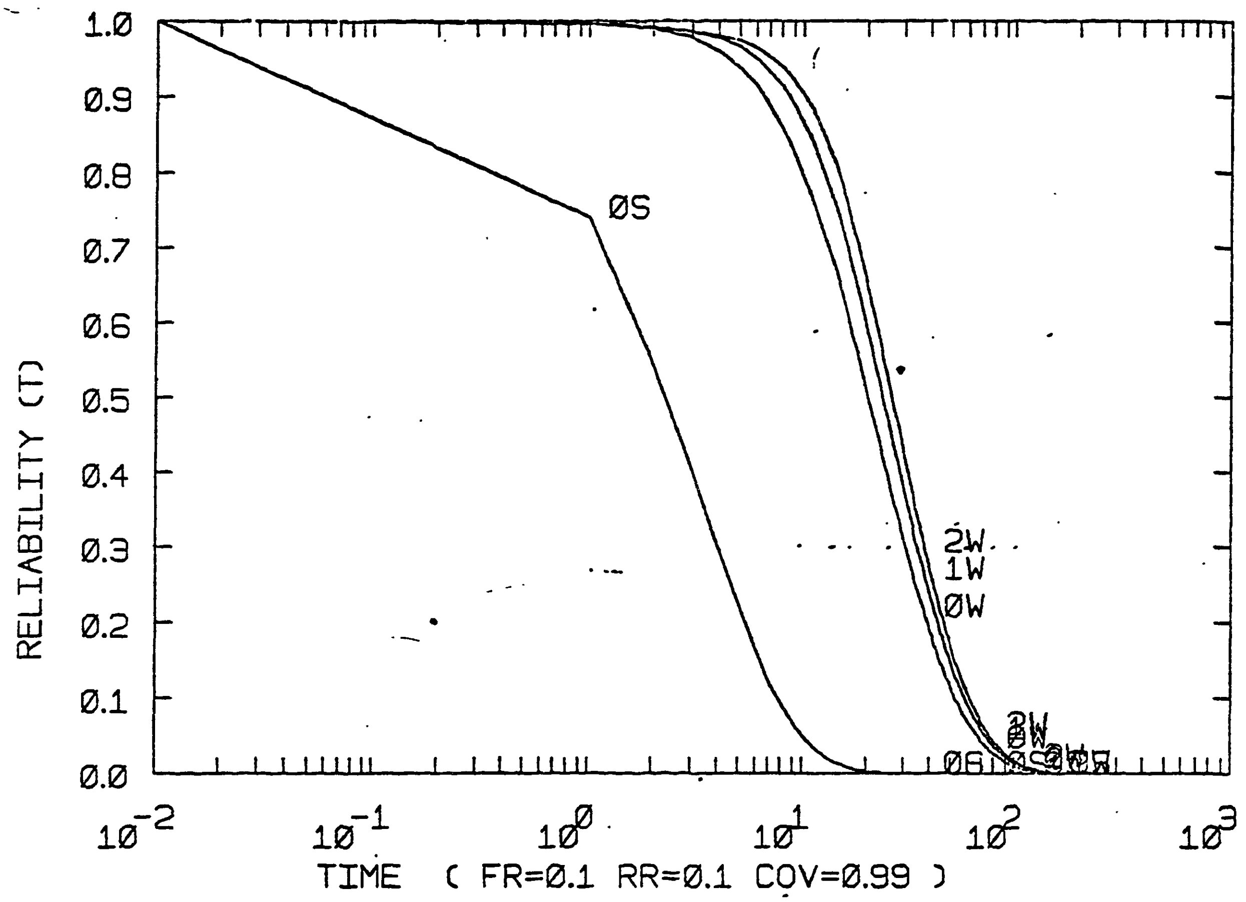
FIG.6.6



RELIABILITY AS A FUNCTION OF THE NUMBER OF SPARES
****

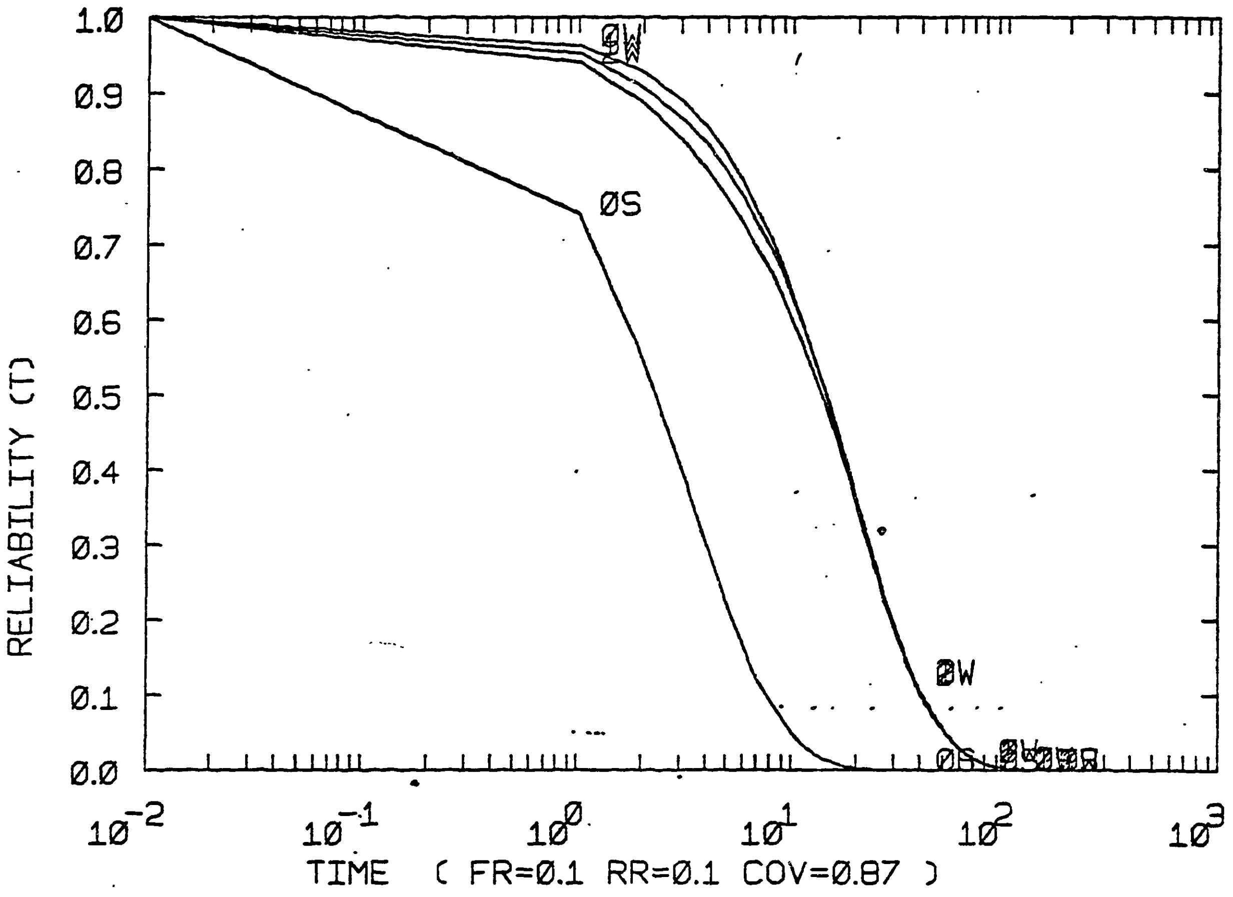
LOW COVERAGE
****

FIG.6.7



HIGH COVERAGE
****

FIG.6.8



INTERMEDIATE COVERAGE
****

FIG.6.9

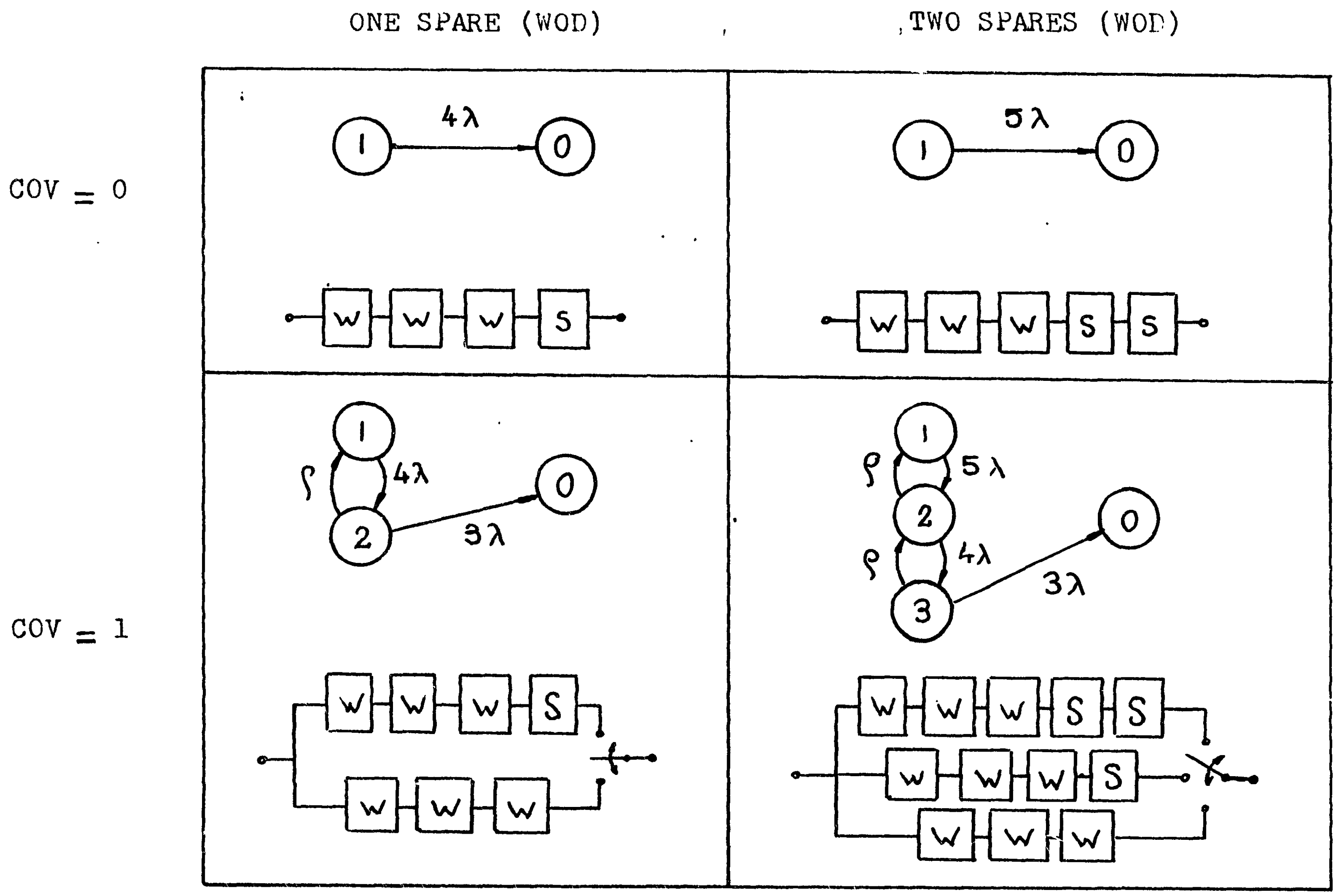


FIG.6.10

NUMBER OF SPARES AS A FUNCTION OF FAILURE RATE
****

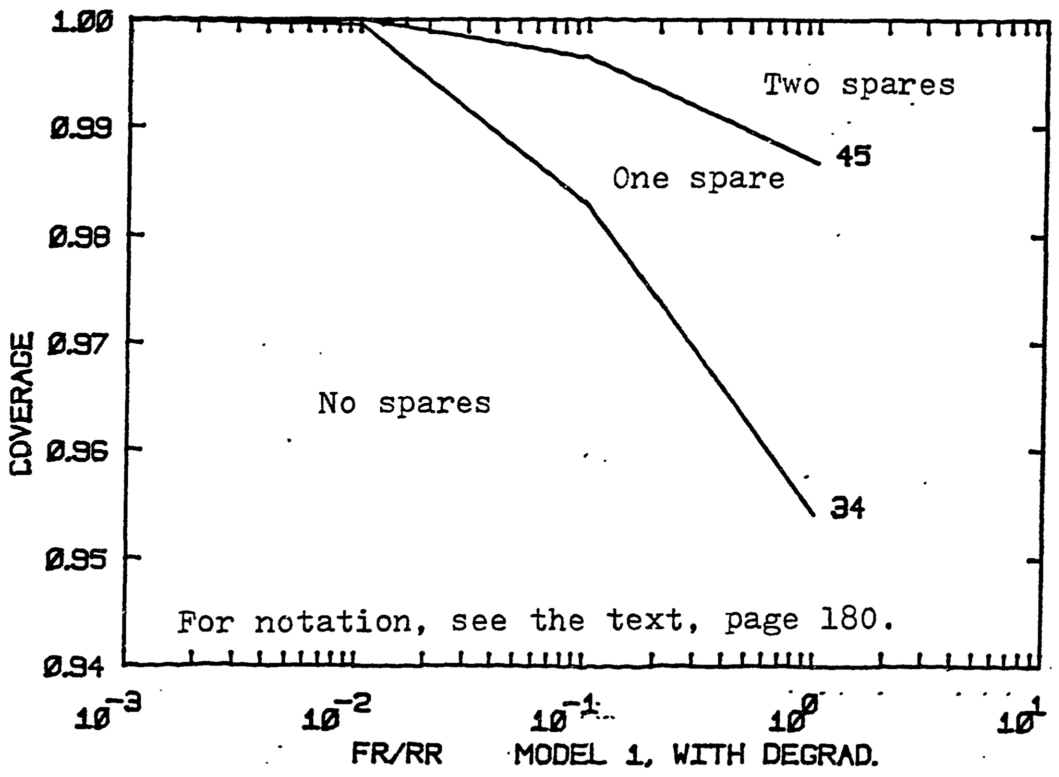
REPAIR RATE AND COVERAGE
****

failure rate / repair rate	3 and 4 units		4 and 5 units	
	coverage	reliabil.	coverage	reliabil.
1.0	0.954	--	0.98667	0.9796
0.1	0.9828	0.9783	0.99646	0.9942
0.01	0.99947	0.9993	0.9999836	0.99997

The reliability is calculated at 1/(3*failure rate) [hours]

These numerical values are plotted in FIG.6.12.

FIG.6.11



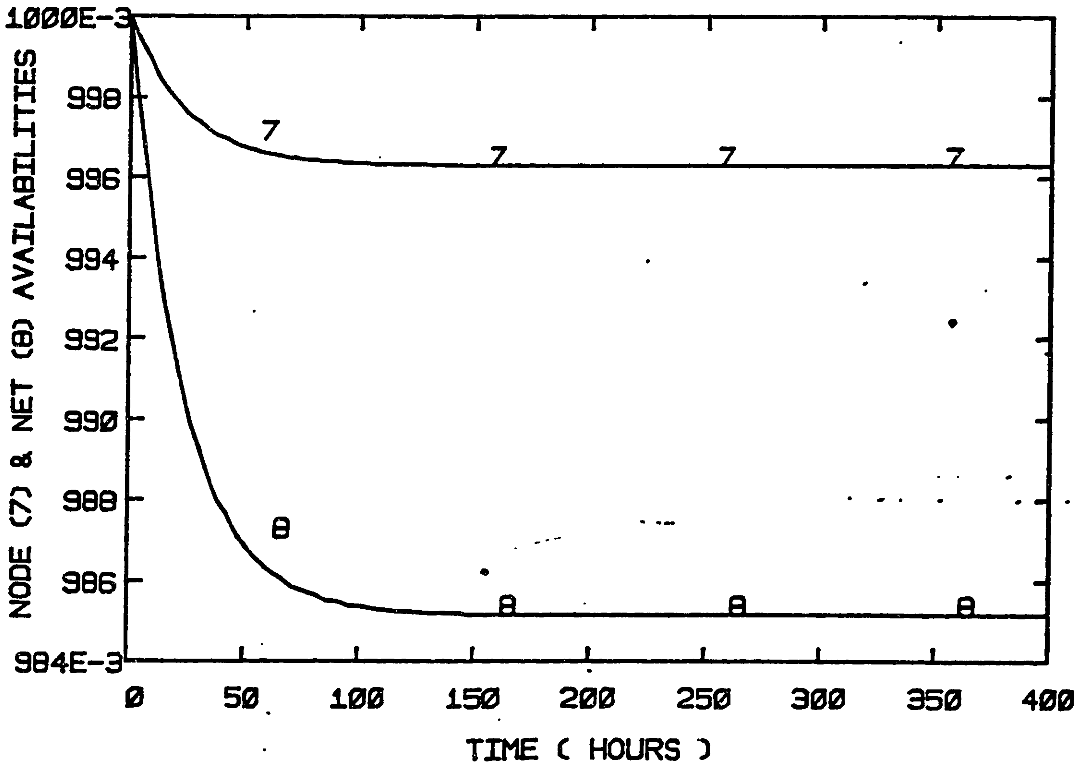
NUMBER OF SPARES AS A FUNCTION OF COVERAGE,
****

FAILURE RATE AND REPAIR RATE
****

FIG.6.12

```
          DYSYS -- DYNAMIC SIMULATION OF MIXED PARAMETER SYSTEM
          ****************************************************

 TIME      VARIABLE      VARIABLE    VARIABLE       NODE        NODE      NETWOR
[HOURS]     ( 1)          ( 2)        ( 3)       UNAVAILAB.  AV

0.0001     1.000       0.0000E+00  0.0000E+00  0.0000E+00   1.000       1.00
0.1001     0.9997      2.7115E-04  2.5331E-08  1.5020E-05
0.2001     0.9995      5.1637E-04  9.4850E-08  2.9802E-05   1.000       0.999
0.3001     0.9992      7.3814E-04  1.9989E-07  4.4644E-05
0.4001     0.9990      9.3871E-04  3.3305E-07  5.9426E-05  0.9999       0.999
0.5001     0.9988      1.1201E-03  4.8799E-07  7.4327E-05
0.6001     0.9986      1.2841E-03  6.5934E-07  8.9049E-05  0.9999       0.999
0.7001     0.9985      1.4325E-03  8.4257E-07  1.0377E-04
0.8001     0.9983      1.5667E-03  1.0338E-06  1.1837E-04  0.9999       0.999
0.9001     0.9982      1.6880E-03  1.2300E-06  1.3280E-04
 10.00     0.9959      2.8328E-03  5.3752E-06  1.2666E-03  0.9987       0.994
 20.00     0.9951      2.8305E-03  5.3729E-06  2.1059E-03
 30.00     0.9945      2.8288E-03  5.3697E-06  2.6571E-03  0.9973       0.989
 40.00     0.9941      2.8278E-03  5.3675E-06  3.0190E-03
 50.00     0.9939      2.8270E-03  5.3661E-06  3.2568E-03  0.9967       0.986
 60.00     0.9938      2.8266E-03  5.3652E-06  3.4130E-03
 70.00     0.9937      2.8263E-03  5.3646E-06  3.5157E-03  0.9965       0.985
 80.00     0.9936      2.8261E-03  5.3642E-06  3.5830E-03
 90.00     0.9935      2.8260E-03  5.3640E-06  3.6272E-03  0.9964       0.985
100.00     0.9934      2.8260E-03  5.3639E-06  3.6561E-03
```

FIG.6.13

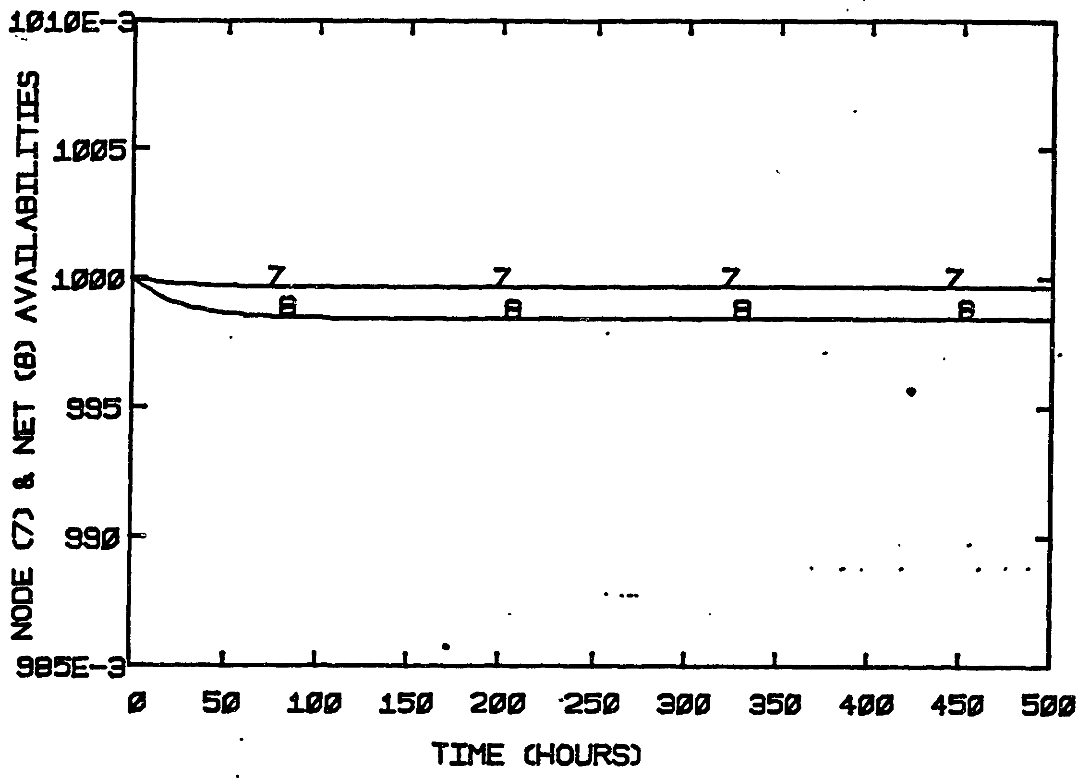


NODE AND NETWORK AVAILABILITIES AS A FUNCTION
****
OF TIME, FAILURE RATE AND COVERAGE
****

Failure rate: 0.001 failures/hour

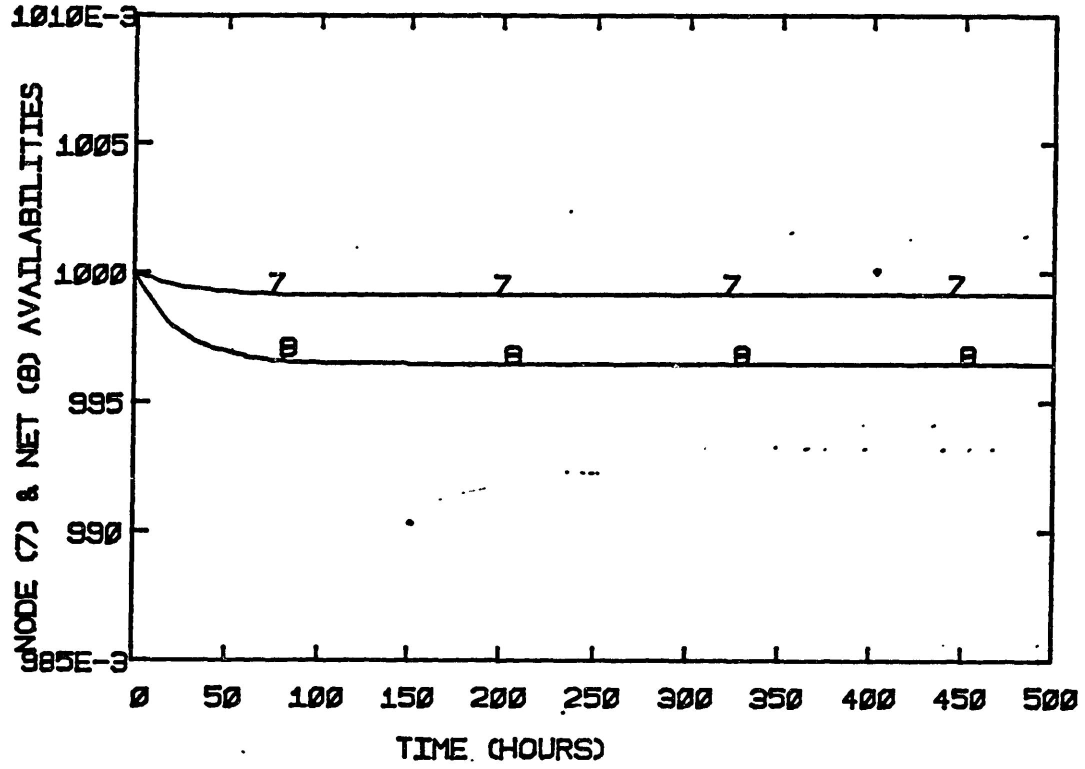
Coverage : 0.95

FIG.6.14



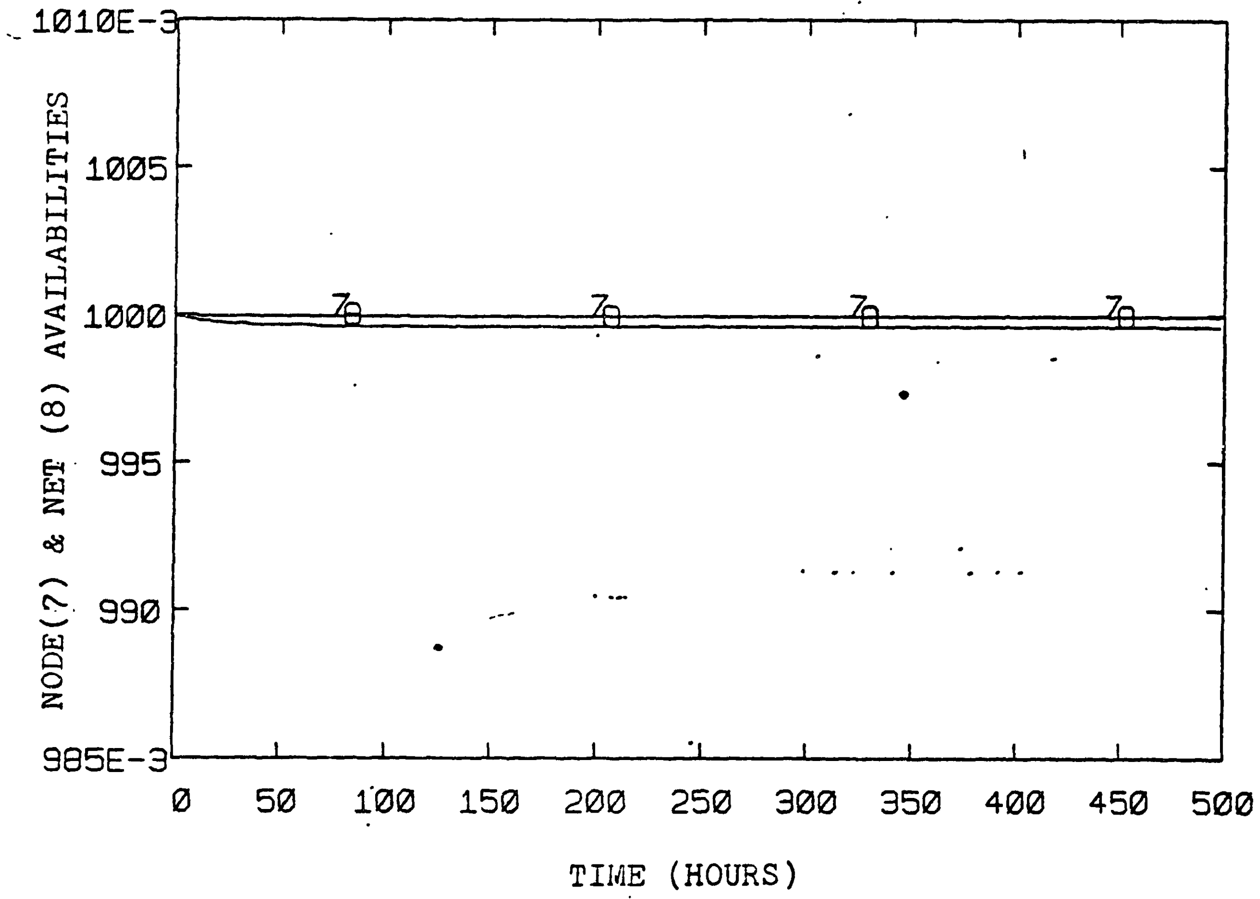
Failure rate: 0.0001 failures/hour
Coverage : 0.95

FIG.6.15



Failure rate : 0.001 failures/hour
Coverage : 0.99

FIG.6.16



Failure rate: 0.0001 failures/hour

Coverage : 0.99

FIG.6.17

NETWORK AVAILABILITY
****

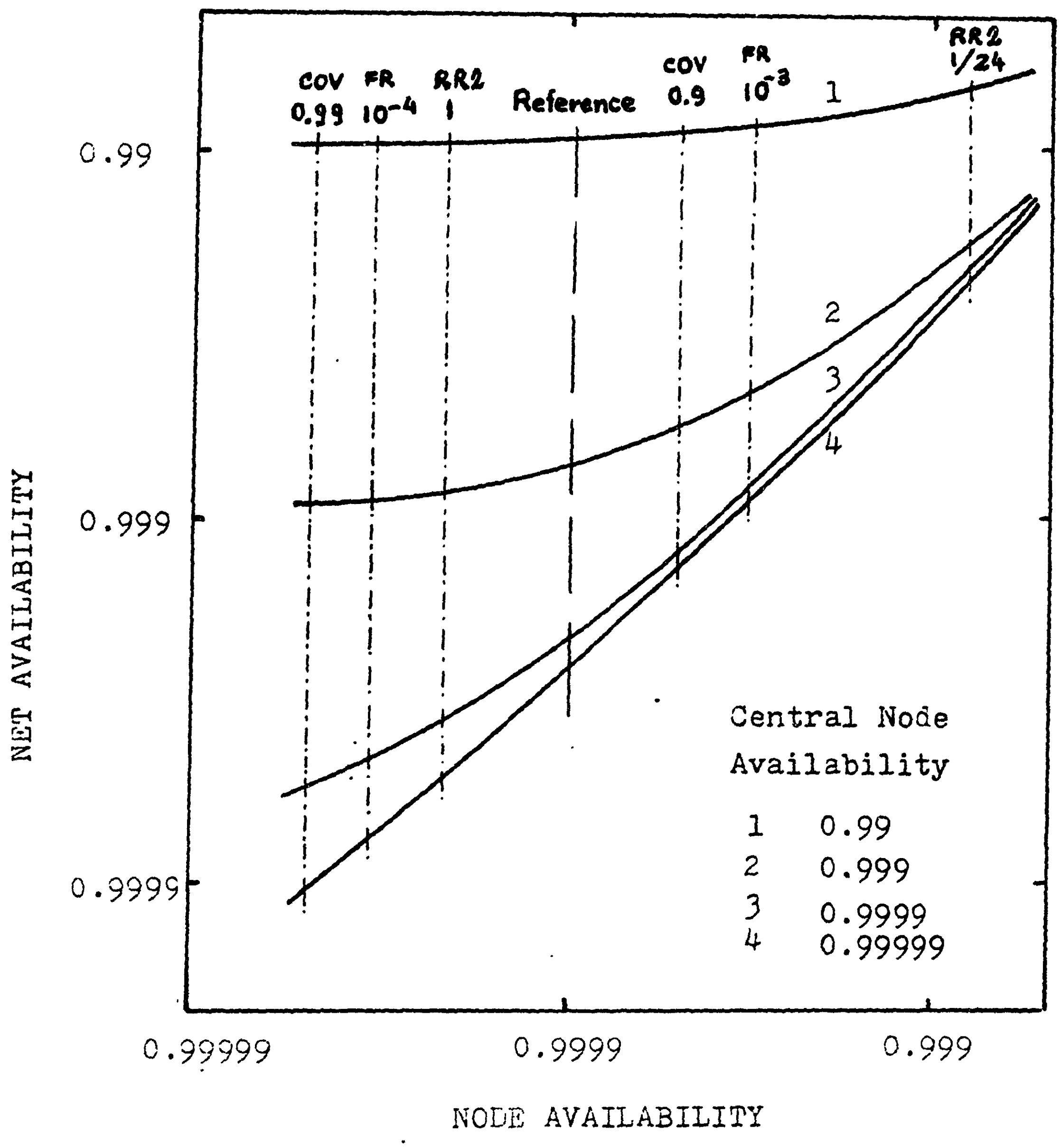
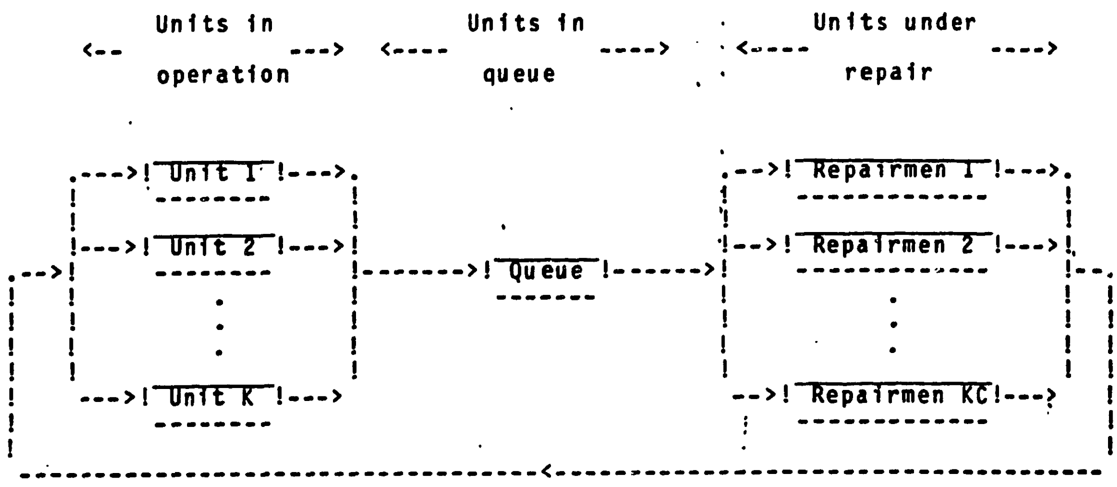


FIG.6.18

M/M/KC/K/K QUEUEING SYSTEM
****



ALFA	=	Failure rate of the units [failures/hour]
AMU	=	Repair rate of one repairman [repairs/hour]
K	=	Total number of units (units per node * # of nodes)
KC	=	Total number of repairmen (KC = 1, 2, ... ; KC < K)
ALQ	=	Average number of units to be repaired
WQ	=	Average waiting time in queue to repair [hours]
D	=	Probability that a unit has to wait to be repaired
EQ	=	Average waiting time for those units who must wait [h]
AL	=	Average number of units down (ALQ + units in repair)
W	=	Total average waiting time (WQ + time in repair) [h]
ALAMBDA	=	Rate of incoming units [units/hour]
RHO	=	Server utilization

Comment : these parameters are modeled in both the graphical method and in the program QWR, with the exception of RHO, which exists only in QWR.

FIG.6.19

EQUATIONS USED IN THE M/M/KC/K/K QUEUEING MODEL
****

E[x] = expected value of x

E[operating time] = 1/ALFA = average operating time per unit

E[repair time] = 1/AMU = average repair time per unit per repairman

pn = probability that there are n units in operation

$$pn/po = \sum_{n}^{K} (ALFA/AMU)^{**}n \qquad n = 1, 2, \ldots, KC$$

$$pn/po = \sum_{n}^{K} \frac{[(ALFA/AMU)^{**}n]^{*}n!}{KC!^{*}KC^{**}(n-KC)} \qquad n = KC + 1, \ldots, K$$

$$po = [\, 1 + \sum_{n=1}^{K} (pn/po)]^{-1}$$

$$ALQ = \sum_{n=KC+1}^{K} (n-KC)^{*}pn$$

$$WQ = \frac{ALQ * (1/ALFA + 1/AMU)}{(\, K - ALQ\,)}$$

$$D = \sum_{n=KC}^{K} pn$$

$$EQ = \frac{WQ}{D}$$

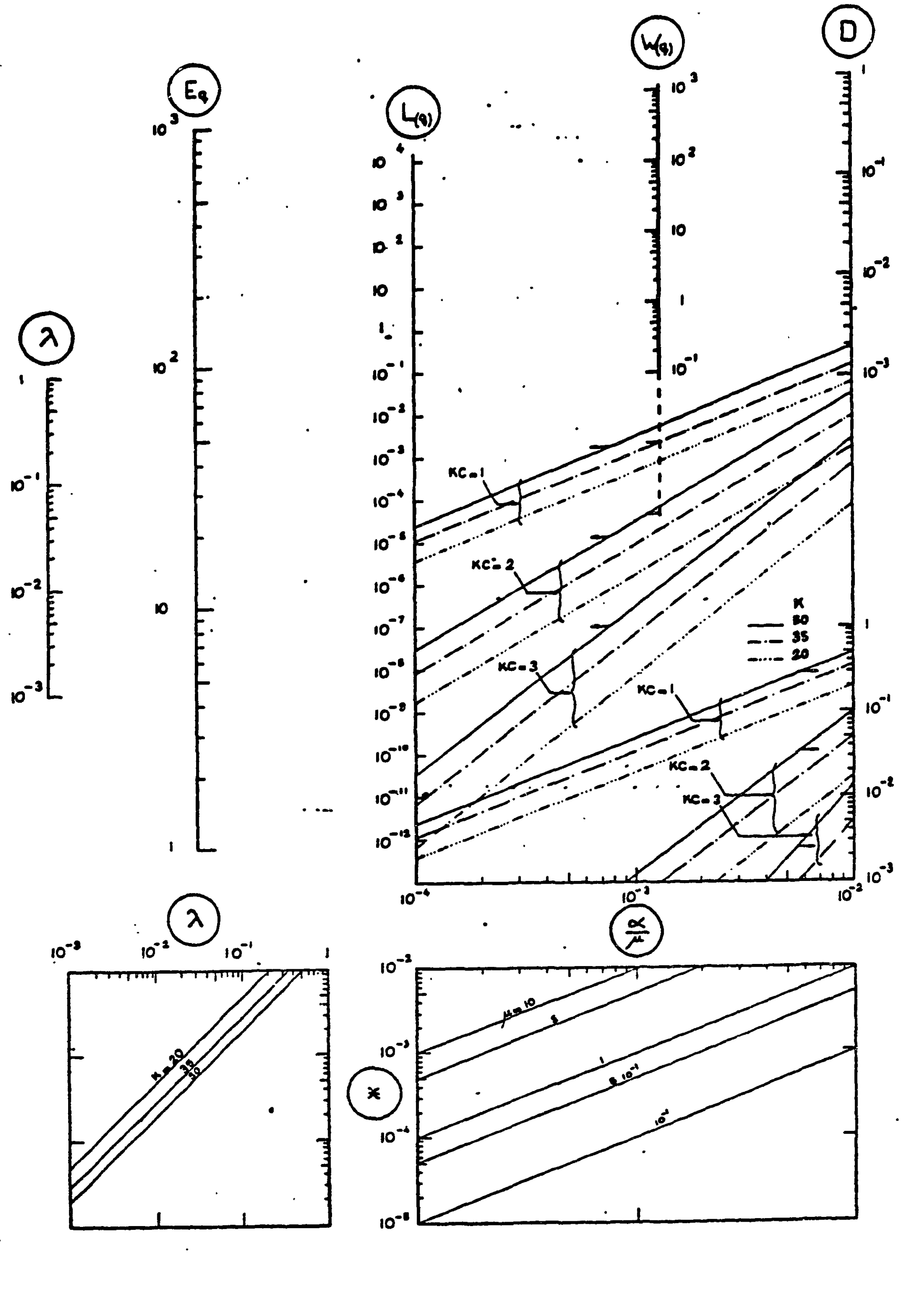
$$W = WQ + \frac{1}{ALFA}$$

$$ALAMBDA = \frac{K}{1/ALFA + 1/AMU + WQ}$$

$$RHO = \frac{ALAMBDA}{AMU * KC}$$

FIG.6.20

M/M/KC/K/K QUEUEING SYSTEM
****



SELECTION OF THE NUMBER OF REPAIRMEN

ALFA/AMU	KC	D	AMU*EQ	EQ		
				AMU=0.1	AMU=1.0	AMU=10.0
1.0E-1	1	0.998	9.05	90.5	9.05	0.905
	2	0.741	1.47	14.7	1.47	0.147
	3	0.332	0.57	5.7	0.57	0.057
1.0E-2	1	0.198	1.15	11.5	1.15	0.115
	2	1.71 E-02	0.49	4.94	0.49	0.049
	3	9.90 E-04	0.30	3.01	0.30	0.030
1.0E-3	1	0.020	0.97	9.67	0.97	0.097
	2	1.88 E-04	0.45	4.54	0.45	0.045
	3	1.12 E-06	0.29	2.85	0.29	0.029
1.0E-4	1	0.002	0.95	9.52	0.95	0.095
	2	1.90 E-06	0.45	4.50	0.45	0.045
	3	1.14 E-08	0.28	2.84	0.28	0.028
1.0E-5	1	2.0 E-04	0.95	9.50	0.95	0.095
	2	1.90 E-08	0.45	4.50	0.45	0.045
	3	1.14 E-12	0.28	2.83	0.28	0.028

Note: K = 20

Nomenclature:	ALFA	= Failure rate of the units [failures/hour]
	AMU	= Repair rate of the units [repairs/hour]
	D	= Prob. that a unit has to wait be repaired
	EQ	= Av. time for those units that must wait [h]

FIG.6.22

CHAPTER 7

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

In this Thesis, issues in the design of a monitoring system for a nuclear power plant are discussed. The first two Chapters address _general_ concepts about nuclear power plant control systems and the characteristics of a system suggested to satisfy the requirements proposed. From Chapters 3 to 6, on the other hand, _detailed_ issues are considered.

In this Chapter, a summary of the characteristics of the proposed control system is presented and recommendations applicable to the design of such a system are given.

7.1 SUMMARY OF THE THESIS

In Chapter 1 _the need for control systems_ is presented. In these control systems a wide variety of techniques could be _integrated_ to aid the operator in his understanding of the status ('state') of the plant. Thus, it is expected that the operation of the plant would be improved. Such a system is called an _Integrated Control System_ (ICS) to stress that integration should exist in the following areas:

- objective definition: the Safety aspect and the Productivity aspect in the operation of the plant,
- cooperation among the user, the supervisor and the designer,
- availability of information at various points in the system, as required.
- design of the computer system.

In Chapter 2, a _methodology for the design of the system_ is introduced (FIG.2.1). The first step of the methodology is the identification of the goal to be accomplished by the system; ie _to assure the continuous safe operation of a nuclear power plant_. Again, a variety of techniques can be invoked. For example: the upgrading of the training of operators by the use of simulators or performing modifications in the design of the plant or installing a monitoring system. It has to be clear that the new system, the ICS, may only be one of the various means of upgrading the operation of the plant and another techniques should be considered as well.

After deciding that a control system is required for the operation of the plant, the functions and characteristics of the system are sought. The analysis of the requirements for the system (FIG.2.2) provides the first ideas on which to base the design. After tradeoffs among the requirements were made, a preliminary design is conceived (FIG.2.3).

Two scenarios are envisioned for the system. During the first scenario, called 'INITIAL', only monitoring of the variables is performed by the system (thus, there are no connection to the actuators). During the second scenario, named 'GROWTH', it is supposed that control on some variables will be allowed, and, consequently, digital to analog converters and actuators will be needed (the system will require hardware modifications). Control algorithms will also have to be developed (software mdifications will be done).

The ICS is a fault tolerant (FT) distributed system consisting of a) a FT central computer (located adjacent to the control room), b) FT distributed computers (situated in the buildings of the power plant), c) a FT network that connects the central computer and distributed computers and d) a number of microcomputers and multiplexers that provide the interface between the sensors and actuators in the plant and the distributed computers. The sensors and actuators are the only interface between the 'processes' of the plant that are monitored (ie core, steam generator, pumps, turbine) and the ICS. The gross distribution of the functions of the system can be seen in FIG.2.3.

Distributed processing is considered as the most viable alternative because it is powerful enough to satisfy the computational requirements for the system. The modularization facilitates the system's design, testing, maintenance and expansion to add more functions. On the other hand, centralized processing is inherently more complicated to design, test, maintain and modify; equivalent processing power to match the capabilities of distributed processing would imply costly solutions of possibly lower reliability.

Several fault tolerance techniques are chosen to meet the reliability requirement that certain single failures of the hardware must not disable the system. These techniques are:

- use of high quality components in units whose malfunctions are strongly dependent of the failure rate of the individual components,
- error detection and correction,
- replication of unreliable and/or critical components or units,
- on-line diagnostic programs,
- self tests,
- <u>triplication</u> (at least) of processing units,
- voting by software the outputs of the processing units (bit by bit or every few bits),

- detection of some malfunctions,
- dynamic reconfiguration of the units and
- use of 'hot standby' spares.

With reference to FIG.2.3 it can be seen that there are three processing levels. The first level is represented by the microcomputers, the second level by the FT distributed processors and the third level by the FT central computer.

The <u>microcomputers</u> are in charge of acquiring signals from the sensors and performing an initial but limited processing.

This includes smoothing signals, obtaining the mean and standard deviation of the measurements, checking the signals against thresholds or upper or lower limits, calibrating sensors and amplifiers, etc.

The <u>FT</u> <u>distributed</u> <u>processors</u> perform more sophisticated processing of the signals and, in addition, in most processors some analytic models can be running in real time for analytic redundancy comparisons. These distributed processors basically receive data from the microcomputers and a) check and combine the data to produce synthesized information to be used by the central computer or b) when automatic control is permitted, the processors perform the computations corresponding to the control algorithms and send the commands to the microcomputers to move the actuators accordingly.

For the purpose of this Thesis a certain number of specific functions have been assumed to have been defined. At the time of an actual design this list must be correctly specified. It is envisioned that the minimum list of functions should include:

- validation of the signals sent by the microcomputers. This validation should consist of comparing the signals with previous values of the signal sent by the same sensor (hypothesis testing) or comparisons with another redundant sensors (note that this validation could be performed by the microcomputers themselves).
- comparison of the signals from the microcomputers with data derived from analytical calculations (analytic redundancy). The analytic calculations may come from models of the processes running in the distributed processors.
- control of some processes for the case of the 'GROWTH' scenario.

However, a more complete list will include the fact that the model may have to be 'adapted' to some of the conditions that will be encountered later in the life of the plant or for changes in operating regimes, and adaptive control could be used for the purpose of performing control even when the plant does not correspond exactly to the original model. Also, the new parameters of the plant should be incorporated into the new models. To estimate the variables subject to a series of constraints, <u>parameter estimation</u> may be used. Sometimes (and at least one commercial design considers its use) special <u>digital signal processing</u> techniques can be used for the purpose of doing <u>noise analysis</u> for processes in which their behavior can be determined by changes in the frequency spectrum (ie vibration analysis of pumps, motors and turbines) as well as <u>sensor noise analysis</u> on pressures, flows and ion chamber currents.

The <u>FT central computer</u> receives information from the distributed processors either to be stored in memory or to be communicated to other computers or used to generate displays for the operator's use.

The operators may interact with terminals to ask for certain variables, operational data of the plant, setpoints, present or past values, etc.

Fiber optics cables are selected to link the components of the ICS due to their high bandwidth, electric safety advantages and electromagnetic interference (EMI) immunity. The use of a single cable to send many signals makes fiber optics cost effective compared to classical hardwiring. The links between the microcomputers and multiplexers are triplicated to comply with the requirement of allowing normal processing even in the case of single failures. The links among the distributed processors, on the other hand, do not need to be replicated because the links and the processors are interconnected in a fault tolerant (FT) network. The FT network allows communications between FT the nodes and the FT central computer even in the case of malfunctions in the links or in the nodes.

From Chapter 3 to Chapter 6, specific details of the ICS are treated.

In Chapter 3, the <u>layout</u> of the ICS in a typical PWR plant is considered. The objective is to distribute and interconnect the elements of the ICS so that the total installation cost of the network is minimized. One method for the minimization of the installation cost has been formulated by considering it a problem with an integer programming solution. The structure of the network is shown in FIG.3.3. The final result of the integer programming solution is portrayed in FIG.3.6.

Four solutions are given to the layout problem, which come about as a combination of two situations that involve the use of 4 or 5 links from the central computer to the distributed computers and other two situations that correspond to the 'INITIAL' or 'GROWTH' scenarios.

In subsequent Chapters (4 and 5), a technique is used to <u>estimate the real time requirements of two algorithms</u>: the Kalman Filter (KF) in Chapter 4 and the Sequential t-Test (StT) in Chapter 5. In the technique, the number of operations that are required per iteration of an algorithm are used to calculate the time and memory requirements for the implementation of the algorithm in a specified processor, considering all the time consuming operations that take place.

The Kalman Filter can be used to estimate variables of a process represented by a linear model. The estimated variables are optimal in the sense that the the expected value of the square of the error between the true variable and the estimate is minimal. In this case it is found the maximum number of variables and measurements that a model can have are determined such that one cycle of the algorithm within the processor would take a certain amount of time. It has been calculated that a) the time spent in the calculations is a function of the third power of the number of variables in the model, b) 14 to 15 variables imply an update time of (roughly) 1 second in a MOTOROLA MC6800-based system and c) storage of the programs and data do not become a problem given the present capacity of memories.

The Sequential t-Test (StT) can be used for signal validation. The test has been developed to 'decide' if a certain variable being measured (for example the mean value of a temperature) is between ranges specified to give not more than predetermined errors. The StT is used because it was originally designed to provide the 'decision' with a minimum average number of measurements from the variable in question. An algorithm to use the StT was developed and it is shown in Appendix B. With the algorithm for the StT and the procedure of Chapter 4 used for the KF, the time and storage required (also in the MC68000-based system) were estimated to be 1 millisecond per measurement and 800 bytes. Based on these numbers, the maximum number of sensors that can be sampled by the microprocessor are calculated.

It might be assumed that the StT, used for signal validation, would be one of the main functions performed by the microcomputers. The 'validated signals' from the microcomputers could be sent to the FT distributed processors a) to be compared with the 'estimated variables' derived from analytic redundancy calculations and b) to perform the parameter estimation and update the model used for analytic redundancy. However, there is some uncertainty at the present stage of the overall research in this area concerning how to use 'validated signals' in order to change a model that is used to check the same signals against analytic calculations in the case of failed sensors (or sensors with uncharacterized noises).

In Chapter 6, the survivability of the ICS is considered. Survivability is used to encompass the more specific concepts of reliability (no 'major' repair) and availability ('major' repairs allowed). Three models are used: MODEL I models the reliability and availability of the nodes, MODEL II considers the reliability and availability of the ICS in general and MODEL III assumes interaction in the repair of units in the ICS (units may refer to sum of processors of the FT distributed processors or these processors plus the microcomputers in the source layer if the failure and repair characteristics are the same).

Both concepts (reliability and availability) include the so called 'minor' repair in which operators may replace faulty units in a short time (average time 1 hour). The node is assumed to work as long as there are more than a specified minimum number of processors (it is recommended that the minimum number of processors should be one to extend the life of the system). Below the minimum, the node is declared 'failed'. The repair action that takes the node from the failed state is called 'major' repair. For reliability it is assumed that the node does not have 'major' repairs (consequently, the steady state reliability goes to zero). For availability, it is assumed that 'major' repairs are allowed (thus, the steady state availability is, for the cases treated, different than zero). For design considerations it is recommended that the concept of availability should be used for those systems that allow repairs from the failed states and that the steady state availability should be higher than the minimum availability required for the system.

In calculations involving MODEL I and in MODEL II, reliability is used to study the characteristics of the node simply because the analysis of the reliability takes less computational time than the availability. Results obtained from reliability calculations have also been converted to availability results.

Using MODEL I it was shown that

- the number of spares to maximize the reliability of the nodes depends on the ratio of the failure rate to the repair rate and the coverage of the units.
- for most of the situations considered in this work and unless extremely high coverages are used, the number of units recommended to work per node is three.

The reliability measures for the network (MODEL II) incorporate the concept that the probability that a node may be disconnected from the network has to be minimized. With this reliability measure, it is found that the probability that a node is disconnected decreases if the number of nodes decreases. Other findings from the use of MODEL II are that:

- the network availability can be no better than the availability of the least available node (distributed or central computer).
- changes in some of the parameters may produce significant variations in the availability of the network only when the availability of the central computer is very high. Thus, if the central computer is not very reliable, it is not necessary to improve the availability of the individual nodes, because the network availability will not change appreciably.
- if the individual nodes are unreliable, there is no point in improving the central computer. Thus, a highly reliable central computer must be matched with equally reliable distributed nodes.

MODEL III is a queueing model of the system, from which it was shown that a number of parameters related to the repairability of the system could be succesfully modeled, again, as a function of the ratio of the failure rate of the individual units to the repair rate of one repairman.

It was also shown that one operator in charge of the 'minor' repairs would be enough for the ranges of the parameters considered (failure rate lower than 0.001 [failures/hour] and repair rate higher than 1 [repair/hour]).

7.2 RECOMMENDATIONS FOR FUTURE WORK

One of the first requirements for designing a control system is to determine the processes that will be controlled, the parameters that will be monitored and the purpose of controlling these processes. For example, a control system could be designed to control the amount of xenon in the core, such calculations are not required in real time because the significant changes can be observed in periods of hours. On the contrary, if the parameter to be controlled is the power in the core, the time constants are of the order of milliseconds and the calculations must be done in real time. In the first example (control of xenon), detailed models of the core could be used; in the second example (power control), only lumped parameter models can be used nowadays to keep costs low.

Thus, as a first step in the actual design of the ICS, <u>the processes that will be monitored and the purpose of monitoring must be determined</u>, so that the <u>functions</u> to be performed by the processors on the signals received from the sensors can be obtained.

The functions (or algorithms, methods, etc) will be related to the characteristics of the processes and their modeling:

- noise present in the var' bles,
- ability to represent the system in terms of linear differential equations about an operating point,
- possibility of errors in the modeling because the characteristics of the system change with time (normal or abrupt changes).

Also characteristics about the 'operator side' of the system must be specified:

- how functions will be distributed between the operator and the machine,
- what kind of data the operator will need,
- how frequently the operator will require data,
- how should the data and alarms be presented (CRTs, boards).

Given the preliminary number of variables necessary to model each of the processes of the plant and once the <u>processing functions</u> and the <u>operator functions</u> are determined, the computational requirements are obtained.

Given the computational requirements, a number of analyses can be made to guide the preliminary design (in the same way that analyses have been made in this Thesis) for the layout problem, the determination of real time constraints for the Kalman Filter and the StT algorithms and the survivability of the network. For example,

- simulations could be used to specify the scheduling of real-time operations per processor,
- queueing models could be used to quantify the contention of resources for individual triads and for the network and to define buffer sizes,
- automatic reconfiguration of the network could be done by using optimal control theory applied to the communication of data among nodes.

After the preliminary design is made, <u>the architecture presented has to be validated</u>. A number of analytical models exist for this purpose.

In order to do a meaningful validation,

- realistic models of the system have to be developed,
- the assumptions used in the development of the models have to be indicated and, if possible, justified in practice (ie the use of exponential distributions to represent the service and repair times in the queueing model),
- the most important characteristics of the system have to be subject to sensitivity analysis so that the tradeoffs are readily observable and the necessity of future feedbacks is minimized.

Once the validation is completed, the design is modified and the validation is redone. This process is iterated until a validated model of the computer system is obtained.

The combined design presented in this Thesis can be used as a guide for future consideration on hardware requirements when additional information has been developed for improving the distributed system design.

By using for the overall design the structured methodology presented in Chapter 2 and analytical models to represent the behavior of the system, it is expected that the amount of feedback between the late phases of the life cycle of the system (development and operation) and the design phase would be minimized.

APPENDIX A

THE SOLUTION TO THE LAYOUT PROBLEM

This Appendix shows the data that was used to generate the results presented in Chapter 3, where the layout problem was discussed. The variables and costs applied to the objective function (3.1) are shown in FIG.A.1, while FIG.A.2 displays the constraints used for the same problem.

The actual input data used for the SESAME code is presented in FIG.A.3 (reference [SESA80]), and the results obtained are shown in the following figures, with the particular condition under consideration between parenthesis: FIG.A.4 (4 links and 'INITIAL' scenario), FIG.A.5 (4 links and 'GROWTH' scenario), FIG.A.6 (5 links and 'INITIAL' scenario) and FIG.A.7 (5 links and 'GROWTH' scenario).

The output of interest to these cases is condensed in three parts: a summary of the results and two SECTIONS called ROWS and COLUMNS (see FIG.A.4).

The summary indicates the total cost (104779.00), the objective function (CASE1) and the bounds to the constraints actually used (FIRSTBND). The objective function and the constraints were the same for the four cases.

The ROWS SECTION indicates how much was used of a certain constraint. For example, the concentrator number 1 has 927 signals connected to its I/O ports (actually through fiber optics) and 3169 signals are available to be connected.

The COLUMNS SECTION presents the values that the Linear Programming Solution provides. For example, w11 = w21 = w31 = w42 = ... = 1.0, indicating that the links indicated with the symbols wij and equal to 1.0 would connect the multiplexer located in building i to the concentrator number

j. Refer to Chapter 3 for the notation used in this Appendix.

VARIABLES AND COSTS
****

1	w11	9704	36	w81	15075
2	w12	12146	37	w82	11169
3	w13	13122	38	w83	6775
4	w14	15075	39	w84	6775
5	w15	12146	40	w85	5310
6	w21	5310	41	w91	21991
7	w22	9704	42	w92	12146
8	w23	14100	43	w93	9216
9	w24	16052	44	w94	7263
10	w25	11169	45	w95	7263
11	w31	19958	46	x1o	25305
12	w32	20934	47	x2o	30579
13	w33	25817	48	x3o	84909
14	w34	28746	49	x4o	24720
15	w35	26793	50	x5o	27648
16	w41	9216	51	x6o	28821
17	w42	5798	52	x7o	34092
18	w43	9216	53	x8o	27648
19	w44	12146	54	x9o	29844
20	w45	10193	55	y12	3235
21	w51	14100	56	y13	5025
22	w52	7751	57	y14	6002
23	w53	7751	58	y15	5025
24	w54	11169	59	y23	2747
25	w55	10193	60	y24	3398
26	w61	17028	61	y25	3072
27	w62	9216	62	y34	2421
28	w63	6775	63	y35	2421
29	w64	8240	64	y45	2421
30	w65	9216	65	z1o	3398
31	w71	19958	66	z2o	2909
32	w72	13122	67	z3o	3137
33	w73	9216	68	z4o	3235
34	w74	8240	69	z5o	3072
35	w75	10193			

FIG.A.1

CONSTRAINTS

Group A

$$w11 + w12 + \dots + w15 + x1^{\vee} = 1$$

$$w21 + w22 + \dots + w25 + x2^{\vee} = 1$$

$$\vdots$$

$$w91 + w92 + \dots + w95 + x5^{\vee} = 1$$

Group B

$$w11 + w21 + \dots + w91 - z1^{\vee} > Ø$$

$$w12 + w22 + \dots + w92 - z2^{\vee} > Ø$$

$$\vdots$$

$$w15 + w25 + \dots + w95 - z5^{\vee} > Ø$$

Group C

$$y12 + y13 + y14 + y15 + y23 + y24 + y25 + y34 + y35 + y45 = TESTC$$

Group D

$$z1^{\vee} + z2^{\vee} + z3^{\vee} + z4^{\vee} + z5^{\vee} = TESTD$$

Group E

(Initially)

$$178\ w11 + 543\ w21 + \dots + 78\ w91 < 4Ø96$$

$$178\ w12 + 543\ w22 + \dots + 78\ w92 < 4Ø96$$

$$\vdots$$

$$178\ w15 + 543\ w25 + \dots + 78\ w95 < 4Ø96$$

(Growth)

$$212\ w11 + 1161\ w21 + \dots + 138\ w91 < 5276$$

$$212\ w12 + 1161\ w22 + \dots + 138\ w92 < 5276$$

$$\vdots$$

$$212\ w15 + 1161\ w25 + \dots + 138\ w95 < 5276$$

Group F

(Initially)

$$178\ x1^{\vee} + 543\ x2^{\vee} + \dots + 78\ x9^{\vee} < 128$$

(Growth)

$$212\ x1^{\vee} + 1161\ x2^{\vee} + \dots + 138\ x9^{\vee} < 256$$

FIG.A.2

INPUT DATA TO SESAME
****

```
 NETTY    DATA     A1   F 80  TRUNC=80 SIZE=100 LINE=0 COLUMN=1
===== * * * TOP OF FILE * * *
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
===== NAME           NETTY
===== ROWS
=====   A1 = EQ, A2 = EQ, A3 = EQ, A4 = EQ, A5 = EQ, A6 = EQ,
=====     A7 = EQ, A8 = EQ, A9 = EQ
=====   B1 = GE, B2 = GE, B3 = GE, B4 = GE, B5 = GE
=====   C1 = EQ
=====   D1 = EQ
=====   E1 = LE, E2 = LE, E3 = LE, E4 = LE, E5 = LE
=====   F1 = LE
===>
                                                              X E D I T  1 FILE
 NETTY    DATA     A1   F 80  TRUNC=80 SIZE=100 LINE=18 COLUMN=1
=====   F1 = LE
=====   COST = FR
===== COLUMNS
=====   W11. A1 = 1, B1 = 1, E1 = 178, COST = 9704
=====   W12: A1 = 1, B2 = 1, E2 = 178, COST = 12146
=====   W13: A1 = 1, B3 = 1, E3 = 178, COST = 13122
=====   W14: A1 = 1, B4 = 1, E4 = 178, COST = 15075
=====   W15: A1 = 1, B5 = 1, E5 = 178, COST = 12146
=====   W21: A2 = 1, B1 = 1, E1 = 543 COST = 5310
=====   W22: A2 = 1, B2 = 1, E2 = 543 COST = 9704
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====   W23: A2 = 1, B3 = 1, E3 = 543 COST = 14100
=====   W24: A2 = 1, B4 = 1, E4 = 543 COST = 16052
=====   W25: A2 = 1, B5 = 1, E5 = 543 COST = 11169
=====   W31: A3 = 1, B1 = 1, E1 = 206, COST = 19958
=====   W32: A3 = 1, B2 = 1, E2 = 206, COST = 20934
=====   W33: A3 = 1, B3 = 1, E3 = 206, COST = 25817
=====   W34: A3 = 1, B4 = 1, E4 = 206, COST = 28746
=====   W35: A3 = 1, B5 = 1, E5 = 206, COST = 26793
=====   W41: A4 = 1, B1 = 1, E1 = 41, COST = 9216
===>
                                                              X E D I T  1 FILE
```

FIG.A.3

```
 NETTY    DATA     A1  F 80  TRUNC=80 SIZE=99 LINE=36 COLUMN=1
=====    W42: A4 = 1, B2 = 1, E2 = 41, COST = 5798
=====    W43: A4 = 1, B3 = 1, E3 = 41, COST = 9216
=====    W44: A4 = 1, B4 = 1, E4 = 41, COST = 12146
=====    W45: A4 = 1, B5 = 1, E5 = 41, COST = 10193
=====    W51: A5 = 1, B1 = 1, E1 = 3803, COST = 14100
=====    W52: A5 = 1, B2 = 1, E2 = 3803, COST = 7751
=====    W53: A5 = 1, B3 = 1, E3 = 3803, COST = 7751
=====    W54: A5 = 1, B4 = 1, E4 = 3803, COST = 11169
=====    W55: A5 = 1, B5 = 1, E5 = 3803, COST = 10193
=====    W61: A6 = 1, B1 = 1, E1 = 12, COST = 17028
       |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====    W62: A6 = 1, B2 = 1, E2 = 12, COST = 9216
=====    W63: A6 = 1, B3 = 1, E3 = 12, COST = 6775
=====    W64: A6 = 1, B4 = 1, E4 = 12, COST = 8240
=====    W65: A6 = 1, B5 = 1, E5 = 12, COST = 9216
=====    W71: A7 = 1, B1 = 1, E1 = 112, COST = 19958
=====    W72: A7 = 1, B2 = 1, E2 = 112, COST = 13122
=====    W73: A7 = 1, B3 = 1, E3 = 112, COST = 9216
=====    W74: A7 = 1, B4 = 1, E4 = 112, COST = 8240
=====    W75: A7 = 1, B5 = 1, E5 = 112, COST = 10193
===>
                                                           X E D I T  1 FILE
 NETTY    DATA     A1  F 80  TRUNC=80 SIZE=99 LINE=54 COLUMN=1
=====    W75: A7 = 1, B5 = 1, E5 = 112, COST = 10193
=====    W81: A8 = 1, B1 = 1, E1 = 198, COST = 15075
=====    W82: A8 = 1, B2 = 1, E2 = 198, COST = 11169
=====    W83: A8 = 1, B3 = 1, E3 = 198, COST = 6775
=====    W84: A8 = 1, B4 = 1, E4 = 198, COST = 6775
=====    W85: A8 = 1, B5 = 1, E5 = 198, COST = 5310
=====    W91: A9 = 1, B1 = 1, E1 = 78, COST = 21911
=====    W92: A9 = 1, B2 = 1, E2 = 78, COST = 12146
=====    W93: A9 = 1, B3 = 1, E3 = 78, COST = 9216
=====    W94: A9 = 1, B4 = 1, E4 = 78, COST = 7263
       |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====    W95: A9 = 1, B5 = 1, E5 = 78, COST = 7263
=====    X10: A1 = 1, F1 = 178, COST = 25305
=====    X20: A2 = 1, F1 = 543, COST = 30579
=====    X30: A3 = 1, F1 = 206, COST = 84909
=====    X40: A4 = 1, F1 = 41, COST = 24720
=====    X50: A5 = 1, F1 = 3803, COST = 27648
=====    X60: A6 = 1, F1 = 12, COST = 28821
=====    X70: A7 = 1, F1 = 112, COST = 34092
=====    X80: A8 = 1, F1 = 198, COST = 27648
===>
                                                           X E D I T  1 FILE
 NETTY    DATA     A1  F 80  TRUNC=80 SIZE=100 LINE=73 COLUMN=1
=====    X80: A8 = 1, F1 = 198, COST = 27648
=====    X90: A9 = 1, F1 = 78, COST = 29844
=====    Y12: C1 = 1, COST = 3235
=====    Y13: C1 = 1, COST = 5025
=====    Y14: C1 = 1, COST = 6002
=====    Y15: C1 = 1, COST = 5025
=====    Y23: C1 = 1, COST = 2747
=====    Y24: C1 = 1, COST = 3398
=====    Y25: C1 = 1, COST = 3072
=====    Y34: C1 = 1, COST = 2421
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====    Y35: C1 = 1, COST = 2421
=====    Y45: C1 = 1, COST = 2421
=====    Z10: B1 = -1, D1 = 1, COST = 3398
=====    Z20: B2 = -1, D1 = 1, COST = 2909
=====    Z30: B3 = -1, D1 = 1, COST = 3137
=====    Z40: B4 = -1, D1 = 1, COST = 3235
=====    Z50: B5 = -1, D1 = 1, COST = 3072
===== RHS
=====    CASE1: A1 = 1, A2 = 1, A3 = 1, A4 = 1, A5 = 1, A6 = 1, A7 = 1,
===>
                                                            X E D I T  1 FILE
```

```
 NETTY    DATA     A1  F 80  TRUNC=80 SIZE=100 LINE=91 COLUMN=1
=====    CASE1: A1 = 1, A2 = 1, A3 = 1, A4 = 1, A5 = 1, A6 = 1, A7 = 1,
=====      A8 = 1, A9 = 1, C1 = 10, D1 = 5, E1 = 4096, E2 = 4096, E3 = 4096,
=====      E4 = 4096, E5 = 4096, F1 = 128
===== BOUNDS
=====    FIRSTBND: W11 = (0,1), W12 = (0,1), W13 = (0,1), W14 = (0,1),
=====      W15 = (0,1), W21 = (0,1), W22 = (0,1), W23 = (0,1), W24 = (0,1),
=====      W25 = (0,1), W31 = (0,1), W32 = (0,1), W33 = (0,1), W34 = (0,1),
=====      W35 = (0,1), W41 = (0,1), W42 = (0,1), W43 = (0,1), W44 = (0,1),
=====      W45 = (0,1), W51 = (0,1), W52 = (0,1), W53 = (0,1), W54 = (0,1),
=====      W55 = (0,1), W61 = (0,1), W62 = (0,1), W63 = (0,1), W64 = (0,1),
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====      W65 = (0,1), W71 = (0,1), W72 = (0,1), W73 = (0,1), W74 = (0,1),
=====      W75 = (0,1), W81 = (0,1), W82 = (0,1), W83 = (0,1), W84 = (0,1),
=====      W85 = (0,1), W91 = (0,1), W92 = (0,1), W93 = (0,1), W94 = (0,1),
=====      W95 = (0,1), X10 = (0,1), X20 = (0,1), X30 = (0,1), X40 = (0,1),
=====      X50 = (0,1), X60 = (0,1), X70 = (0,1), X80 = (0,1), X90 = (0,1),
=====      Y12 = (0,1), Y13 = (0,1), Y14 = (0,1), Y15 = (0,1), Y23 = (0,1),
=====      Y24 = (0,1), Y25 = (0,1), Y34 = (0,1), Y35 = (0,1), Y45 = (0,1),
=====      Z10 = (0,1), Z20 = (0,1), Z30 = (0,1), Z40 = (0,1), Z50 = (0,1)
===== ENDATA
===> FILE
                                                            X E D I T  1 FILE
```

OUTPUT FROM SESAME, 4 LINKS AND INITIAL SCENARIO
****

```
                    ...NAME...     ...ACTIVITY...     DEFINED AS

                    FUNCTIONAL       104779.00        COST
                    RESTRAINTS                        CASE1
                    BOUNDS....                        FIRSTBND


SOLUTION

ROWS SECTION

 NUMBER  ...ROW..  AT   ...ACTIVITY...  SLACK ACTIVITY  ..LOWER LIMIT.  ..UPPER LIMIT.  .DUAL ACTIVITY

      1  A1        EQ     1.0000000          .            1.0000000       1.0000000      -9704.0000
      2  A2        EQ     1.0000000          .            1.0000000       1.0000000      -5310.0000
      3  A3        EQ     1.0000000          .            1.0000000       1.0000000      -20934.000
      4  A4        EQ     1.0000000          .            1.0000000       1.0000000      -9216.0000
      5  A5        EQ     1.0000000          .            1.0000000       1.0000000      -7751.0000
      6  A6        EQ     1.0000000          .            1.0000000       1.0000000      -8240.0000
      7  A7        EQ     1.0000000          .            1.0000000       1.0000000      -8240.0000
      8  A8        EQ     1.0000000          .            1.0000000       1.0000000      -5310.0000
      9  A9        EQ     1.0000000          .            1.0000000       1.0000000      -7263.0000
     10  B1        BS     3.0000000     -3.0000000            .             NONE              .
     11  B2        BS     1.0000000     -1.0000000            .             NONE              .
     12  B3        BS         .              .                .             NONE              .
     13  B4        BS         .              .                .             NONE              .
     14  B5        BS     1.0000000     -1.0000000            .             NONE              .
     15  C1        EQ     6.0000000          .            6.0000000       6.0000000      -3235.0000
     16  D1        EQ     4.0000000          .            4.0000000       4.0000000      -3235.0000
     17  E1        BS     927.00000      3169.0000          NONE          4096.0000           .
     18  E2        BS     3844.0000      252.00000          NONE          4096.0000           .
     19  E3        BS     12.000000      4084.0000          NONE          4096.0000           .
     20  E4        BS     112.00000      3984.0000          NONE          4096.0000           .
     21  E5        BS     276.00000      3820.0000          NONE          4096.0000           .
     22  F1        BS         .          128.00000          NONE          128.00000           .
     23  COST      BS     104779.00     -104779.00          NONE            NONE          1.0000000
```

FIG.A.4

```
COLUMNS SECTION

  NUMBER  .COLUMN   AT   ...ACTIVITY...  ..INPUT COST..  ..LOWER LIMIT.  ..UPPER LIMIT.  .REDUCED COST.

      24  W11       BS     1.0000000       9704.0000           .           1.0000000           .
      25  W12       LL         .           12146.000           .           1.0000000       2442.0000
      26  W13       LL         .           13122.000           .           1.0000000       3418.0000
      27  W14       LL         .           15075.000           .           1.0000000       5371.0000
      28  W15       LL         .           12146.000           .           1.0000000       2442.0000
      29  W21       BS     1.0000000       5310.0000           .           1.0000000           .
      30  W22       LL         .           9704.0000           .           1.0000000       4394.0000
      31  W23       LL         .           14100.000           .           1.0000000       8790.0000
      32  W24       LL         .           16052.000           .           1.0000000       10742.000
      33  W25       LL         .           11169.000           .           1.0000000       5859.0000
      34  W31       UL     1.0000000       19958.000           .           1.0000000      -976.00000
      35  W32       BS         .           20934.000           .           1.0000000           .
      36  W33       LL         .           25817.000           .           1.0000000       4883.0000
      37  W34       LL         .           28746.000           .           1.0000000       7812.0000
      38  W35       LL         .           26793.000           .           1.0000000       5859.0000
      39  W41       BS         .           9216.0000           .           1.0000000           .
      40  W42       UL     1.0000000       5798.0000           .           1.0000000      -3418.0000
A     41  W43       LL         .           9216.0000           .           1.0000000           .
      42  W44       LL         .           12146.000           .           1.0000000       2930.0000
      43  W45       LL         .           10193.000           .           1.0000000       977.00000
      44  W51       LL         .           14100.000           .           1.0000000       6349.0000
      45  W52       BS     1.0000000       7751.0000           .           1.0000000           .
A     46  W53       LL         .           7751.0000           .           1.0000000           .
      47  W54       LL         .           11169.000           .           1.0000000       3418.0000
      48  W55       LL         .           10193.000           .           1.0000000       2442.0000
      49  W61       LL         .           17028.000           .           1.0000000       8788.0000
      50  W62       LL         .           9216.0000           .           1.0000000       976.00000
      51  W63       UL     1.0000000       6775.0000           .           1.0000000      -1465.0000
      52  W64       BS         .           8240.0000           .           1.0000000           .
      53  W65       LL         .           9216.0000           .           1.0000000       976.00000
      54  W71       LL         .           19958.000           .           1.0000000       11718.000
      55  W72       LL         .           13122.000           .           1.0000000       4882.0000
      56  W73       LL         .           9216.0000           .           1.0000000       976.00000
      57  W74       BS     1.0000000       8240.0000           .           1.0000000           .
      58  W75       LL         .           10193.000           .           1.0000000       1953.0000
      59  W81       LL         .           15075.000           .           1.0000000       9765.0000
      60  W82       LL         .           11169.000           .           1.0000000       5859.0000
      61  W83       LL         .           6775.0000           .           1.0000000       1465.0000
      62  W84       LL         .           6775.0000           .           1.0000000       1465.0000
      63  W85       BS     1.0000000       5310.0000           .           1.0000000           .
      64  W91       LL         .           21911.000           .           1.0000000       14648.000
      65  W92       LL         .           12146.000           .           1.0000000       4883.0000
      66  W93       LL         .           9216.0000           .           1.0000000       1953.0000
      67  W94       BS         .           7263.0000           .           1.0000000           .
A     68  W95       UL     1.0000000       7263.0000           .           1.0000000           .
      69  X10       LL         .           25305.000           .           1.0000000       15601.000
      70  X20       LL         .           30579.000           .           1.0000000       25269.000
      71  X30       LL         .           84909.000           .           1.0000000       63975.000
      72  X40       LL         .           24720.000           .           1.0000000       15504.000
      73  X50       LL         .           27648.000           .           1.0000000       19897.000
      74  X60       LL         .           28821.000           .           1.0000000       20581.000
      75  X70       LL         .           34092.000           .           1.0000000       25852.000
      76  X80       LL         .           27648.000           .           1.0000000       22338.000
      77  X90       LL         .           29844.000           .           1.0000000       22581.000
      78  Y12       BS     1.0000000       3235.0000           .           1.0000000           .
      79  Y13       LL         .           5025.0000           .           1.0000000       1790.0000
      80  Y14       LL         .           6002.0000           .           1.0000000       2767.0000
      81  Y15       LL         .           5025.0000           .           1.0000000       1790.0000
      82  Y23       UL     1.0000000       2747.0000           .           1.0000000      -488.00000
      83  Y24       LL         .           3398.0000           .           1.0000000       163.00000
      84  Y25       UL     1.0000000       3072.0000           .           1.0000000      -163.00000
      85  Y34       UL     1.0000000       2421.0000           .           1.0000000      -814.00000
      86  Y35       UL     1.0000000       2421.0000           .           1.0000000      -814.00000
      87  Y45       UL     1.0000000       2421.0000           .           1.0000000      -814.00000
      88  Z10       LL         .           3398.0000           .           1.0000000       163.00000
      89  Z20       UL     1.0000000       2909.0000           .           1.0000000      -326.00000
      90  Z30       UL     1.0000000       3137.0000           .           1.0000000      -98.000000
      91  Z40       BS     1.0000000       3235.0000           .           1.0000000           .
      92  Z50       UL     1.0000000       3072.0000           .           1.0000000      -163.00000
```

OUTPUT FROM SESAME, 4 LINKS AND GROWTH SCENARIO
****

```
                        ...NAME...    ...ACTIVITY...      DEFINED AS

                        FUNCTIONAL      104779.00         COST
                        RESTRAINTS                        CASE1
                        BOUNDS....                        FIRSTBND


SOLUTION

ROWS SECTION

  NUMBER  ...ROW..  AT   ...ACTIVITY...  SLACK ACTIVITY  ..LOWER LIMIT.  ..UPPER LIMIT.  .DUAL ACTIVITY

       1  A1        EQ    1.0000000          .            1.0000000       1.0000000       -9704.0000
       2  A2        EQ    1.0000000          .            1.0000000       1.0000000       -5310.0000
       3  A3        EQ    1.0000000          .            1.0000000       1.0000000       -20934.000
       4  A4        EQ    1.0000000          .            1.0000000       1.0000000       -5798.0000
       5  A5        EQ    1.0000000          .            1.0000000       1.0000000       -7751.0000
       6  A6        EQ    1.0000000          .            1.0000000       1.0000000       -8240.0000
       7  A7        EQ    1.0000000          .            1.0000000       1.0000000       -8240.0000
       8  A8        EQ    1.0000000          .            1.0000000       1.0000000       -5310.0000
       9  A9        EQ    1.0000000          .            1.0000000       1.0000000       -7263.0000
      10  B1        BS    3.0000000      -3.0000000           .              NONE               .
      11  B2        BS    .97142857      -.97142857           .              NONE               .
      12  B3        BS    .02857143      -.02857143           .              NONE               .
      13  B4        BS        .              .                .              NONE               .
      14  B5        BS    1.0000000      -1.0000000           .              NONE               .
      15  C1        EQ    6.0000000          .            6.0000000       6.0000000       -3235.0000
      16  D1        EQ    4.0000000          .            4.0000000       4.0000000       -3235.0000
      17  E1        BS    1593.0000       3683.0000         NONE          5276.0000            .
A     18  E2        UL    5276.0000          .              NONE          5276.0000            .
      19  E3        BS    178.00000       5098.0000         NONE          5276.0000            .
      20  E4        BS    366.00000       4910.0000         NONE          5276.0000            .
      21  E5        BS    1318.0000       3958.0000         NONE          5276.0000            .
      22  F1        BS        .           256.00000         NONE          256.00000            .
      23  COST      BS    104779.00      -104779.00         NONE            NONE           1.0000000
```

FIG.A.5

```
COLUMNS SECTION

 NUMBER  .COLUMN    AT    ...ACTIVITY...  ..INPUT COST..  ..LOWER LIMIT.  ..UPPER LIMIT.  .REDUCED COST.

     24  W11        BS     1.0000000       9704.0000          .            1.0000000            .
     25  W12        LL         .           12146.000          .            1.0000000        2442.0000
     26  W13        LL         .           13122.000          .            1.0000000        3418.0000
     27  W14        LL         .           15075.000          .            1.0000000        5371.0000
     28  W15        LL         .           12146.000          .            1.0000000        2442.0000
     29  W21        BS     1.0000000       5310.0000          .            1.0000000            .
     30  W22        LL         .           9704.0000          .            1.0000000        4394.0000
     31  W23        LL         .           14100.000          .            1.0000000        8790.0000
     32  W24        LL         .           16052.000          .            1.0000000        10742.000
     33  W25        LL         .           11169.000          .            1.0000000        5859.0000
     34  W31        UL     1.0000000       19958.000          .            1.0000000        -976.00000
     35  W32        BS         .           20934.000          .            1.0000000            .
     36  W33        LL         .           25817.000          .            1.0000000        4883.0000
     37  W34        LL         .           28746.000          .            1.0000000        7812.0000
     38  W35        LL         .           26793.000          .            1.0000000        5859.0000
     39  W41        LL         .           9216.0000          .            1.0000000        3418.0000
     40  W42        BS     1.0000000       5798.0000          .            1.0000000            .
     41  W43        LL         .           9216.0000          .            1.0000000        3418.0000
     42  W44        LL         .           12146.000          .            1.0000000        6348.0000
     43  W45        LL         .           10193.000          .            1.0000000        4395.0000
     44  W51        LL         .           14100.000          .            1.0000000        6349.0000
     45  W52        BS      .97142857      7751.0000          .            1.0000000            .
     46  W53        BS      .02857143      7751.0000          .            1.0000000            .
     47  W54        LL         .           11169.000          .            1.0000000        3418.0000
     48  W55        LL         .           10193.000          .            1.0000000        2442.0000
     49  W61        LL         .           17028.000          .            1.0000000        8788.0000
     50  W62        LL         .           9216.0000          .            1.0000000        976.00000
     51  W63        UL     1.0000000       6775.0000          .            1.0000000        -1465.0000
     52  W64        BS         .           8240.0000          .            1.0000000            .
     53  W65        LL         .           9216.0000          .            1.0000000        976.00000
     54  W71        LL         .           19958.000          .            1.0000000        11718.000
     55  W72        LL         .           13122.000          .            1.0000000        4882.0000
     56  W73        LL         .           9216.0000          .            1.0000000        976.00000
     57  W74        BS     1.0000000       8240.0000          .            1.0000000            .
     58  W75        LL         .           10193.000          .            1.0000000        1953.0000
     59  W81        LL         .           15075.000          .            1.0000000        9765.0000
     60  W82        LL         .           11169.000          .            1.0000000        5859.0000
     61  W83        LL         .           6775.0000          .            1.0000000        1465.0000
     62  W84        LL         .           6775.0000          .            1.0000000        1465.0000
     63  W85        BS     1.0000000       5310.0000          .            1.0000000            .
     64  W91        LL         .           21911.000          .            1.0000000        14648.000
     65  W92        LL         .           12146.000          .            1.0000000        4883.0000
     66  W93        LL         .           9216.0000          .            1.0000000        1953.0000
     67  W94        BS         .           7263.0000          .            1.0000000            .
A    68  W95        UL     1.0000000       7263.0000          .            1.0000000            .
     69  X10        LL         .           25305.000          .            1.0000000        15601.000
     70  X20        LL         .           30579.000          .            1.0000000        25269.000
     71  X30        LL         .           84909.000          .            1.0000000        63975.000
     72  X40        LL         .           24720.000          .            1.0000000        18922.000
     73  X50        LL         .           27648.000          .            1.0000000        19897.000
     74  X60        LL         .           28821.000          .            1.0000000        20581.000
     75  X70        LL         .           34092.000          .            1.0000000        25852.000
     76  X80        LL         .           27648.000          .            1.0000000        22338.000
     77  X90        LL         .           29844.000          .            1.0000000        22581.000
     78  Y12        BS     1.0000000       3235.0000          .            1.0000000            .
     79  Y13        LL         .           5025.0000          .            1.0000000        1790.0000
     80  Y14        LL         .           6002.0000          .            1.0000000        2767.0000
     81  Y15        LL         .           5025.0000          .            1.0000000        1790.0000
     82  Y23        UL     1.0000000       2747.0000          .            1.0000000        -488.00000
     83  Y24        LL         .           3398.0000          .            1.0000000        163.00000
     84  Y25        UL     1.0000000       3072.0000          .            1.0000000        -163.00000
     85  Y34        UL     1.0000000       2421.0000          .            1.0000000        -814.00000
     86  Y35        UL     1.0000000       2421.0000          .            1.0000000        -814.00000
     87  Y45        UL     1.0000000       2421.0000          .            1.0000000        -814.00000
     88  Z10        LL         .           3398.0000          .            1.0000000        163.00000
     89  Z20        UL     1.0000000       2909.0000          .            1.0000000        -326.00000
     90  Z30        UL     1.0000000       3137.0000          .            1.0000000        -98.000000
     91  Z40        BS     1.0000000       3235.0000          .            1.0000000            .
     92  Z50        UL     1.0000000       3072.0000          .            1.0000000        -163.00000
```

OUTPUT FROM SESAME, 5 LINKS AND INITIAL SCENARIO
****

```
                  ...NAME...     ...ACTIVITY...     DEFINED AS

                  FUNCTIONAL       127627.00        COST
                  RESTRAINTS                        CASE1
                  BOUNDS....                        FIRSTBND


SOLUTION

ROWS SECTION

  NUMBER  ...ROW..  AT   ...ACTIVITY...  SLACK ACTIVITY  ..LOWER LIMIT.  ..UPPER LIMIT.  .DUAL ACTIVITY

       1  A1        EQ     1.0000000          .           1.0000000       1.0000000      -12146.000
       2  A2        EQ     1.0000000          .           1.0000000       1.0000000      -9704.0000
       3  A3        EQ     1.0000000          .           1.0000000       1.0000000      -19958.000
       4  A4        EQ     1.0000000          .           1.0000000       1.0000000      -9216.0000
       5  A5        EQ     1.0000000          .           1.0000000       1.0000000      -7751.0000
       6  A6        EQ     1.0000000          .           1.0000000       1.0000000      -6775.0000
       7  A7        EQ     1.0000000          .           1.0000000       1.0000000      -8240.0000
       8  A8        EQ     1.0000000          .           1.0000000       1.0000000      -5310.0000
       9  A9        EQ     1.0000000          .           1.0000000       1.0000000      -7263.0000
      10  B1        BS     2.0000000     -2.0000000           .              NONE             .
      11  B2        BS         .              .               .              NONE             .
      12  B3        BS     1.0000000     -1.0000000           .              NONE             .
A     13  B4        LL         .              .               .              NONE             .
      14  B5        BS     1.0000000     -1.0000000           .              NONE             .
      15  C1        EQ     10.000000          .           10.000000       10.000000      -6002.0000
      16  D1        EQ     5.0000000          .           5.0000000       5.0000000      -3398.0000
      17  E1        BS     927.00000      3169.0000          NONE         4096.0000            .
      18  E2        BS     41.000000      4055.0000          NONE         4096.0000            .
      19  E3        BS     3815.0000      281.00000          NONE         4096.0000            .
      20  E4        BS     112.00000      3984.0000          NONE         4096.0000            .
      21  E5        BS     276.00000      3820.0000          NONE         4096.0000            .
      22  F1        BS         .          128.00000          NONE         128.00000            .
      23  COST      BS     127627.00     -127627.00          NONE           NONE         1.0000000
```

FIG.A.6

```
COLUMNS SECTION

  NUMBER  .COLUMN   AT   ...ACTIVITY...  ..INPUT COST..  ..LOWER LIMIT.  ..UPPER LIMIT.  .REDUCED COST.

      24  W11       UL     1.0000000       9704.0000          .            1.0000000      -2442.0000
A     25  W12       LL         .          12146.000           .            1.0000000          .
      26  W13       LL         .          13122.000           .            1.0000000       976.00000
      27  W14       LL         .          15075.000           .            1.0000000       2929.0000
      28  W15       BS         .          12146.000           .            1.0000000          .
      29  W21       UL     1.0000000       5310.0000          .            1.0000000      -4394.0000
      30  W22       BS         .           9704.0000          .            1.0000000          .
      31  W23       LL         .          14100.000           .            1.0000000       4396.0000
      32  W24       LL         .          16052.000           .            1.0000000       6348.0000
      33  W25       LL         .          11169.000           .            1.0000000       1465.0000
      34  W31       BS     1.0000000      19958.000           .            1.0000000          .
      35  W32       LL         .          20934.000           .            1.0000000       976.00000
      36  W33       LL         .          25817.000           .            1.0000000       5859.0000
      37  W34       LL         .          28746.000           .            1.0000000       8788.0000
      38  W35       LL         .          26793.000           .            1.0000000       6835.0000
      39  W41       BS         .           9216.0000          .            1.0000000          .
      40  W42       UL     1.0000000       5798.0000          .            1.0000000      -3418.0000
A     41  W43       LL         .           9216.0000          .            1.0000000          .
      42  W44       LL         .          12146.000           .            1.0000000       2930.0000
      43  W45       LL         .          10193.000           .            1.0000000       977.00000
      44  W51       LL         .          14100.000           .            1.0000000       6349.0000
      45  W52       BS         .           7751.0000          .            1.0000000          .
A     46  W53       UL     1.0000000       7751.0000          .            1.0000000          .
      47  W54       LL         .          11169.000           .            1.0000000       3418.0000
      48  W55       LL         .          10193.000           .            1.0000000       2442.0000
      49  W61       LL         .          17028.000           .            1.0000000       10253.000
      50  W62       LL         .           9216.0000          .            1.0000000       2441.0000
      51  W63       BS     1.0000000       6775.0000          .            1.0000000          .
      52  W64       LL         .           8240.0000          .            1.0000000       1465.0000
      53  W65       LL         .           9216.0000          .            1.0000000       2441.0000
      54  W71       LL         .          19958.000           .            1.0000000       11718.000
      55  W72       LL         .          13122.000           .            1.0000000       4882.0000
      56  W73       LL         .           9216.0000          .            1.0000000       976.00000
      57  W74       BS     1.0000000       8240.0000          .            1.0000000          .
      58  W75       LL         .          10193.000           .            1.0000000       1953.0000
      59  W81       LL         .          15075.000           .            1.0000000       9765.0000
      60  W82       LL         .          11169.000           .            1.0000000       5859.0000
      61  W83       LL         .           6775.0000          .            1.0000000       1465.0000
      62  W84       LL         .           6775.0000          .            1.0000000       1465.0000
      63  W85       BS     1.0000000       5310.0000          .            1.0000000          .
      64  W91       LL         .          21911.000           .            1.0000000       14648.000
      65  W92       LL         .          12146.000           .            1.0000000       4883.0000
      66  W93       LL         .           9216.0000          .            1.0000000       1953.0000
      67  W94       BS         .           7263.0000          .            1.0000000          .
      68  W95       BS     1.0000000       7263.0000          .            1.0000000          .
      69  X10       LL         .          25305.000           .            1.0000000       13159.000
      70  X20       LL         .          30579.000           .            1.0000000       20875.000
      71  X30       LL         .          84909.000           .            1.0000000       64951.000
      72  X40       LL         .          24720.000           .            1.0000000       15504.000
      73  X50       LL         .          27648.000           .            1.0000000       19897.000
      74  X60       LL         .          28821.000           .            1.0000000       22046.000
      75  X70       LL         .          34092.000           .            1.0000000       25852.000
      76  X80       LL         .          27648.000           .            1.0000000       22338.000
      77  X90       LL         .          29844.000           .            1.0000000       22581.000
      78  Y12       UL     1.0000000       3235.0000          .            1.0000000      -2767.0000
      79  Y13       UL     1.0000000       5025.0000          .            1.0000000      -977.00000
      80  Y14       BS     1.0000000       6002.0000          .            1.0000000          .
      81  Y15       UL     1.0000000       5025.0000          .            1.0000000      -977.00000
      82  Y23       UL     1.0000000       2747.0000          .            1.0000000      -3255.0000
      83  Y24       UL     1.0000000       3398.0000          .            1.0000000      -2604.0000
      84  Y25       UL     1.0000000       3072.0000          .            1.0000000      -2930.0000
      85  Y34       UL     1.0000000       2421.0000          .            1.0000000      -3581.0000
      86  Y35       UL     1.0000000       2421.0000          .            1.0000000      -3581.0000
      87  Y45       UL     1.0000000       2421.0000          .            1.0000000      -3581.0000
      88  Z10       BS     1.0000000       3398.0000          .            1.0000000          .
      89  Z20       UL     1.0000000       2909.0000          .            1.0000000      -489.00000
      90  Z30       UL     1.0000000       3137.0000          .            1.0000000      -261.00000
      91  Z40       UL     1.0000000       3235.0000          .            1.0000000      -163.00000
      92  Z50       UL     1.0000000       3072.0000          .            1.0000000      -326.00000
```

```
                    ...NAME...    ...ACTIVITY...      DEFINED AS

                    FUNCTIONAL       127627.00        COST
                    RESTRAINTS                        CASE1
                    BOUNDS....                        FIRSTBND


SOLUTION

ROWS SECTION

 NUMBER  ...ROW..  AT   ...ACTIVITY...  SLACK ACTIVITY  ..LOWER LIMIT.  ..UPPER LIMIT.  .DUAL ACTIVITY

      1  A1        EQ      1.0000000          .           1.0000000       1.0000000      -12146.000
      2  A2        EQ      1.0000000          .           1.0000000       1.0000000      -9704.0000
      3  A3        EQ      1.0000000          .           1.0000000       1.0000000      -19958.000
      4  A4        EQ      1.0000000          .           1.0000000       1.0000000      -9216.0000
      5  A5        EQ      1.0000000          .           1.0000000       1.0000000      -7751.0000
      6  A6        EQ      1.0000000          .           1.0000000       1.0000000      -6775.0000
      7  A7        EQ      1.0000000          .           1.0000000       1.0000000      -8240.0000
      8  A8        EQ      1.0000000          .           1.0000000       1.0000000      -5310.0000
      9  A9        EQ      1.0000000          .           1.0000000       1.0000000      -7263.0000
     10  B1        BS      2.0000000     -2.0000000           .              NONE               .
     11  B2        BS          .              .               .              NONE               .
     12  B3        BS      1.0000000     -1.0000000           .              NONE               .
     13  B4        BS      1.0000000     -1.0000000           .              NONE               .
     14  B5        BS          .              .               .              NONE               .
     15  C1        EQ      10.000000          .           10.000000       10.000000      -6002.0000
     16  D1        EQ      5.0000000          .           5.0000000       5.0000000      -3398.0000
     17  E1        BS      1593.0000      3683.0000          NONE         5276.0000             .
     18  E2        BS      244.00000      5032.0000          NONE         5276.0000             .
     19  E3        BS      5210.0000      66.000000          NONE         5276.0000             .
     20  E4        BS      504.00000      4772.0000          NONE         5276.0000             .
     21  E5        BS      1180.0000      4096.0000          NONE         5276.0000             .
     22  F1        BS          .          256.00000          NONE         256.00000             .
     23  COST      BS      127627.00     -127627.00          NONE            NONE         1.0000000
```

```
COLUMNS SECTION

 NUMBER  .COLUMN   AT   ...ACTIVITY...  ..INPUT COST.. ..LOWER LIMIT.. ..UPPER LIMIT. .REDUCED COST.

     24  W11       UL     1.0000000       9704.0000          .            1.0000000     -2442.0000
A    25  W12       LL         .           12146.000          .            1.0000000          .
     26  W13       LL         .           13122.000          .            1.0000000      976.00000
     27  W14       LL         .           15075.000          .            1.0000000      2929.0000
     28  W15       BS         .           12146.000          .            1.0000000          .
     29  W21       UL     1.0000000       5310.0000          .            1.0000000     -4394.0000
     30  W22       BS         .           9704.0000          .            1.0000000          .
     31  W23       LL         .           14100.000          .            1.0000000      4396.0000
     32  W24       LL         .           16052.000          .            1.0000000      6348.0000
     33  W25       LL         .           11169.000          .            1.0000000      1465.0000
     34  W31       BS     1.0000000       19958.000          .            1.0000000          .
     35  W32       LL         .           20934.000          .            1.0000000      976.00000
     36  W33       LL         .           25817.000          .            1.0000000      5859.0000
     37  W34       LL         .           28746.000          .            1.0000000      8788.0000
     38  W35       LL         .           26793.000          .            1.0000000      6835.0000
     39  W41       BS         .           9216.0000          .            1.0000000          .
     40  W42       UL     1.0000000       5798.0000          .            1.0000000     -3418.0000
A    41  W43       LL         .           9216.0000          .            1.0000000          .
     42  W44       LL         .           12146.000          .            1.0000000      2930.0000
     43  W45       LL         .           10193.000          .            1.0000000      977.00000
     44  W51       LL         .           14100.000          .            1.0000000      6349.0000
A    45  W52       LL         .           7751.0000          .            1.0000000          .
     46  W53       BS     1.0000000       7751.0000          .            1.0000000          .
     47  W54       LL         .           11169.000          .            1.0000000      3418.0000
     48  W55       LL         .           10193.000          .            1.0000000      2442.0000
     49  W61       LL         .           17028.000          .            1.0000000      10253.000
     50  W62       LL         .           8216.0000          .            1.0000000      2441.0000
     51  W63       BS     1.0000000       6775.0000          .            1.0000000          .
     52  W64       LL         .           8240.0000          .            1.0000000      1465.0000
     53  W65       LL         .           9216.0000          .            1.0000000      2441.0000
     54  W71       LL         .           19958.000          .            1.0000000      11718.000
     55  W72       LL         .           13122.000          .            1.0000000      4882.0000
     56  W73       LL         .           9216.0000          .            1.0000000      976.00000
     57  W74       BS     1.0000000       8240.0000          .            1.0000000          .
     58  W75       LL         .           10193.000          .            1.0000000      1953.0000
     59  W81       LL         .           15075.000          .            1.0000000      9765.0000
     60  W82       LL         .           11169.000          .            1.0000000      5859.0000
     61  W83       LL         .           6775.0000          .            1.0000000      1465.0000
     62  W84       LL         .           6775.0000          .            1.0000000      1465.0000
     63  W85       BS     1.0000000       5310.0000          .            1.0000000          .
     64  W91       LL         .           21911.000          .            1.0000000      14648.000
     65  W92       LL         .           12146.000          .            1.0000000      4883.0000
     66  W93       LL         .           9216.0000          .            1.0000000      1953.0000
     67  W94       BS     1.0000000       7263.0000          .            1.0000000          .
A    68  W95       LL         .           7263.0000          .            1.0000000          .
     69  X10       LL         .           25305.000          .            1.0000000      13159.000
     70  X20       LL         .           30579.000          .            1.0000000      20875.000
     71  X30       LL         .           84909.000          .            1.0000000      64951.000
     72  X40       LL         .           24720.000          .            1.0000000      15504.000
     73  X50       LL         .           27648.000          .            1.0000000      19897.000
     74  X60       LL         .           28821.000          .            1.0000000      22046.000
     75  X70       LL         .           34092.000          .            1.0000000      25852.000
     76  X80       LL         .           27648.000          .            1.0000000      22338.000
     77  X90       LL         .           29844.000          .            1.0000000      22581.000
     78  Y12       UL     1.0000000       3235.0000          .            1.0000000     -2767.0000
     79  Y13       UL     1.0000000       5025.0000          .            1.0000000     -977.00000
     80  Y14       BS     1.0000000       6002.0000          .            1.0000000          .
     81  Y15       UL     1.0000000       5025.0000          .            1.0000000     -977.00000
     82  Y23       UL     1.0000000       2747.0000          .            1.0000000     -3255.0000
     83  Y24       UL     1.0000000       3398.0000          .            1.0000000     -2604.0000
     84  Y25       UL     1.0000000       3072.0000          .            1.0000000     -2930.0000
     85  Y34       UL     1.0000000       2421.0000          .            1.0000000     -3581.0000
     86  Y35       UL     1.0000000       2421.0000          .            1.0000000     -3581.0000
     87  Y45       UL     1.0000000       2421.0000          .            1.0000000     -3581.0000
     88  Z10       BS     1.0000000       3398.0000          .            1.0000000          .
     89  Z20       UL     1.0000000       2909.0000          .            1.0000000     -489.00000
     90  Z30       UL     1.0000000       3137.0000          .            1.0000000     -261.00000
     91  Z40       UL     1.0000000       3235.0000          .            1.0000000     -163.00000
     92  Z50       UL     1.0000000       3072.0000          .            1.0000000     -326.00000
```

APPENDIX B

THE SEQUENTIAL t-TEST ALGORITHM

THE DERIVATION OF THE ALGORITHM

Following the procedure indicated in reference [AMS749] cited in Chapter 5 and using the same notation, by means of the transformations

$$u = \sum_{i=1}^{n} \sqrt{(xi - \theta o)^2} \; / \; \sigma$$

and
.tp 6

$$v = \sum_{i=1}^{n} (xi - \theta 0) / \sqrt{\sum_{i=1}^{n} (xi - \theta o)^2}$$

equation (5.2) becomes

$$\frac{\beta}{1-\alpha} < \exp\left(-\frac{n\delta^2}{2}\right) F\left(\frac{n-1}{2}, \frac{1}{2}; \frac{\delta^2 v^2}{2}\right) < \frac{1-\beta}{\alpha} \qquad (B.1)$$

where $F(\alpha,\gamma;x)$ is the confluent hypergeometric function defined as

$$F(\alpha,\gamma;x) = \sum_{j=0}^{\infty} \frac{\Gamma(\gamma)\ \Gamma(\alpha+j)\ x^j}{\Gamma(\alpha)\ \Gamma(\gamma+j)\ j!}$$

Using the recurent relation

$$F(\alpha+1,\gamma;x) = \frac{x+2\alpha-\gamma}{\alpha} F(\alpha,\gamma;x) + \frac{\gamma\alpha}{\alpha} F(\alpha-1,;x)$$

and with

$$\alpha = \frac{n-3}{2} \quad ; \gamma = \frac{1}{2} ; \quad x = \frac{\delta^2 v^2}{2}$$

the following is obtained

$$F(\frac{n-1}{2},\frac{1}{2};x) = An(n)\ F(\frac{n-3}{2},\frac{1}{2};x) + Bn(n)\ F(\frac{n-5}{2},\frac{1}{2};x)$$

where

$$An(n) = \frac{2(x+n)-7}{n-3} \quad \text{and} \quad Bn(n) = \frac{4-n}{n-3} .$$

From the above relations it is seen that it is not possible with this formulation to reach a decision before the fifth measurement is taken.

Equation B.1 can now be written

$$\frac{\beta}{1-\alpha} < \exp[\ln F(\frac{n-1}{2},\frac{1}{2};\frac{\delta^2 v^2}{2}) - \frac{n\delta^2}{2}] < \frac{1-\beta}{\alpha} \qquad (B.2)$$

The initial values are (reference [AMS349], Chapter 5)

if n is even $F(\frac{1}{2},\frac{1}{2};x) = \exp(x)$ and

$$F(\frac{3}{2},\frac{1}{2};x) = \exp(x)(1+2x)$$

if n is odd $F(0,\frac{1}{2};x) = 1$

The only calçulation left is the corresponding one for $F(1,1/2;x)$. Also from [AMS349]

$$F\left(\frac{m}{2},\frac{1}{2};x\right) = \exp(x)\left[1 + (m-1)x + \frac{(m-1)(m-3)}{1.3}\,\frac{x^2}{2!} + \ldots\right]$$

which can be written, for n = 3 (or m = n-1 = 2)

$$F\left(1,\frac{1}{2};x\right) = \exp(x)\sum_{i=1}^{\infty}\frac{\prod_{j=1}^{2i}(3-2j)\;x^{i}}{\prod_{j=1}^{2i-1}(2j-1)\;i!} \qquad (B.3)$$

However, for real-time applications it is more convenient to find a recurrent relation to accelerate the computations. It can be shown that (B.3) can also be written as

$$F\left(1,\frac{1}{2};x\right) = \exp(x)\left[1 + \sum_{r=1}^{\infty}U(r,x)\right] \qquad (B.4)$$

where

$$U(r,x) = -\frac{1}{r}\,\frac{2r-3}{2r-1}\,x\,U(r-1,x) \quad \text{and} \quad U(1,x) = x \qquad (B.5)$$

The parameter r must be adjusted to obtain the desired degree of accuracy. Although if r increases more accuraccy will be obtained, the rate of convergence [U(r+1,x)-U(r,x)]÷[U(r,x)-U(r-1,x)] decreases approximately as of 1/r. Then, increasing r will slow down the calculations and will not necessarily increase the accuracy too much. The tradeoff will depend on the particular application.*

AN EXAMPLE IMPLEMENTED IN A HP-67

---

*The previous infinite series contained in (B.4) may be convergent or divergent depending on the value of δ. That can be verified using the above definition of x and from Cauchy's test of convergence operating on eq. (B.5). The maximun δ is found to be $\delta\text{max} = 3.146/v$. For values of δ larger than δmax the series will diverge. The selection of δ can be incorporated into the actual implementation.

The flowchart shown in FIG.B.1 indicates how the algorithm is actually executed. In FIG.B.2 the instructions to perform it are indicated and in FIG.B.3 the algorithm is written down in the HP-67 program language, with an indication of how the registers are used.

The example can be taken from reference [AMS749] (cited in Chapter 5), page XV. In that publication the example is described in the following terms: "suppose that the measurements of one characteristic of the products from a certain" "process should have values centering about 1300, ie if the process is working properly the population of values from which the values for a given lot are taken should be divided into two equal parts by the number 1300, half the population should have values less than 1300 and half greater. The values are assumed to be normally distributed. If the process produces a population of measurements so different that 15 percent or less have values below 1300 and 85 percent or more above, or if 15 percent or less have values above 1300 and 85 percent or more below, we want to be reasonably sure that the lots produced under these conditions are rejected"."Sampling is necessary. We cannot be certain of correct decisions by sampling, but we can set probabilities of certain incorrect decisions. In this case it is required that lots of acceptable products be rejected only 5 percent of the time and that lots for which 1300 divides the population in the ratio 15:85 or worse be accepted not more that 5 percent of the time".

By referring to table 2 of the same publication it is seen that a $\delta = 1$ corresponds to a division of a normal population approximately in the ratio 1587:8413, close enough to what is required.

It will be assumed that the number of iterations performed at the fifth measurement will be 10, then, rlast = 10.

The test is initiated with $\theta o = 1300$, $\alpha = 0.05$, $\beta = 0.05$, $\delta = 1$ and rlast = 10.

How the testing proceeds is shown in FIG.B.4. The test terminates before the 14th sample is taken by <u>accepting</u> the batch of measurements as representative of the hypothesis made.

As an extention of the example presented in the cited reference, the test has been repeated for different values of δ to show how this parameter affects the length of the test. Remember,from Chapter 5, that this value can be set arbitrarily in a limited range, defined by characteristics of the real process to be sampled, and that selection could be incorporated in the implementation of the program.

```
                          !Load constants!
                          --------------
                                !<------------*
                                v
                          !   Reset n    !
                          --------------
                                !<------------.---- **
                                              !
                          !Obtain sample !    !
                          ! Calculate z  !    !
                          --------------      !
                                !             !
                                          <   !
                           ! Test n < 5 !-------
                            ------------
                                ! >
         odd                                       even
   .-----------!Test n odd or even!------------.
   !            ------------------             !
   v                                           v
.----------------------------.   .--------------------------------.
! n=5 .initialization        !   ! n=6 .initialization            !
!     .F(0,1/2;x),F(1,1/2;x) !   !     .F(1/2,1/2;x),F(3/2,1/2;x) !
! n>5 .calculate An and Bn   !   ! n>6 .calculate An and Bn       !
!     .F((n+1)/2,1/2;x)      !   !     .F((n+1/2,1/2;x)           !
 ----------------------------     --------------------------------
   !                                           !
   ------ ------------------------------------
                                !
                                v
                        .----------------------.
                        ! Calculate L and exp(L)!
                         ----------------------
                                !
                                v
             B                                        1-B
exp(L) < -----     !             Test            !   ----- > exp(L)
          1-a      !     B                 1-B   !    a
          .--------!   ----- < exp(L) < -----    !------.
          !        !    1-a                 a    !      !
          !         ------------------------------      !
          v                     ! otherwise             v
                                !
      ! ACCEPT !                v                  ! REJECT !
       --------             go to **                --------
          !                                             !
          ---------------> go to * <-----------------
```

$$L = \ln F\left(\frac{n-1}{2},\frac{1}{2};\frac{\delta^2 v^2}{2}\right) - \frac{n\delta^2}{2}$$

FIG.B.1

PROGRAM FOR THE SEQUENTIAL t-TEST
****

STEP	INSTRUCTION	INPUT	KEY	OUTPUT
1	Load card			
2	Input data	α	A	α
		β	R/S	β
		δ	R/S	δ
		rlast	R/S	rlast
		θ_o	R/S	θ_o
3	First 4 samples	xn	B	Ø
4	From 5th sample on	xn	B	exp(L)
5	Decision		D	Ø (continue) go to 4 1 (reject) go to 6 or 7 -1 (accept) go to 6 or 7
6	For a new case clear registers	Ø	STO Ø	Ø
			STO 1	Ø
			f P=S	Ø
			STO Ø	Ø
			STO 1	Ø
			STO 5	Ø
			f P=S	Ø
			h STI	Ø go to 2
7	End			

FIG.B.2

HP-67 IMPLEMENTATION OF THE StT ALGORITHM

001	f LBL A	036	g x^2	071	STO 1	106	g e^x
002	f P⇄S	037	STO+1	072	f LBL 4	107	×
003	STO 2	038	RCL 0	073	1	108	STO 7
004	R/S	039	RCL 1	074	RCL D	109	f LBL 3
005	STO 3	040	÷	075	+	110	4
006	f P⇄S	041	STO 5	076	STO D	111	h RC I
007	R/S	042	f P⇄S	077	RCL 0	112	−
008	STO C	043	RCL C	078	RCL 5	113	h RC I
009	R/S	044	g x^2	079	×	114	3
010	STO E	045	×	080	RCL D	115	−
011	R/S	046	2	081	÷	116	÷
012	STO B	047	÷	082	3	117	RCL 6
013	R/S	048	STO 5	083	RCL D	118	×
014	f LBL B	049	h RC I	084	2	119	RCL 5
015	STO A	050	5	085	×	120	h RC I
016	h RC I	051	g x≤y	086	−	121	+
017	1	052	GTO 0	087	×	122	2
018	+	053	GTO 1	088	RCL D	123	×
019	h ST I	054	f LBL 0	089	2	124	7
020	f P⇄S	055	h RC I	090	×	125	−
021	RCL 0	056	2	091	1	126	h RC I
022	f $\sqrt{x}$	057	÷	092	−	127	3
023	f P⇄S	058	g FRAC	093	÷	128	−
024	RCL A	059	f x=0	094	STO 0	129	÷
025	RCL B	060	GTO 2	095	STO+1	130	RCL 7
026	−	061	f LBL C	096	RCL D	131	STO 6
027	+	062	h RC I	097	RCL E	132	×
028	g x^2	063	5	098	g x=y	133	+
029	f P⇄S	064	g x≠y	099	GTO 5	134	STO 7
030	STO 0	065	GTO 3	100	GTO 4	135	STO 4
031	f P⇄S	066	1	101	f LBL 5	136	GTO C
032	RCL A	067	STO 0	102	RCL 1	137	f LBL 2
033	RCL B	068	STO 6	103	1	138	h RC I
034	−	069	0	104	+	139	6
035	f P⇄S	070	STO D	105	RCL 5	140	g x≠y

FIG.B.3

141	GTO6	176	+	211	−	
142	RCL5	177	STO9	212	f x<0	
143	g e^x	178	STO4	213	GTO8	
144	STO8	179	f LBL C	214	f LBL 1	
145	RCL5	180	RCL4	215	0	
146	2	181	f LN	216	R/S	
147	×	182	RCL C	217	f LBL 7	
148	1	183	g x^2	218	1	
149	+	184	h RCI	219	CHS	
150	×	185	×	220	R/S	
151	STO9	186	2	221	f LBL 8	
152	f LBL 6	187	÷	222	1	
153	RCL8	188	−	223	R/S	
154	4	189	g e^x	224	R/S	
155	h RCI	190	R/S	REGISTERS		
156	−	191	f LBL D	I	n	
157	h RCI	192	STO3	Ⓟ 0	V_{i-1}	
158	3	193	f P≷S	1	ΣV_i	
159	−	194	RCL3	3	e^L	
160	÷	195	1	4	$F(\frac{n-1}{2},\frac{1}{2};x)$	
161	×	196	RCL2	5	x	
162	RCL5	197	−	6	USED	
163	h RCI	198	÷	7	USED	
164	+	199	f P≷S	8	USED	
165	2	200	−	9	USED	
166	×	201	f x<0	A	X_n	
167	7	202	GTO7	B	θ_0	
168	−	203	f P≷S	C	δ	
169	h RCI	204	1	D	L	
170	3	205	RCL3	E	i_{LAST}	
171	−	206	−	Ⓢ 0	USED	
172	÷	207	RCL2	1	USED	
173	RCL9	208	÷	2	α	
174	STO8	209	f P≷S	3	β	
175	×	210	RCL3	5	σ^2	

EXAMPLE
****

Parameters = α = Ø.Ø5
β = Ø.Ø5
δ = .5(.5)2
rlast = 1Ø
θo = 13ØØ

sample number	sample value	exp(L)			
n	xn	δ=Ø.5	δ=1.Ø	δ=1.5	δ=2.5
1	1341				
2	1376				
3	1365				
4	1462				
5	1189	.866	.389	.Ø6485	.ØØ376
6	1329	.955	.418	.Ø5358	------
7	1298	.832	.24Ø	.Ø1511	ACCEPT
8	133Ø	.934	.259	------	
9	1257	.78Ø	.134	ACCEPT	
1Ø	1318	.888	.146		
11	1543	.741	.Ø78		
12	1211	.831	.Ø79		
13	1273	.689	.Ø42		
14	1439	.775	----		
15	1259	.635	ACCEPT		
16	128Ø	.713			

		CONTINUE			

Comments: * in this case, δmax = 3.Ø2.

* upper limit = $(1-\beta)/\alpha$ = 19.Ø
  lower limit = $\beta/(1-\alpha)$ = Ø.Ø5263

* above upper limit ---> reject hypothesis
  below lower limit ---> accept hypothesys
  between limits ---> continue sampling

FIG. B.4

APPENDIX C

SUBROUTINES USED IN THE STT ALGORITHM

In this Appendix efficient methods of calculating square roots, natural logarithms and exponentials will be shown. The idea is to implement non elemental functions with the four arithmetic operations encountered in every microprocessor.

For every algorithm it will be assumed that 10 iterations will be performed.

SQUARE ROOT

For software estimates, the square root is assumed to be calculated by HERON's iterative method where

$$\sqrt{x} = \frac{1}{2}\left(\frac{(xn)^2 + x}{(xn)}\right)$$

How fast it converges will depend on the initial xo.

The total number of operations is 20 multiplications, 10 add/substr., 10 divisions, 10 calls to memory and 10 comparisons.

The total storage (program + data) equals 98 bytes.

NATURAL LOGARITHM

This operation can be implemented in the following way:

$$\ln x = 2 \left[p + \frac{1}{3} p^3 + \frac{1}{5} p^5 + \ldots \right]$$

where

$$p = \frac{x - 1}{x + 1}$$

or, equivalently:

$$\ln x = 2 * p \left(1 + p^2 \left(\frac{1}{3} + p^2 \left(\frac{1}{5} + p^2 \left(\frac{1}{7} + \ldots \right)\right)\right)\right)$$

The total number of operations is 12 multiplications, 10 additions and 22 calls to memory.

The total storage is 100 bytes.

EXPONENTIAL

A continued fraction expansion has been chosen for its implementation and it can be written as

$$\exp(x) = \frac{1}{1-}\;\frac{x}{1+}\;\frac{x}{2-}\;\frac{x}{3+}\;\frac{x}{2-}\;\frac{x}{5+}\;\frac{x}{2-}\;\ldots$$

which can be understood as

$$\exp(x) = \cfrac{1}{1 - \cfrac{x}{1 + \cfrac{x}{2 - \cfrac{x}{3 + \cfrac{x}{2 - \cfrac{x}{5 + \cfrac{x}{2 - \ldots}}}}}}}$$

The total number of operations per iteration is 20 multiplications, 21 additions and 20 calls to memory.

Total storage equals 120 bytes.

Although some of these techniques have been implemented in floating point software packages or ROMs, they are generally slower than those implemented in fixed point special purpose subroutines. Here, it has been assumed that the latter method will be used.

Consequently, the total number of operations of these subroutines (executed once per iteration) is 52 multiplications, 41 add/substr., 10 divisions, 52 calls to memory and 10 comparisons. The total storage (ROM) is 318 bytes.

APPENDIX D

TIME AND MEMORY REQUIREMENTS

It is considered that every time a sample is taken an iteration is performed. FIG.D.1 shows the number of operations performed per iteration, including the operations performed by the subroutines in Appendix C. The last column indicates the instruction mix for this algorithm.

Considering that the algorithm will be implemented in a MOTOROLA MC68000 - based system the total time required is (4262.875 + 966.25 i) microseconds, where i is the number of iterations to be performed. Roughly, it can be considered that every iteration requires <u>one milisecond</u>.

The total storage is the sum of the main program (224 words in the HP-67 calculator = 448 bytes) and the corresponding storage for the subroutines introduced in Appendix B (318 bytes). The total storage becomes 766 bytes or, approximately, <u>.8 Kilobytes</u>.

OPERATION	SAMPLE NUMBER N					INSTRUCTION MIX (percentage)
	< 5	5		6	> 6	
		(1)	(2)			
ARITHMETIC						
Multipl.	24	57	4	49	61	26.1
Add/Subst.	13	62	3	41	53	22.6
Divisions	12	5	2	15	17	7.3
LOGIC						
Compare	11	3	2	13	14	6.0
CONTROL						
Store	5	9	2	10	9	3.8
Recall	20	78	8	51	76	32.5
Compare	1	1	1	3	3	1.3
Incr.& store	1	-	1	1	1	0.4

(1) This set of operations is performed once.
(2) This set of operations is performed r times (1 < r < rlast).

FIG.D.1

APPENDIX E

MARKOV MODELING

It is known that Markov models can be used to model with high fidelity and relative simplicity some characteristics of computer systems. In particular, it is useful to model the reconfiguration procedure that occurs automatically in the system proposed in Chapter 2.

A system is said to be governed by a continuous-time Markov model if the next state of the system depends only on its present state and is completely independent of the past history of the system states. In the case at hand the states are assumed to be discrete. For example, if the system to be modeled consists of N units (boards, processors, computers, links) one state will consist of all those units working correctly, ie, according to specifications. Another state will contain those situations in which (N-1) units work but one unit has failed. The general state will consist of (N-I) units working and I units failed. The last state before failure can be considered to have only one unit working. The previous states can be called 'operating states' because it is assumed that the functions performed by the working units are not interfered by the failure of the non-working units.

The operational states will be assumed to consist of units divided into two classes: working units and spare units. Operational units are all subject to the same failure rates because, even when a spare unit may not be performing the same funtions than the working units, they will continuosly being subject to tests to assure that they will be working properly when they are needed.

The 'failed state' considers the single case where all N units have failed. This failed state could mean that all the N units have simultaneously experienced an irrecoverable malfunction or that the remaining working units attempted a reconfiguration after the malfunction and the process failed to be completed because the malfunction actually inhibited the reconfiguration.

Any continuous-time discrete-states Markov model is completely specified by defining its states (N operating states and 1 failed state in the example given above), the probability densities of the transitions that may occur among the states and one boundary condition for each one of the states * (given that the model will be structured into a set of N + 1 differential equations).

Transitions among the operating states depend on the failure rate of the units, their repair rate (which depends on the feasibility of identifying the failed element and the easiness of change for a new one) and the coverage.

The previous concepts of states and transitions can be structured in a set of differential equations.

Assume that the probability of being in state i at time t is indicated by the symbol Pi(t). Then, the following expression, indicates that the probability of the system being in state i, at $t+\Delta t$, is equal to:

$$Pi(t+\Delta t) = pii.Pi(t) + \sum_{j=i} pij.Pj(t)$$

where pij = probability that the system state changes from state i to state j (assumed constant)

Defining the symbol aij as the transition rate from i to j

$$aij = \lim_{\Delta t \to 0} \left[\frac{pij}{\Delta t} \right]$$

and knowing that Σ pij = 1, the basic differential equation describing the behavior of the <u>state</u> is obtained:

$$\frac{dPi(t)}{dt} = - aii.Pi(t) + \sum_{j=i} aij.Pj(t)$$

From here, the set of equations describing the behavior of the <u>system</u> can be written in matrix notation as

---

* Actually, N states and boundary conditions will be enough for a N + 1 states model (as long as they are all specified at the same time t) because of the extra condition that the sum of the probabilities of being in every state at time t is equal to 1.

$$\frac{dP(t)}{dt} = A.P(t) \qquad (AE.1)$$

where $P(t)$ = vector of states $P1(t)$, $P2(t)$,..., $Pi(t)$,..., $PN(t)$ = dim(Nx1)

A = state transition (rate) matrix = dim(NxN)

subject to the initial conditions $P(0)$. The usual case is to specify these probabilities at time 0, when it is assumed that the system starts working with all of its units on, so that $P1(0)=1$ and the rest of the states are not occupied, due to the exclusion nature of the assumption.

The difference between the definitions of reliability and availability has been seen to be that the latter concept accepts external repairs from the failed state. According to a Markov model ther repair is modeled as a transition from the failed state to one of the operating states. To solve the set of differential equations, completely different approaches must be used. Reliability solutions can exhibit analytic solutions only for elemental systems. Availability solutions are normally found by numerical procedures. In references [BOUR71] and [NG76], some reliability solutions are displayed, along with more general problems and their solutions. Reference [BOUR71] mentions the use of the program REL70 to find the reliability of a 'typical' computer (processor and memory only). The program CARE, version III, cited in reference [STIF79], models a rather general computer, although its use is fairly complicated to use by the amount of detail it needs to model the system. Simplified, although non the less, still accurate general programs seem in order and in reference [NG76] the program ARIES is presented for that purpose.

Ultimately, as systems become complicated, integration routines are used, even for reliability models, because of roundoff errors and truncations that affect the results sensibly.

The solution of the availability model is made in this Thesis by integrating the set of equations (AE.1) with an efficient fourth-order Runge-Kutta integration procedure.

If the matrix A in (AE.1) is constant, because the transitions rates are time-invariant, the solution is

$$P(t) = \exp(A*t) * P(0)$$

An efficient procedure to find the exponential of a matrix involves the use of the Interpolation Method [ZADL63]. The exponential exp(A*t) is computed in the form of matrix coefficients times the natural modes exp(αi*t) so that

$$\exp(A) = \sum_{j=1}^{N} \frac{\mathop{}\limits_{\substack{i=1\\ i=k}}^{N} (A - \lambda i * I)}{\mathop{}\limits_{\substack{i=1\\ i=k}}^{N} (\lambda k - \lambda i)} \exp(\lambda k) \qquad (AE.2)$$

where λi is the ith eigenvalue of A.

This equation is used whenever the matrix A has simple eigenvalues. NG in [NG76] reports that in the reliability modeling of fault tolerant systems, the condition of having simple eigenvalues is met in all practical cases.

To model the reliability of the nodes, it will be assumed that, if there are J operational units in a node, they can be subject to three types of transitions, in which the following situations can be identified:

- a malfunction may occur, it will be detected by the units and the system will be reconfigured switching a spare unit on (if the failed unit was a working unit) or having one less spare (if a spare unit failed). Consequently, there will now be J-1 operational units in the node** and the transition will be denominated J--->J-1,
- the malfunction could be such that it can not be detected or that a reconfiguration is impossible. The final state will be the failed state. This transition can be labelled J--->0.
- a repair can be made, taking the system to the situation of having one more operational unit. This transition will be labelled J--->J+1.

These transitions, and the corresponding transition rates, are seen in FIG.E.1 and the corresponding system of

---

** It is assumed that, given the failure rates considered (typically 10^{3} to 10^{4} [failures/(hour*unit)], transitions J--->J-N with N>1 are negligible (ie that the probability that more than one unit fails in a differential dt is negligible).

differential equations is

$$\dot{P}1(t) = -N*\lambda*P1(t) + \rho P2(t)$$

$$\dot{P}2(t) = N*\lambda*c*P1(t) - ((N-1)*\lambda+\rho)*P2(t) + \rho*P3(t)$$

$$\vdots$$

$$\dot{P}I(t) = (N+2-I)*\lambda*c*PI-1(t)-((N+1-I)*\lambda+\rho)*PI(t)+\rho*PI+1(t)$$

$$\vdots$$

$$\dot{P}N(t) = 2*\lambda*c*PN-1(t) - (\lambda+\rho)*PN(t) \qquad \text{(AE.3)}$$

and

$$\dot{P}0(t) = -\sum_{I=1}^{N} \dot{P}I(t)$$

subject to the initial condition P1(0)=1.0 and Pi(0)=0.0 for all i=j. The reliability of the system is, by definition, the sum of the probabilities of being in the operational states (note that a subset of the N number of states 1, 2,...,N can also be defined as a valid set; in this situation, the system will be less reliable than considering the totallity of N states). Then, the reliability will be

$$\text{Reliability } (t) = R(t) = \sum_{I=1}^{N} PI(t)$$

The program RELNODE has been used in this Thesis to calculate the reliability of the nodes and to plot the results (see FIG.E.2 and subsequent figures for an output of the program and the plot of the results). The source program is included in Appendix G. The reliability is found with the help of equation (AE.2) to obtain the exponential of the matrix A*t. The eigenvalues are found with the QR algorithm (see references [FRAN62] and [WICK65]).

With respect to the availability modeling the only difference with the modeling of reliability is the transition from the failed state to an operating state. Although a transition to any operating state is possible (ie the final state of the transition may be a node having any combination of working and spare units), it is found that the availability will increase if the terminal state has as many working units as possible (ie to the configuration that the node presented at time 0). Thus, it will be assumed that this transition takes the node from the failed state to the state that has more units working. This situation can be seen in FIG.E.4. Taking this transition in consideration, the system of differential equations used to evaluate the availability of the node will a modification of the system of differential equations (AE.3), where only the first and last equations of the set will be modified (corresponding to the time derivatives of P1 and P0, respectively).

The modifications will be the following

$$\dot{P1}(t) = -N*\lambda*P1(t) + \rho*P2(t) + \rho2*P0(t) \qquad \text{(AE.4)}$$

and

$$\dot{P0}(t) = -\sum_{I=1}^{N} \dot{PI}(t) - \rho2*P0(t) \qquad \text{(AE.5)}$$

where $\rho2$ = repair rate to take the node from the failed state to the operational state 1, in [repairs/hour].

From its definition,availability can be calculated by adding the probabilities of being in the operational states, which is

$$\text{Availability (t)} = A(t) = \sum_{I=1}^{N} PI(t)$$

The system of differential equations (AE.3) with the modifications (AE.4) and (AE.5) was solved with a numerical procedure: a fourth-order RUNGE-KUTTA algorithm, used in the program DYSYS, available at the Joint Computer Facility at MIT. The subroutine used to write down the differential equations (called EQSIM) is included in Appendix G. In this Appendix, the availability of the node considered above (in FIG.E.2 and FIG.E.3) is calculated. The input data file used is shown in FIG.E.5 and the outputs from DYSYS, FIG.E.6 and the plot in FIG.E.7, are included.

BIBLIOGRAPHY FOR APPENDIX E

[FRAN62] FRANCIS, J., The QR transformation, The computer journal, October 1961 and January 1962

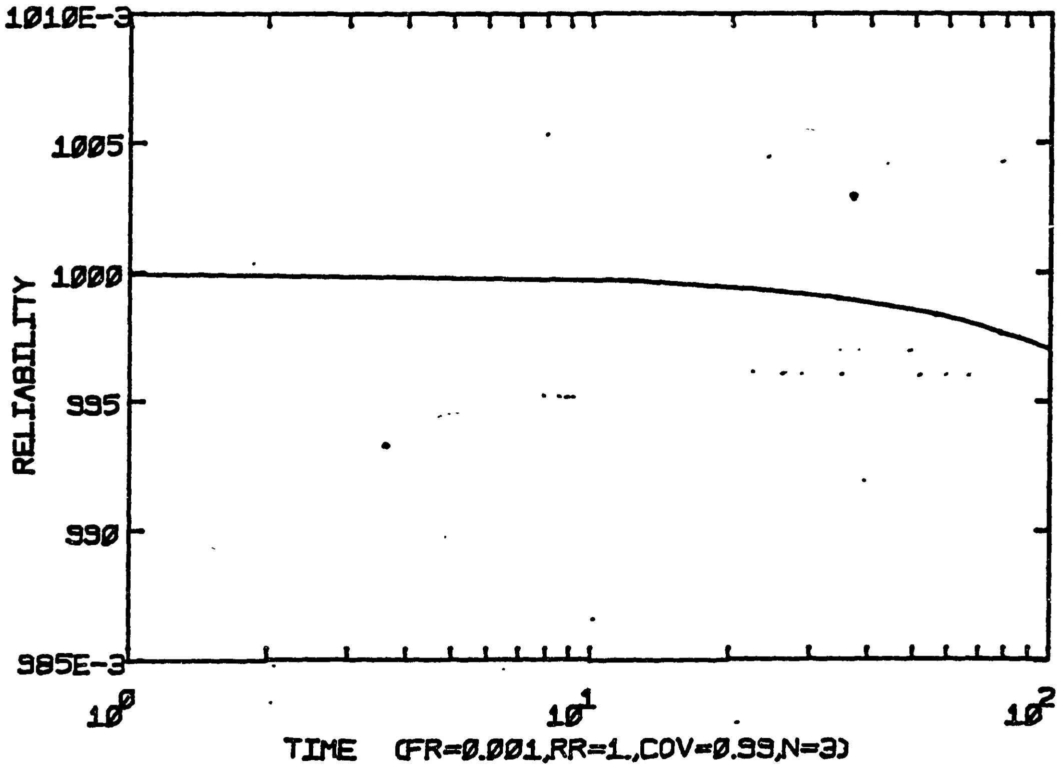
[NG76] NG, Y., Reliability modeling and analysis for fault-tolerant computers, UCLA-ENG-7698, September 1976

[STIF79] STIFFLER, J. et al, Computer aided reliability estimation CARE III final report, NASA-CR-159122, November 1979

[WILK62] WILKINSON, J., The algebraic eigenvalue problem, Clarendon Press, Oxford, 1965

```
OUTPUT FROM THE PROGRAM RELNODE
********************************


Comments: in this case, we are interested in
          the reliability of a node with 3
units. The reliability will be calculated
for 4 days (100 hours). The rest of the
parameters are typical of a better-than-standard
computer system.


INPUTS
******

    FR  =      0.001  FAILURES/HOUR
    RR  =      1.0    REPAIRS/HOUR
    COV =      0.99
    N   =      3      UNITS

OUTPUTS
*******

         MTFF = 3342.6832 HOURS
         xxxx

         EFR =  0.000029916087 FAILURES/HOUR
         xxx

    Comment: the failure rate of the 3 units
             (EFR) is 30 times smaller than the
             failure rate of one on them (FR).


                 RELIABILITY
                 xxxxxxxxxxx

        Time                  Reliability

    0.10000000E+01        0.99996996E+00
    0.12000000E+02        0.99964082E+00
    0.23000000E+02        0.99931192E+00
    0.34000000E+02        0.99898314E+00
    0.45000000E+02        0.99865443E+00
    0.56000000E+02        0.99832588E+00
    0.67000000E+02        0.99799740E+00
    0.78000000E+02        0.99766904E+00
    0.89000000E+02        0.99734080E+00
    0.10000000E+03        0.99701262E+00
```

FIG.E.2



NODE RELIABILITY
****

FIG. E.3

AVAILABILITY MODELING

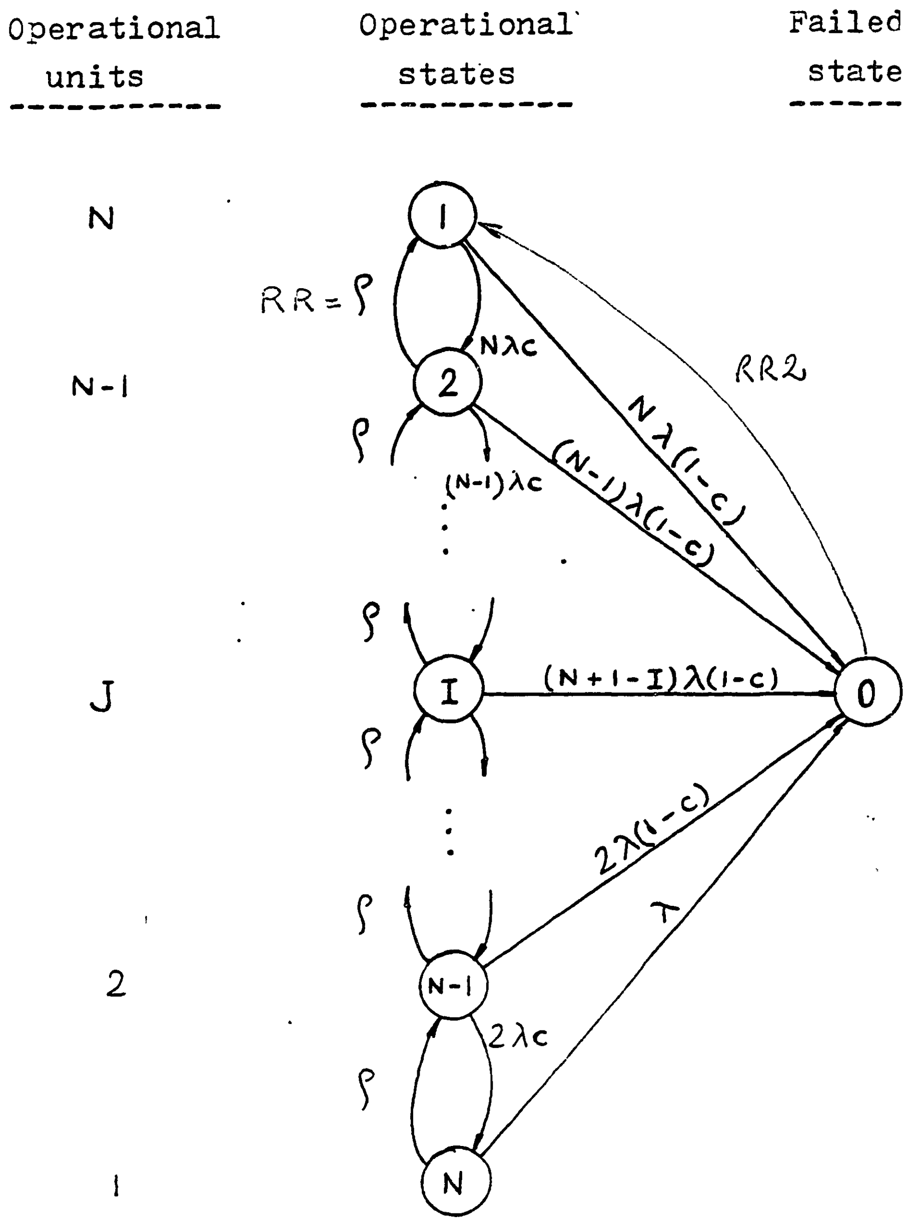
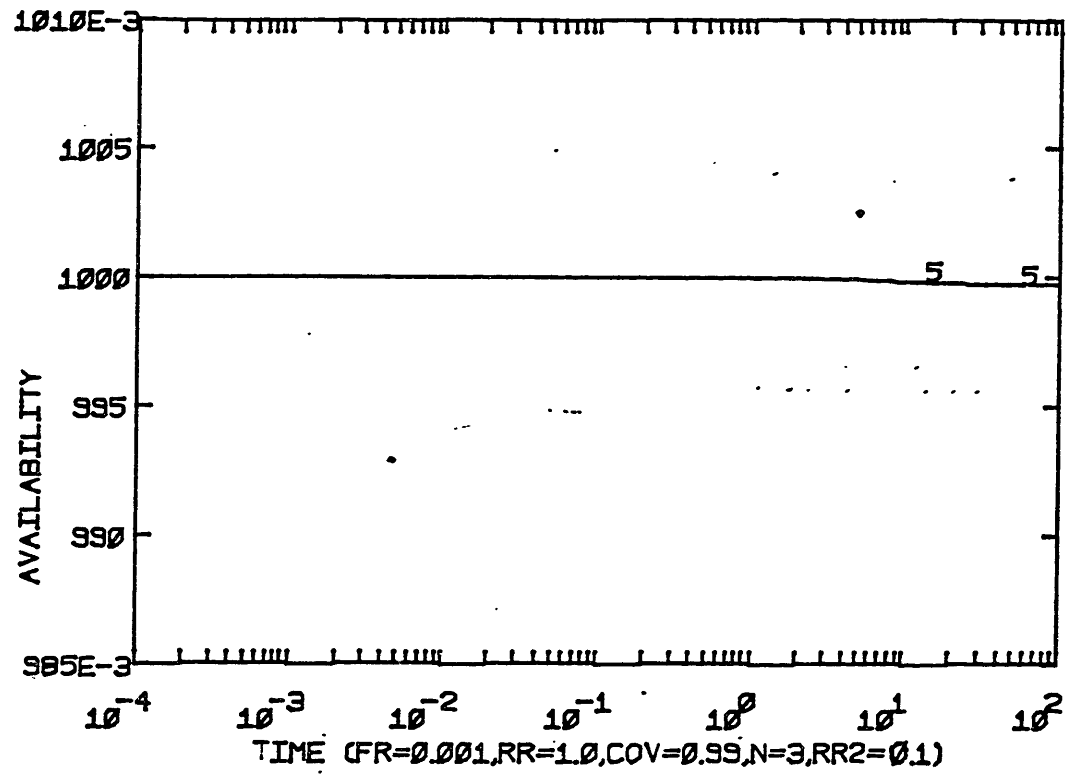
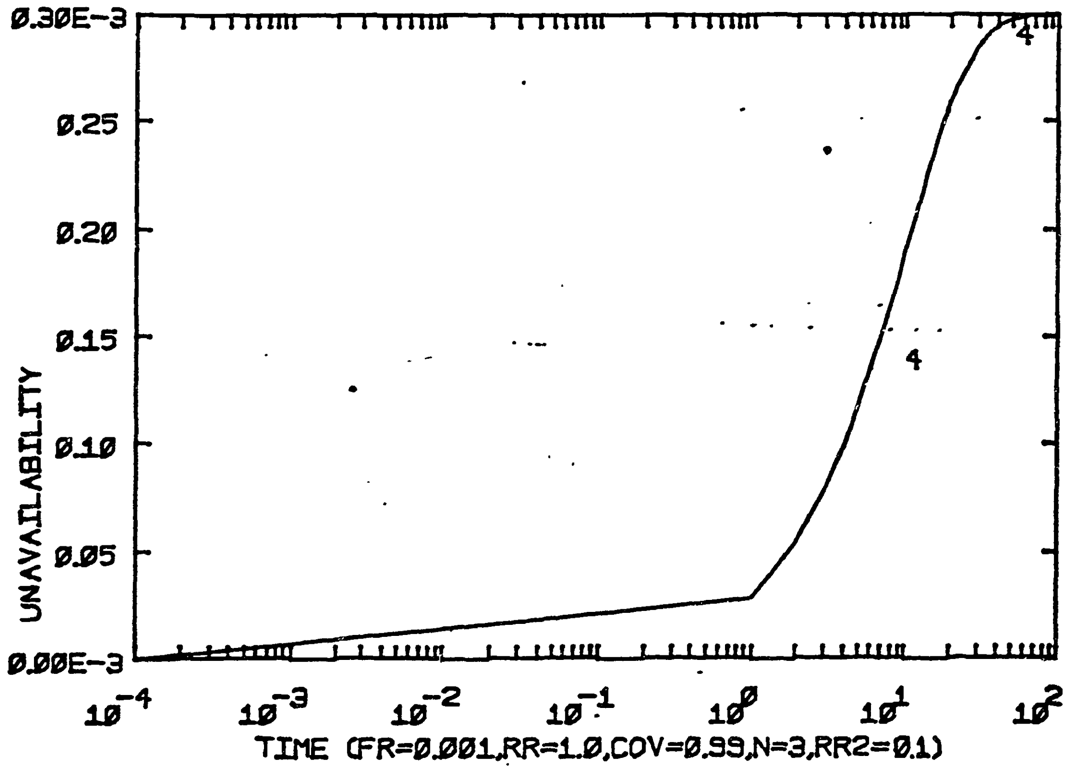


FIG. E.4



NODE AVAILABILITY
****

FIG.E.7



NODE UNAVAILABILITY
****

FIG.E.8

APPENDIX F

THE NETWORK SURVIVABILITY PROBLEM

The basis of the procedure to solve the network reliability and availability problem is to use a recursive algorithm presented and mentioned in Chapter 6. The details of the algorithm can be seen in the original paper.

The algorithm divides the network in smaller networks and, to calculate the probability the nodes are disconnected in the network, finds the probability the nodes are disconnected in the subnetworks. The procedure considers the nodes and the links separate entities and assumes that their reliability parameters (failure rates and repair rates) are independent.

If a node is failed all paths through that node are failed. The approach by which the reliability measures: network reliability and network availability can be calculated is to enumerate all 2**N possible combinations of working and failed nodes, where N is the number of nodes.

Let s(T) be the probability that subset T of the nodes are working and subset N-T of the nodes are failed and let E(T) be the value of the reliability measure in the network on the nodes in T (those nodes that are working). Then, E, the overall reliability measure, is given by

$$E = \sum_{T} S(T) * E(T)$$

where the summation is over all subsets T of N.

The term S(T) will not take into account the existence of links between nodes and will be concentrated on the reliability (availability) of the nodes.

On the other hand, E(T) will assume that the nodes are perfect and will only take into account the failure of the links.

©&Calculation of S(T)®&

The general combinatorial expression for the probability that subset T of the nodes are working S(T) is

$$S(T) = rc(t) * \binom{NN}{T-1} * (r(t)**(T-1))*((1-r(t))**(NN-T+1))+$$

$$+(1-rc(t))* \binom{NN}{T-1} *(r(t)**T)*((1-r(t))**(NN-T))$$

where rc(t)	=	reliability (availability) of the central computer at (up to) time t.
r(t)	=	reliability (availability) of the nodes at (up to) time t, as calculated by program RELNODE (AVALNODE).
NN	=	number of nodes in the network including the central computer.
T	=	variable that indexes the subnetworks: T = 1, 2, ..., NN-1.

However, for the purposes of being conservative, it will be stated that, if a single node fails, the network will be considered to fail. Then, the above expression is reduced to

$$S(T) = rc(t)*(r(t)**(NN-1))$$

The assumptions implied in this equation are a) at time 0, all nodes of the network are working with probability one and b) all nodes are identical.

©&Calculation of E(T)®&

The value of the reliability measure in the fully connected network with T perfect nodes is E(T) (see reference [BUZA80] in Chapter 6) which may be expressed as:

$$E(T) = 1 - \sum_{IU-1}^{T-1} *E(IU)*((1-rl(t))**(IU*(T-IU)))$$

with E(1) = 1.0 and

where rl(t) = reliability (availability) of a single link at (up to) time t.

The reliability of the links can be calculated by considering that the process can be modeled as a single unit subject to a constant failure rate FRL [failures/hour] or

$$rl(t) = \exp(- FRL * t)$$

The availability of the links will be calculated by considering that the process can be modeled by a two state Markov process in which one of the states is working and the other is failed. That can be written as

$$\dot{P1}(t) = - FRL * P1(t) + RRL * P2(t)$$

$$\dot{P2}(T) = FRL * P1(t) - RRL * P2(t)$$

where P1(t) = probability that the link is working at time t

P2(t) = probability that the link is failed at time t

RRL = repair rate of the link [repairs/hour]

and P1(0) = 1.0

The solution to this system of two differential equations is rl(t), the term used in the equation that finds E(T). Thus,

$$rl(t) = \frac{1}{FRL + RRL} (FRL * \exp(-(FRL+RRL)*t) + RRL)$$

The availability of the central computer may also be calculated using the same approach that is employed with the links. Thus,

$$rc(t) = \frac{1}{FRC + RRC} (FRC * \exp(-(FRC+RRC)*t) + RRC)$$

where the new terms FRC and RRC are the failure rate and the repair rate of the central computer, respectively.

The steady state availabilities

$$\frac{\text{repair rate}}{\text{failure rate + repair rate}}$$

are displayed for different combinations of failure rates and repair rates in FIG.F.1.

STEADY STATE AVAILABILITIES
****

Repair rate [repairs/hour]	Failure rate [failures/hour]			
	0.00001	0.0001	0.001	0.01
0.01	0.999	0.9901	0.909	0.5
0.04167	0.99976	0.997606	0.9766	0.8065
0.1	0.9999	0.999001	0.9901	0.909
1.0	0.99999	0.9999	0.999001	0.9901

FIG.F.1

BLANK PAGE

APPENDIX G

PROGRAMS USED IN THE THESIS

The programs that follow were used in this Thesis. The necessary inputs and the outputs are indicated in the text of the program. The use of double precision is recommended for most of the variables.

Where necessary, the programs were linked with the libraries PENPLOT and SSP (Scientific Subroutine Package) installed at the Joint Computer Facility at MIT.

```
      PROGRAM RELNODE
C     ***************
C
C         This  program  can be  used  to  calculate   the
C     reliability  of a system  described by the Markovian
C     model presented in Chapter 6 of this Thesis.
C
C         The program is interactive and instructions  are
C     given during execution.
C
C     OPTIONS:   Exit                  ---> 0
C                Proceed               ---> 1
C                 One point in time    ---> 1
C                 Many points in time ---> >1 (Note)
C                 No plot              ---> 0
C                 Plot                 ---> 1
C
C     INPUTS:    FR: Failure  rate of the individual units
C                    [failures/hour] (Real)
C                RR: Repair rate for  individual  units in
C                    [repairs/hour]  (Real)
C                C : Coverage (Real)
C                N : Number  of  operational  units in the
C                    node (Integer)
C
C     OUTPUTS:   MTFF : Mean time to first failure [hours]
C                EFR  : Effective failure rate of the node
C                       [1/hour]
C                RT   : Reliability of the node
C     (Note): the last  point  in  time and the  number of
C             decades up to that point must be specified.
C
      DIMENSION B(10,10),BD(10,10),E(10),EI(10),EBT(10)
      DIMENSION IANA(10),Y(10,100),XSCL(4)
      COMMON C(10,10,10),CD(10,10,10)
      REAL MTFF
      INTEGER DEC,OPT,P
      CHARACTER*40 CURVLAB,XLAB,YLAB
10    WRITE(6,39)
39    FORMAT(/,' ENTER OPTION',/)
      WRITE(6,*)' EXIT--->0'
      WRITE(6,*)' PROCEED--->1'
      READ(5,13)OPT
13    FORMAT(I3)
      WRITE(6,37)OPT
37    FORMAT (5X,//,' OPTION = ',I3,//)
      IF(OPT.EQ.0)GO TO 1000
      WRITE(6,*)' ENTER FR,RR,COV,N'
      READ(5,11)FR,RR,COV,N
11    FORMAT(3E15.5,I3)
      WRITE(6,38)FR,RR,COV,N
```

```
 38     FORMAT(5X,' FR  = ',E15.5,' FAILURES/HOUR',/,
     1         5X,' RR  = ',E15.5,' REPAIRS/HOUR',/,
     1         5X,' COV = ',E15.5,/,
     1         5X,' N   = ',12X,I3,' UNITS',//)
        WRITE(6,*)' ENTER NPTS'
        READ(5,14)NPTS
 14     FORMAT(I4)
        IF(NPTS.EQ.1)GO TO 15
        WRITE(6,*)' ENTER TMAX,DEC'
        READ(5,17)TMAX,DEC
 17     FORMAT(E15.5,I3)
        TMIN=TMAX/(10**DEC)
        DELT=(TMAX-TMIN)/(NPTS-1)
        TIME=TMIN
        GO TO 18
 15     WRITE(6,*)' ENTER TIME'
        READ(5,16)TIME
 16     FORMAT(E15.5)
 18     DO 12 I=1,N
        DO 12 J=1,N
          B(I,J)=0.0
 12       BD(I,J)=0.0
        DO 1 I=1,N-1
          BD(I,I)=-(N-I+1)*FR-RR
          BD(I+1,I)=(N-I+1)*FR*COV
 1        BD(I,I+1)=RR
        BD(1,1)=-N*FR
        BD(N,N)=-FR-RR
        DO 2 K=1,N
        DO 2 L=1,N
 2        B(K,L)=BD(K,L)
C
C       EIGENVALUES (QR ALGORITHM)
C
        CALL HSBG(N,BD,10)
        CALL ATEIG(N,BD,E,EI,IANA,10)
        DO 3 I=1,N
        DO 3 J=1,N
        DO 3 K=1,N
          CD(I,J,K)=0.0
 3        C(I,J,K)=0.0
        CALL EXBT(N,B,E)
C
C
C       MEAN TIME TO FIRST FAILURE
C
C
        MTFF=0.0
        DO 9 I=1,N
          SUMMTF=0.0
        DO 8 J=1,N
 8        SUMMTF=SUMMTF+C(I,J,1)
 9      MTFF=MTFF-SUMMTF/E(I)
```

```
C
C
C       EFFECTIVE FAILURE RATE
C
C
        EFR=1./MTFF
C
C
C       RELIABILITY
C
C
        DO 19 M=1,NPTS
          IF(M.EQ.1)GO TO 20
          TIME=TIME+DELT
20        DO 21 I=1,N
          DO 21 J=1,N
          DO 21 K=1,N
21          CD(I,J,K)=C(I,J,K)
          DO 4 I=1,N
          DO 4 J=1,N
4           CD(I,J,1)=CD(I,J,1)*EXP(E(I)*TIME)
          DO 5 J=1,N
5           EBT(J)=0.0
          DO 6 I=1,N
          DO 6 J=1,N
6           EBT(J)=EBT(J)+CD(I,J,1)
          SUMEBT=0.0
          DO 7 J=1,N
7           SUMEBT=SUMEBT+EBT(J)
          Y(1,M)=TIME
          Y(2,M)=SUMEBT
          IF(M.NE.1)GO TO 22
C
C         OUTPUTS
C
          WRITE(6,23)MTFF,EFR
23        FORMAT(//,10X,' MTFF =',E15.8,' HOURS',/,11X,'xx
     1    xx',//,10X,' EFR = ',E15.8,' 1/HOUR',/,11X,'xxx',
     1    //)
          WRITE(6,*)'                    RELIABILITY'
          WRITE(6,*)'                    xxxxxxxxxxx'
          WRITE(6,*)'         Time                   Reliability'
22        WRITE(6,24)Y(1,M),Y(2,M)
24        FORMAT(4X,E15.8,6X,E15.8)
C
C
C         PLOTTING
C
C
C         IF(NPTS.EQ.1)GO TO 19
          IF(M.NE.NPTS)GO TO 19
          WRITE(6,41)
41        FORMAT(///,' TO PLOT ENTER 1',/,'
     1            0 OTHERWISE',/)
        READ(5,25)P
 25     FORMAT(I2,/)
        IF(P.EQ.0)GO TO 19
        WRITE(6,28)
 28     FORMAT(/,' ENTER MOVE',//,
     1  '       FIRST AND LAST      ->      0',/,
     1  '       FIRST, NOT LAST     ->      1',/,
     1  '       NOT FIRST BUT LAST  ->     10',/,
     1  '       NOT FIRST, NOT LAST ->     11')
        READ(5,31)MOVE
 31     FORMAT(I3)
        IF(MOVE.GE.10)GO TO 26
        WRITE(6,*)' ENTER XLAB'
        READ (5,29)XLAB
 29     FORMAT(A40)
 26     IF(MOVE.EQ.01.OR.MOVE.EQ.11)GO TO 27
        WRITE(6,*)' ENTER YLAB'
        READ (5,29)YLAB
 27     IF(MOVE.NE.00)GO TO 36
        LABEL=4
        ISCL=12
        GO TO 32
 36     IF(MOVE.NE.01)GO TO 30
        LABEL=4
        ISCL=12
        GO TO 32
 30     IF(MOVE.NE.10)GO TO 35
        LABEL=4
        ISCL=8
        GO TO 32
 35     LABEL=4
        ISCL=8
 32     WRITE(6,*)' ENTER CURVLAB (Two spaces first)'
        READ(5,33)CURVLAB
 33     FORMAT(A4)
        CALL QPICTR(Y,10,NPTS,QY(2),QX(1),QMOVE(MOVE),
     1    QISCL(ISCL),QCURVLAB(CURVLAB),QXLAB(XLAB),
     1    QYLAB(YLAB),QXSCL(XSCL),QLABEL(LABEL))
 19   CONTINUE
      GO TO 10
 1000 STOP
      END

      SUBROUTINE EXBT(N,B,E)
      COMMON C(10,10,10)
      DIMENSION B(10,10),E(10),EIK(10),B1(10,10)
      DIMENSION B2(10,10),T(10,10)
      DO 600 I=1,N
      DO 15 K=1,N
 15   EIK(K)=E(I)-E(K)
      DO 500 J=1,N
      IF(J-I)100,500,200
```

```
100       IF(J-1)110,110,150
200       IF(I-1)300,300,400
300       IF(J-I-1)110,110,150
400       IF(J-I-1)110,150,150
110       DO 5 K=1,N
          DO 5 L=1,N
5         B1(K,L)=B(K,L)
          DO 20 K=1,N
          B1(K,K)=B(K,K)-E(J)
          DO 20 L=1,N
20        B1(K,L)=B1(K,L)/EIK(J)
          GO TO 500
150       DO 40 K=1,N
          DO 40 L=1,N
40        B2(K,L)=B(K,L)
          DO 25 K=1,N
          B2(K,K)=B(K,K)-E(J)
          DO 25 L=1,N
25        B2(K,L)=B2(K,L)/EIK(J)
          DO 30 K=1,N
          DO 30 L=1,N
          T(K,L)=0.0
          DO 30 M=1,N
30        T(K,L)=T(K,L)+B1(K,M)*B2(M,L)
          DO 35 K=1,N
          DO 35 L=1,N
35        B1(K,L)=T(K,L)
500       CONTINUE
          DO 60 K=1,N
          DO 60 L=1,N
60        C(I,K,L)=B1(K,L)
600       CONTINUE
          RETURN
          END
```

```
C        PROGRAM AVALNODE
C        ****************
C
C
         SUBROUTINE EQSIM
C
C        This program is a subroutine of the program DYSYS,
C        used to calculate the availability of a node
C        described by the set of differential equations
C        mentioned in the text (Chapter 6 and Appendix E).
C
C        RR2 : Repair rate of one repairman, to take the
C              failed node from the failed state to an
C              operational state in [repairs/hour].
C
C        The rest of the parameters are defined as in the
C        program RELNODE.
C
         COMMON T,DT,Y(30),F(30),STIME,FTIME,NEWDT,NEWRUN,N,
     1         IPR,ICD,ICN,TBREAK,PNEXT,TBACK
C
C        INPUTS
C
         IF(NEWRUN.EQ.-1)THEN
         TYPE 10
10       FORMAT('ENTER FR,RR,COV,RR2')
         ACCEPT * ,FR,RR,COV,RR2
         END IF
C
C        The system of differential equations is:
C
         F(1)=-3.*FR*Y(1)+RR*Y(2)+RR2*Y(4)
         F(2)=3*FR*COV*Y(1)-(2.*FR+RR)*Y(2)+RR*Y(3)
         F(3)=2.*FR*COV*Y(2)-(FR+RR)*Y(3)
         F(4)=-(F(1)+F(2)+F(3))
         Y(5)=Y(1)+Y(2)+Y(3)
C
C        where (1), (2) and (3) are operational states
C        and (0) is the failed state.
C        F(i) indicates the time derivative of Y(i).
C        Y(5) models the availability, while Y(4) calculates
C        the unavailability.
C
         RETURN
         END
```

```
C        PROGRAM AVALCOMP
C        ****************
C
C
C
         SUBROUTINE EQSIM
C
C        This program can be used to find the reliability of
C        a node with 3 working  units and 0, 1 or 2  spares.
C        Thus, the total number of operational units becomes
C        3, 4 or 5.
C        The term 'degradation' implies that,  if  the  node
C        is allowed to work with less  than  three units  in
C        parallel, a degradation of the functions  performed
C        by the node may occur, although not necessarily.
C
         COMMON T,DT,Y(98),F(98),STIME,FTIME,NEWDT,NEWRUN,N,
     1          IPR,ICD,ICN,TBREAK,PNEXT,TBACK
         IF(NEWRUN.EQ.-1)THEN
         TYPE 10
10       FORMAT('ENTER FR,RR,COV')
         ACCEPT * ,FR,RR,COV
         END IF
C        FIRST: ONE SPARE W/O DEGRADATION
         F(1)=-4.*FR*Y(1)+RR*Y(2)
         F(2)=4.*FR*COV*Y(1)-(3.*FR+RR)*Y(2)
C        SECOND: ONE SPARE W/DEGRADATION
         F(3)=-4.*FR*Y(3)+RR*Y(4)
         F(4)=4.*FR*COV*Y(3)-(3.*FR+RR)*Y(4)+RR*Y(5)
         F(5)=3.*FR*COV*Y(4)-(2.*FR+RR)*Y(5)+RR*Y(6)
         F(6)=2.*FR*COV*Y(5)-(FR+RR)*Y(6)
C        THIRD: TWO SPARES W/O DEGRADATION
         F(7)=-5.*FR*Y(7)+RR*Y(8)
         F(8)=5.*FR*COV*Y(7)-(4.*FR+RR)*Y(8)+RR*Y(9)
         F(9)=4.*FR*COV*Y(8)-(3.*FR+RR)*Y(9)
C        FOURTH: TWO SPARES W/DEGRADATION
         F(10)=-5.*FR*Y(10)+RR*Y(11)
         F(11)=5.*FR*COV*Y(10)-(4.*FR+RR)*Y(11)+RR*Y(12)
         F(12)=4.*FR*COV*Y(11)-(3.*FR+RR)*Y(12)+RR*Y(13)
         F(13)=3.*FR*COV*Y(12)-(2.*FR+RR)*Y(13)+RR*Y(14)
         F(14)=2.*FR*COV*Y(13)-(FR+RR)*Y(14)
C        FIFTH: NO SPARES W/O DEGRADATION
         F(15)=-3.*FR*Y(15)
C        SIXTH: NO SPARES W/DEGRADATION
         F(16)=-3.*FR*Y(16)+RR*Y(17)
         F(17)=3*FR*COV*Y(16)-(2.*FR+RR)*Y(17)+RR*Y(18)
         F(18)=2.*FR*COV*Y(17)-(FR+RR)*Y(18)
C        (1) RELIABILITY OF 3P+1S W/O D
         Y(19)=Y(1)+Y(2)
C        (2) RELIABILITY OF 3P+1S W/D
         Y(22)=Y(3)+Y(4)+Y(5)+Y(6)
C        (3) RELIABILITY OF 3P+2S W/O D
```

```
      Y(20)=Y(7)+Y(8)+Y(9)
C     (4) RELIABILITY OF 3P+2S W/D
      Y(23)=Y(10)+Y(11)+Y(12)+Y(13)+Y(14)
C     (5) RELIABILITY OF 3P W/O D
      Y(15)=Y(15)
C     (6) RELIABILITY OF 3P W/D
      Y(21)=Y(16)+Y(17)+Y(18)
C     IMPROVEMENTS
      IF(Y(15).LE.1.E-24)GO TO 2
      Y(24)=Y(21)/Y(15)
      Y(27)=Y(19)/Y(15)
      Y(29)=Y(27)/Y(15)
2     IF(Y(19).LE.1.E-24)GO TO 3
      Y(25)=Y(22)/Y(19)
      Y(28)=Y(20)/Y(19)
3     IF(Y(20).LE.1.E-24)GO TO 4
      Y(26)=Y(23)/Y(20)
4     IF(Y(21).LE.1.E-24)GO TO 5
      Y(30)=Y(22)/Y(21)
      Y(32)=Y(23)/Y(21)
5     IF(Y(22).LE.1.E-24)GO TO 6
      Y(31)=Y(23)/Y(22)
6     RETURN
      END
      PROGRAM RELNET
C     **************
C
C         This program calculates the reliability of a
C     network with unreliable nodes and links. The
C     program is interactive and instructions are given
C     as the computations proceed.
C         Some of the parameters have been defined in
C     the program RELNODE. New parameters are defined
C     below.
C
C     CENTRAL COMPUTER: RCC = Reliability of the central
C             computer. Assume the minimun value it may
C             exhibit in the interval of interest.
C
C     NET: NN = Total number of nodes in the network not
C              considering the central node.
C          FRL = Failure rate of the links [failures/hour]
C
C
      DIMENSION B(10,10),BD(10,10),E(10),EI(10),EBT(10)
      DIMENSION IANA(10),P(10),S(1),Y(10,100),XSCL(4)
      COMMON C(10,10,10),CD(10,10,10)
      REAL MTF
      INTEGER DEC,OPT,PLOT,TT
      CHARACTER*40 CURVLAB,XLAB,YLAB
10    WRITE(6,*)' ENTER OPTION'
      WRITE(6,*)'       0      -->    EXIT  '
      WRITE(6,*)'  OTHERWISE --> CONTINUE'
      READ(5,13)OPT
13    FORMAT(I3)
      IF(OPT.EQ.0)GO TO 1000
      WRITE(6,*)' NODES: ENTER NU,FR,RR,COV'
      READ(5,11)N,FR,RR,COV
11    FORMAT(I3,3E15.5)
      WRITE(6,*)' CENTRAL NODE: ENTER RFTMP'
      READ(5,38)RFTMP
38    FORMAT(E15.7)
      WRITE(6,*)' NET: ENTER NN,FRL'
      READ(5,37)NN,FRL
37    FORMAT(I3,E15.7)
      WRITE(6,*)' ENTER NPTS'
      READ(5,14)NPTS
14    FORMAT(I4)
      IF(NPTS.EQ.1)GO TO 15
      WRITE(6,*)' ENTER TMAX,DEC'
      READ(5,17)TMAX,DEC
17    FORMAT(E15.5,I3)
      TMIN=TMAX/(10**DEC)
      DELT=(TMAX-TMIN)/(NPTS-1)
      TIME=TMIN
      GO TO 18
15    WRITE(6,*)' ENTER TIME'
      READ(5,16)TIME
16    FORMAT(E15.5)
18    DO 12 I=1,N
      DO 12 J=1,N
        B(I,J)=0.0
12      BD(I,J)=0.0
      DO 1 I=1,N-1
        BD(I,I)=-(N-I+1)*FR-RR
        BD(I+1,I)=(N-I+1)*FR*COV
1       BD(I,I+1)=RR
      BD(1,1)=-N*FR
      BD(N,N)=-FR-RR
      DO 2 K=1,N
      DO 2 L=1,N
2       B(K,L)=BD(K,L)
      CALL HSBG(N,BD,10)
      CALL ATEIG(N,BD,E,EI,IANA,10)
      DO 3 I=1,N
      DO 3 J=1,N
      DO 3 K=1,N
        CD(I,J,K)=0.0
3       C(I,J,K)=0.0
      CALL EXBT(N,B,E)
C
C
C     MEAN TIME TO FIRST FAILURE
C
C
      MTF=0.0
      DO 9 I=1,N
        SUMMTF=0.0
      DO 8 J=1,N
8       SUMMTF=SUMMTF+C(I,J,1)
9     MTF=MTF-SUMMTF/E(I)
C
C
C     RELIABILITY
C
C
      DO 19 M=1,NPTS
        IF(M.EQ.1)GO TO 20
        TIME=TIME+DELT
20      DO 21 I=1,N
        DO 21 J=1,N
        DO 21 K=1,N
21        CD(I,J,K)=C(I,J,K)
        DO 4 I=1,N
        DO 4 J=1,N
4         CD(I,J,1)=CD(I,J,1)*EXP(E(I)*TIME)
        DO 5 J=1,N
5         EBT(J)=0.0
```

```
          DO 6 I=1,N
          DO 6 J=1,N
6           EBT(J)=EBT(J)+CD(I,J,1)
          SUMEBT=0.0
          DO 7 J=1,N
7           SUMEBT=SUMEBT+EBT(J)
          RENODE=SUMEBT
C
C         PROBABILITY THAT ALL OF THE NODES ARE WORKING
C
          S(NN)=RFTMP*RENODE**(NN-1)
C
C         RELIABILITY OF THE NET WITH RELIABLE NODES
C
          P(1)=1.0
          DO 40 TT=2,NN
            AUX=0.0
            DO 40 IU=1,TT-1
              IF(RENODE.NE.1.0)THEN
                RL=EXP(-FRL*TIME)
                AUX=AUX+COM(TT-1,IU-1)*P(IU)*
     1              (1.0-RL)**(IU*(TT-IU))
              ELSE
                AUX=0.0
              ENDIF
            P(TT)=1.0-AUX
40        CONTINUE
C
C         RELIABILITY OF THE NET WITH UNRELIABLE NODES
C
          RENET=S(NN)*P(NN)
          Y(1,M)=TIME
          Y(2,M)=RENODE
          Y(3,M)=RENET
          IF(M.NE.1)GO TO 22
          WRITE(6,23)MTF
23        FORMAT(30X,' MTFF =',E15.8,/,30X,'xxxx',//)
          WRITE(6,44)
44        FORMAT(37X,'RELIABILITY',/,
     1           37X,'xxxxxxxxxxx',//,
     1           20X,'TIME',
     1           10X,'NODE RELIABILITY',
     1            9X,'NET RELIABILITY',/)
22        WRITE(6,24)Y(1,M),Y(2,M),Y(3,M)
24        FORMAT(13X,E16.8,4X,E16.8,10X,E15.8)
C
C
C         PLOTTING
C
C
C         IF(NPTS.EQ.1)GO TO 19
          IF(M.NE.NPTS)GO TO 19
          WRITE(6,*)' TO PLOT ENTER 1'
```

```
            WRITE(6,*)'      0 OTHERWISE'
            READ(5,25)PLOT
 25         FORMAT(I2)
            IF(PLOT.EQ.0)GO TO 19
            WRITE(6,28)
 28         FORMAT(' ENTER MOVE',//,
      1     '        FIRST AND LAST       ->      0',/,
      1     '        FIRST, NOT LAST      ->      1',/,
      1     '        NOT FIRST BUT LAST   ->     10',/,
      1     '        NOT FIRST, NOT LAST  ->     11')
            READ(5,31)MOVE
 31         FORMAT(I3)
            IF(MOVE.GE.10)GO TO 26
            WRITE(6,*)' ENTER XLAB'
            READ (5,29)XLAB
 29         FORMAT(A40)
 26         IF(MOVE.EQ.01.OR.MOVE.EQ.11)GO TO 27
            WRITE(6,*)' ENTER YLAB'
            READ (5,29)YLAB
 27         IF(MOVE.NE.00)GO TO 36
            LABEL=4
            ISCL=12
            GO TO 32
 36         IF(MOVE.NE.01)GO TO 30
            LABEL=4
            ISCL=12
            GO TO 32
 30         IF(MOVE.NE.10)GO TO 35
            LABEL=4
            ISCL=8
            GO TO 32
 35         LABEL=4
            ISCL=8
 32         WRITE(6,*)' ENTER CURVLAB (Two spaces first)'
            READ(5,33)CURVLAB
 33         FORMAT(A6)
            CALL QPICTR(Y,10,NPTS,QY(2,3),QX(1),QMOVE(MOVE),
      1       QLABEL(LABEL),
      1       QISCL(ISCL),QCURVLAB(CURVLAB),QXLAB(XLAB),
      1       QYLAB(YLAB),QXSCL(XSCL))
 19       CONTINUE
          GO TO 10
 1000     STOP
          END


          FUNCTION COM(N1,N2)
            COM=KFACT(N1)/(KFACT(N2)*KFACT(N1-N2))
          RETURN
          END
          FUNCTION KFACT(N)
          KFACT=1
          IF(N.LT.2)THEN
```

```
              RETURN
            ELSE
              DO 1 J=N,2,-1
                KFACT=KFACT*J
 1            CONTINUE
            ENDIF
            RETURN
            END


            SUBROUTINE EXBT(N,B,E)
            COMMON C(10,10,10)
            DIMENSION B(10,10),E(10),EIK(10),
          1 B1(10,10),B2(10,10),T(10,10)
            DO 600 I=1,N
            DO 15 K=1,N
 15         EIK(K)=E(I)-E(K)
            DO 500 J=1,N
            IF(J-I)100,500,200
 100        IF(J-1)110,110,150
 200        IF(I-1)300,300,400
 300        IF(J-I-1)110,110,150
 400        IF(J-I-1)110,150,150
 110        DO 5 K=1,N
            DO 5 L=1,N
 5          B1(K,L)=B(K,L)
            DO 20 K=1,N
            B1(K,K)=B(K,K)-E(J)
            DO 20 L=1,N
 20         B1(K,L)=B1(K,L)/EIK(J)
            GO TO 500
 150        DO 40 K=1,N
            DO 40 L=1,N
 40         B2(K,L)=B(K,L)
            DO 25 K=1,N
            B2(K,K)=B(K,K)-E(J)
            DO 25 L=1,N
 25         B2(K,L)=B2(K,L)/EIK(J)
            DO 30 K=1,N
            DO 30 L=1,N
            T(K,L)=0.0
            DO 30 M=1,N
 30         T(K,L)=T(K,L)+B1(K,M)*B2(M,L)
            DO 35 K=1,N
            DO 35 L=1,N
 35         B1(K,L)=T(K,L)
 500        CONTINUE
            DO 60 K=1,N
            DO 60 L=1,N
 60         C(I,K,L)=B1(K,L)
 600        CONTINUE
            RETURN
            END
```

```
C       PROGRAM AVALNET
C       ***************
C
        SUBROUTINE EQSIM
C
C            This  program  is a subroutine  of the program
C       DYSYS. The  program  calculates  the  availability
C       of the nodes and the  network of a  fully connected
C       network with unreliable nodes and links.
C            New parameters are defined below.
C
C       NODE: RR2 =  repair  rate  from  the  failed  state
C                    [repairs/hour]
C
C       CENTRAL COMPUTER: FRC = failure rate of the  central
C                               computer [failures/hour]
C
C               RRC = repair rate of  the  central  computer
C                     [repairs/hour]
C
C       LINKS: RRL = repair rate of the links [repairs/hour]
C
C
      COMMON T,DT,Y(30),F(30),STIME,FTIME,NEWDT,NEWRUN,N,
     1       IPR,ICD,ICN,TBREAK,PNEXT,TBACK
      DIMENSION S(1),P(10)
      INTEGER TT
      IF(NEWRUN.EQ.-1)THEN
        WRITE(6,1)
1       FORMAT('  ENTER THE FOLLOWING DATA',///,
     1  '  NODES : NU   = Number of units per node',/,
     1  '  *****   FR   = Failure rate of individual
     1   units',/,
     1  '          RR   = Repair rate (minor)',/,
     1  '          RR2  = Repair rate (major)',/,
     1  '          COV  = Coverage',/,
     1  '          NN   = Number of nodes (with CC)',/,
     1  '  CENTRAL COMPUTER : FRC = Failure rate of the
     1   CC',/,
     1  '  ****************    RRC = Repair rate of the
     1   CC',/,
     1  '  LINKS : FRL = Failure rate of the links',/,
     1  '  *****   RRL = Repair rate of the links',/)
        ACCEPT*,NU,FR,RR,RR2,COV,NN,FRC,RRC,FRL,RRL
      ENDIF
      IF(NU.EQ.3)THEN
        F(1)=-3.*FR*Y(1)+RR*Y(2)+RR2*Y(4)
        F(2)=3.*FR*COV*Y(1)-(2.*FR+RR)*Y(2)
        F(3)=2.*FR*COV*Y(2)-(FR+RR)*Y(3)
        Y(4)=1.-(Y(1)+Y(2)+Y(3))
        AN=1.-Y(4)
```

```
          Y(7)=AN
        ELSE IF(NU.EQ.4)THEN
          F(1)=-4.*FR*Y(1)+RR*Y(2)+RR2*Y(5)
          F(2)=4.*FR*COV*Y(1)-(3.*FR+RR)*Y(2)+RR*Y(3)
          F(3)=3.*FR*COV*Y(2)-(2.*FR+RR)*Y(3)+RR*Y(4)
          F(4)=2.*FR*COV*Y(3)-(FR+RR)*Y(4)
          Y(5)=1.-(Y(1)+Y(2)+Y(3)+Y(4))
          AN=1.-Y(5)
          Y(7)=AN
        ELSE IF(NU.EQ.5)THEN
          F(1)=-5.*FR*Y(1)+RR*Y(2)+RR2*Y(6)
          F(2)=5.*FR*COV*Y(1)-(4.*FR+RR)*Y(2)+RR*Y(3)
          F(3)=4.*FR*COV*Y(2)-(3.*FR+RR)*Y(3)+RR*Y(4)
          F(4)=3.*FR*COV*Y(3)-(2.*FR+RR)*Y(4)+RR*Y(5)
          F(5)=2.*FR*COV*Y(4)-(FR+RR)*Y(5)
          Y(6)=1.-(Y(1)+Y(2)+Y(3)+Y(4)+Y(5))
          AN=1.-Y(6)
        ENDIF
C
C
C       CENTRAL COMPUTER AVAILABILITIES
C
C
        ANA=(FRC*EXP(-(FRC+RRC)*T)+RRC)/(FRC+RRC)
C
C
C       LINK AVAILABILITIES
C
C
        AL=(FRL*EXP(-(FRL+RRL)*T)+RRL)/(FRL+RRL)
C
C
C       UNAVAILABILITY OF THE NET CONSIDERING RELIABLE NODES
C
C
        P(1)=1.0
        DO 10 TT=2,NN
          AUX=0.0
          DO 20 IU=1,TT-1
            IF(RENODE.NE.1.0)THEN
              AUX=AUX+COM(TT-1,IU-1)*P(IU)*(1.0-AL)**(IU*
     1  (TT-IU))
            ELSE
              AUX=0.0
            ENDIF
20        CONTINUE
          P(NN)=1.0-AUX
10      CONTINUE
C
C
C       PROBABILITY THAT ALL OF THE NODES ARE WORKING
C
C
```

```
      S(NN)=ANA*Y(7)**(NN-1)
C
C
C     AVAILABILITY OF THE NET CONSIDERING
C     UNRELIABLE NODES
C
C
      Y(8)=S(NN)*P(NN)
C     FOR THE NODE
      Y(9)=1.0-Y(7)
C     FOR THE NET
      Y(10)=1.0-Y(8)
      Y(11)=1.-S(NN)
      Y(12)=1.0-P(NN)
      Y(13)=1.0-AL
      RETURN
      END


      FUNCTION COM(N1,N2)
        COM=KFACT(N1)/(KFACT(N2)*KFACT(N1-N2))
      RETURN
      END


      FUNCTION KFACT(N)
      IF(N.LT.2)THEN
        KFACT=1
        RETURN
      ELSE
        KFACT=1
        DO 1 J=N,2,-1
          KFACT=KFACT*J
1       CONTINUE
      ENDIF
      RETURN
      END
      PROGRAM QWR
C     ***********
C
C     The following program calculates characteristics of
C     a M/M/KC/K/K queuing system.
C
C     INPUTS: ALFA     = Failure rate of the units [f./h]
C             AMU      = Repair rate, one unit [rep./hour]
C             K        = Number of units
C             KC       = Number of repairmen
C
C     OUTPUTS:ALQ      = Av. number of units to be repaired
C             WQ       = Av. waiting time in queue
C             D        = Probab. that a unit has to wait to
C                        be repaired
C             EQ       = Av. waiting time for those units
C                        that must wait repair
C             AL       = Av. number of units down
C             W        = Total av. waiting time
C             ALAMBDA  = Rate of incoming units
C             RHO      = Server utilization
C
      DOUBLE PRECISION ALFA,AMU,PP,POAUX,PZERO,P,ALQ,AL,
     1 ALAMBDA,W,WQ,D,EQ,RHO,TEST
      DIMENSION P(100),PP(100)
1     WRITE(6,2)
2     FORMAT(' ENTER: ALFA Failure Rate of Units [1/hr]',
     1 /,8x,'AMU  Repair Rate of Units [1/hr]',/,8x,
     1 'K    Total # of units',/,8X,'TEST',/)
      READ(5,3)ALFA,AMU,K
3     FORMAT(2D10.3,I4)
      TEST=1.0D-20
      IF(M.EQ.1)GO TO 6
4     WRITE(6,*)' ENTER: KC    # of Repairmen'
      READ(5,5)KC
5     FORMAT(I4)
C
C     HERE BEGINS THE PROGRAM
C
6     DO 7 I=1,KC
7       PP(I)=(FN(K,I)/FACT(I))*(ALFA/AMU)**I
      DO 8 I=KC+1,K
        AKC=FLOAT(KC)
C
C     THE PARAMETER  TEST  IS USED TO AVOID CALCULATION OF
C     LARGE FACTORIALS.
C
        IF(PP(I-1).LT.TEST)THEN
          PP(I)=0.0
          GO TO 8
        ELSE
          PP(I)=FN(K,I)*(ALFA/AMU)**I/
     1          (FACT(KC)*AKC**(I-KC))
```

```
             ENDIF
8          CONTINUE
           POAUX=0.0
           DO 9 I=1,K
9            POAUX=POAUX+PP(I)
           PZERO=1./(1.+POAUX)
           DO 10 I=1,K
             P(I)=PP(I)*PZERO
             WRITE(6,103)I,P(I)
103          FORMAT(10X,'I =',I3,5X,'P(I) =',E10.3)
10         CONTINUE
C
C          **CALCULATE OTHER PARAMETERS**
C
           ALQ=0.0
           DO 11 I=KC+1,K
11           ALQ=ALQ+(I-KC)*P(I)
           AL=(AMU*ALQ+ALFA*K)/(ALFA+AMU)
           ALAMBDA=ALFA*AMU*(K-ALQ)/(ALFA+AMU)
           WQ=ALQ*(1./ALFA+1./AMU)/(K-ALQ)
           W=WQ+1./AMU
           D=0.0
           DO 12 I=KC,K
12           D=D+P(I)
           EQ=WQ/D
           RHO=ALAMBDA/(AMU*KC)
C
C          **OUTPUT**
C
           WRITE(6,13)ALFA,AMU,K,KC,ALQ,WQ,D,EQ,AL,W
           WRITE(6,13)ALAMBDA,RHO
13         FORMAT(//,10X,'INPUTS',/,10X,'******',//,
     1      ' ALFA    Failure rate of units', 24x,f10.5,
     1      ' 1/hr',/,' AMU     Repair rate of one repairman',
     1       17x,f10.5,' 1/hr',/,
     1      ' K       Number of servers',29x,I3,/,
     1      ' KC      Number of repairmen',27x,I3,//,
     1      10X,'OUTPUTS',/,10X,'******',//,
     1      ' ALQ     Average # of units to be repaired',10x,
     1      E12.5,/,
     1      ' WQ      Average waiting time in queue to repair',
     1      6X,F10.5,' hrs',/,
     1      ' D       Prob. a unit has to wait to be repaired',
     1      6x,F10.5,/,' EQ      Average waiting time for those
     1      who must wait',1x,
     1      f10.5,' hrs',/,
     1      ' AL      Average # of units down ( ALQ+in repair )
     1      ',4X,f10.5,/,
     1      ' W       Total average waiting time ( queue +
     1      repair )',f10.5,' hrs',/,
     1      ' ALAMBDA Rate of incoming units',23x,f10.5,'1/hr',
     1      /,' RHO     Server utilization',27x,f10.5,//)
          WRITE(6,*)' TO MODIFY ALFA, AMU OR K  ---> 1'
```

```
      WRITE(6,*)' TO MODIFY KC                       ---> 2'
      WRITE(6,*)' TO END THIS RUN                    ---> 0'
      READ(5,5)M
      IF(M.EQ.1) GO TO 1
      IF(M.EQ.2) GO TO 4
      STOP
      END
      FUNCTION FACT(N)
      AN=FLOAT(N)
      FACT=1.0
      IF(AN.LT.2.0)THEN
        RETURN
      ELSE
        DO 1 J=N,2,-1
1         FACT=FACT*J
      ENDIF
      RETURN
      END
      FUNCTION FN(M,N)
      AM=FLOAT(M)
      FN=AM
      IF(N.EQ.1)THEN
        RETURN
      ELSE
        DO 1 J=1,N-1
1         FN=FN*(AM-J)
        RETURN
      ENDIF
      END
```

INDEX