ALGORITHMS FOR OPTIMISING CONTROL

BY

KAMBHAMPATI. C

A THESIS SUBMITTED FOR THE DEGREE

OF

DOCTOR OF PHILOSOPHY

CONTROL ENGINEERING CENTRE
CITY UNIVERSITY
NORTHAMPTON SQUARE
LONDON EC1V0HB

JULY 1988

CONTENTS

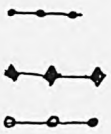
LIST OF TABLES

LIST OF FIGURES

Key to figure 5.3

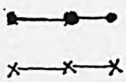
Linear Model
Non-Linear Model I
Non-Linear Model II

Key to figure 5.8



Linear Model
Non-Linear Model I
Non-Linear Model II

Key to figure 5.10



Linear Model
Non-Linear Model

ACKNOWLEDGEMENT

I would like to thank Dr. J. E. Ellis, who throughout this research has not only been my supervisor and teacher, but has also been my guide and mentor. Without his advice and encouragement and understanding character this thesis would not have seen the light of day.

I would also like to thank [redacted], the Director of the Control Engineering Centre, not only for his help and valuable suggestions during the course of this research, but also for having given me the opportunity to carry out this work.

I wish also to thank [redacted] from Xian Jai Tong University China and [redacted] from the Technical University of Warsaw, for their help and advice regarding the theoretical and practical aspects of the research. Some of the results in this thesis were obtained based on their suggestions.

I would also like to thank [redacted], and [redacted], who as managers of the Prime 750 Computer facility gave me invaluable assistance.

Finally I would like to thank ICI plc for having permitted the use of their model of the Fuel-Gas Mixing system.

Last but not the least I would like to thank all the members, both staff and students, of the Control Engineering Centre who assisted me while working on this project.

DECLARATION

The author grants powers of discretion to the University Librarian to allow this thesis to be copied in whole or in part without further reference to him. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

PUBLICATIONS

The following papers based on the work described in this thesis will be published soon :

Ellis. J. E, Sheng. G. H, Kambhampati. C, and Roberts. P. D (1988) " Approaches to the Optimising Control Problem " — Int. J. Sys. Sci

Kambhampati, C, Ellis J. E, Roberts. P. D " Generalisation of Integrated System Optimisation and Parameter Estimation Techniques" — Automatica

The following paper based on the work described in this thesis has been presented at the IEE CONTROL 88 Conference, Oxford

Kambhampati. C, Ellis. J. E (1988) " A Passive Coordination Technique for Solving the On-Line Decentralised Optimising Control Problem " — IEE Int. Conference CONTROL 88, University of Oxford, Pub No 285 pp 532 — 535

The following paper based on the work described in this thesis has been sent in a draft form for presentation at the IFAC/IFORS/IMACS Symposium Large Scale Systems : Theory and Applications to be held in the German Democratic Republic

Kambhampati. C, Ellis. J. E "On Model-Reality Differences in Integrated System Optimisation and Parameter Estimation Techniques".

The following papers, which are related to the work described in this thesis, have been presented/submitted for publication

Lin J, Roberts. P. D, Kambhampati. C, Hu B, (1987) " A Two Model Approach in a Double Iterative Structure for Integrated System Optimisation and Parameter Estimation of Large Scale Industrial Processes" — The National Conference on Automatic Control, China, MAY 1988

Lin J, Kambhampati. C, Roberts. P. D (1987c) " A New Two Model Approach in a Double Iterative Structure for Integrated System Optimisation and Parameter Estimation of Large Scale Industrial Processes" — Submitted to the IMA Journal on Mathematical Control and Information

ABSTRACT

This research is concerned with the problem of determining optimum operating conditions of a system where a mathematical model is employed to compute the performance index. A family of algorithms known as Integrated System Optimisation and Parameter Estimation (ISOPE) algorithms have been developed. These algorithms are based on the Modified Two-step approach, where the model contains certain free parameters which are estimated periodically using system measurements, and are then used to solve an optimisation problem in an iterative manner. An advantage with these methods is that the optimum operating conditions are obtained even when the model structure is not identical to that of the system. It is then quite evident that the model-reality differences can be described by the magnitude of the parameters which are estimated. However, such a simplistic notion of model-reality differences is not an accurate description of these differences. For often it is possible that the value of the parameters estimated are small, but the efficiency of the iterative process could be poor. Then one of the main aim of this research is to identify what consitutes these model-reality differences, and how do the ISOPE algorithms overcome these differences.

The ISOPE algorithms studied in this research cover both the centralised as well as the decentralised cases. However, a detailed analysis of the model-reality differences, is under taken for the centralised case. For it can be seen that the analysis is also valid for the decentralised cases.

The ISOPE algorithms have many similarities. They are derived in a similar manner, the algorithmic and control structures are similar, so also is the problem formulation for the various algorithms. It is in this context that a generalisation of these algorithms is attempted.

NOMENCLATURE

Unless otherwise stated the following nomenclature is used throughout this thesis.

Symbol	Definition
$\tilde{A}$, A_*, $A(\alpha)$, $A(\Theta)$	Lipschitz constants
$\tilde{a}$, a_*, $a(\alpha)$, $a(\Theta)$	Monotone constants
arg	Argument
C	Control Vector
C_i	Control Vector for the i^{th} sub-system
$\hat{C}$	Control solution mapping of optimisation
F	Model Input/Output mapping
F_i	Model Input/Output mapping for the i^{th} sub-system
F_*	System Input/Output mapping
F_{*i}	System Input/Output mapping for the i^{th} sub-system
G, g, G_*, g_*	Constraint mappings
H, H_i	Interconnection matrices
I	Identity matrix of appropriate dimensions
K_*	System Input/Output mapping
K	Model Input/Output mapping
$L_*(.)$	Lagrangian, based on system I/O mappings
L(.)	Lagrangian based on model I/O mappings
N	Number of sub-systems
N(a)	Neighbourhood of some point a
P	Price vector
P_i	Price vector for the i^{th} sub-system
q(.), Q(.)	Performance index (Objective function)
$q_i(.)$, $Q_i(.)$	Performance index of the i^{th} sub-system
T(V)	A continuous function of V
U	Interaction input vector
U_*	Interaction input vector
U_i	Interaction input vector of the i^{th} sub-system
V	Controls which are applied to the system
Y	Output Vector
Y_i	Output vector of the i^{th} sub-system
α	Free model parameter vector
α_i	Model parameter vector for the i^{th} sub-system

β	Lagrangian multiplier
$\delta, \sigma, \tau, \epsilon$	Positive scalars
λ	Lagrangian multiplier, modifier
μ, η	Kuhn–Tucker multipliers
ς	Convergence accuracy
ξ_V, ξ	Iterative loop gains
ξ_P	Step coefficient
Ω	Solution set of the algorithm
$\Psi(.)$	Operator equation for updating controls
Θ	Generalised model parameter vector
$\Phi(X,Z)$	Generalised quadratic penalty function
Ξ	Set of all Θ
Λ_{min}	Minimum eigen value
$\Lambda_1(V), \Lambda_2(V), \Lambda_3(V)$	Functions of V
ρ	Penalty factor
$\mathcal{C}$	Set of feasible controls
$\mathcal{CU}$	Set of feasible controls and inputs
$\Re^a$	An a-dimensional real space, a=m, n, 1
$\mathcal{C}_n, \mathcal{C}_m$	Matrix operations of adding rows or columns
$\mathcal{G}$	Generalised feasible set
$\mathcal{A}$	Set of all α
$\|\cdot\|$	Euclidean Norm
$\|\cdot\|_S$	Spectral Norm

CHAPTER 1 INTRODUCTION

1.1 INTRODUCTION

The traditional concept of control of industrial systems is the maintenance of certain selected reference variables at their desired value and/or to follow changes in the reference variables, in spite of disturbances acting on them. A control scheme which incorporates this view ignores other factors which are becoming increasingly important, viz:

a) the increasing demand on the product quality.

b) the ever depleting sources of raw materials, and energy, which places greater emphasis on the productivity or efficiency of the process.

Then, while designing a control system, if due care is not taken to ensure that the productivity of the system is maintained at its best, and hence the optimal use of raw materials, the system will not function efficiently. That is, we determine that set of controls at which the system operates most efficiently. In practice, the efficient operating condition of the system can be achieved by either minimising or maximising an index of performance, where the independent variables take the form of set-points, which are then used as reference inputs in controlling the system. Thus the design or modelling of the control scheme for a system can be seen as consisting of two parts :

a) part where the control is optimised.

b) part where the system variables are maintained at their desired value.

Such an integration of the two concepts leads to the notion of Integrated System Control **(Lefkowitz (1977))**. Here, the overall control model is seen as a two level scheme, where the lower level is a follow-up or regulatory control layer, and the upper level is the optimising control layer also known as the supervisory control layer. In fact, such a multi-strata model can be identified in many automated industrial systems, as can be seen in figure 1.1. Such multi-strata models have some typical features, which are **(Mesarovic et al (1970))**:

a) the higher level units are concerned with a larger portion or broader aspects of the overall system behaviour.

b) the decision period of a higher level unit is longer than that of the lower level units.

c) the higher level units are concerned with the slower aspects of the

overall system behaviour.

Thus, in terms of the two layers, we see that the follow-up control layer has a shorter span of time and has to maintain the performance of the system at some prescribed level, inspite of fast disturbances acting on it. The controllers at the lower level generally take the form of two or three term, proportional plus integral (PI) or proportional plus integral plus derivative (PID), controllers, where the reference inputs are provided by the higher level.

On the other hand the task of the supervisory layer is to ensure that the overall system performs efficiently. The task of the supervisory layer is, thus, to determine and maintain optimal values of the set-points (reference inputs) under slowly varying disturbances. The problem, thus essentially exists in the steady-state, and is basically one of optimising a given measure of performance.

Here, in this thesis, we are concerned with investigating the various techniques for optimising control. In other words we study the various techniques for solving the supervisory control problem. In the following chapters, we describe some of the techniques which can be implemented, either in a centralised manner, or in a decentralised manner.

1.2 OPTIMISING CONTROL

It has already been mentioned that the task of the supervisory layer is to determine optimum set-point values, by optimising a given measure of performance. This criterion is the maximisation and minimisation of a given performance index. There are three ways in which the problem of optimising the control can be solved. We can use the mathematical descriptions (models) of the system, and apply the results so obtained to the system. The second method is to utilise the system measurements directly. Both these techniques have their disadvantages. The third is a combination of these, and is known as the adaptive technique.

The model, which is used to represent a system, must satisfy certain sufficiency conditions **(Ellis and Roberts (1982))** in order to guarantee the system optimum. These conditions demand that the model and the system have certain properties the same. This demand is not, in general, realistic, as we usually have to make some form of a compromise in the modelling process. Optimisation techniques are essentially complex mathematical processes, and a complex model — so that it is a faithful representation of the system — would tend to increase the computational complexity of the optimisation process. On the other hand, a very simple model would not be a reasonably accurate description of the system and the use of which would be likely to result in sub-optimal results. At the same time, one

cannot model external uncertainities — like measurement noise, environmental conditions etc —, which play a part in the performance of the system, accurately. Hence, using a model in solving the optimising control problem would not in general provide optimal results.

The use of measurements directly from the system means that every time measurements are required, we have to ensure that the system attains its steady-state. If the system response is slow, this method can be time consuming. Also the presence of uncertainities could result in poor results.

The third method is to integrate these two methods and it leads to the concept of adaptive models. An adaptive model is a model which incorporates free parameters, which are periodically estimated and the model updated. This updated model is now used in determining the optimum set-point values. These two problems of parameter estimation and system optimisation are solved repeatedly until, hopefully, the iterative process converges to the optimum. However, such a scheme will not in general produce optimal results **(Roberts (1979), Ellis and Roberts(1982))**. The reason for this is that the model still does not satisfy the sufficiency conditions.

To overcome these problems **Roberts (1979)** proposed a model based method, known as a modified two-step method (MSTP), where an application of the method to a problem produces optimum results. The key feature of this procedure is to formulate an optimisation problem, which integrates the problems of parameter estimation and system optimisation. An analysis of the first-order necessary conditions of such an integrated problem results in a new model based problem, (see chapter 3), the solution of which is the system optimum. The solution of this problem is iterative. Using this method it does not matter whether the model fulfills the sufficiency conditions or not. In fact convergence will always be obtained whether the system optimisation problem is convex or not **(Brdys and Roberts (1987))**. In fact, using this algorithm, as a basis, a whole range of algorithms which provide optimum results have been developed. All these algorithms are classified as Integrated System Optimisation and Parameter Estimation (ISOPE) techniques. These cover various control configurations for both centralised and hierarchical decentralised situations.

1.3 CONTROL STRUCTURES

The system optimisation problem can be solved in either a centralised manner or a decentralised manner. The decision of which type of control

configuration to use, centralised or decentralised, would depend on a number of factors. In fact, there are no general hard and fast rules, which would tell us under what conditions a decentralised configuration is more advantageous than a centralised configuration **(Maxwell(1983))**. However, one can provide some guide lines which could would be helpful in deciding the type of configuration. For instance if the system dimension (number of set-points) are large, or if the system is such that it can be broken down into a number of different sub-systems, then it could be advantageous to evolve a decentralised control configuration. However, here the economics of designing and implementing the system have to be taken into account. Sometimes even if the system has the features mentioned above, it is posible that it could be more costly to implement a decentralised configuration than a centralised one **(Maxwell(1983))**.

1.3.1 Centralised Control Structures

In a centralised control configuration, all the information needed to make a control decision is taken to and processed at a central location. In such a case, the system is treated as a single entity, and it does not matter whether the overall system consists of a number of sub-systems or not. Figure 1.2, gives a schematic representation of the information flow which takes place in a centralised system.

The MSTP algorithm in its original form, solves the centralised system optimisaiton problem. Here, the features of the simple two-step approach are retained. We still estimate the parameters of the model, and use these parameters to update the model. This updated model is then used to solve the optimisation problem. However, in the MSTP approach the optimisation problem is modified by a parameter known as the modifier. Indeed, as wiil be seen, the modifier plays a crucial role throughout the ISOPE algorithms. This modifier quantifies the violations of the sufficiency conditions by the model. To evaluate this modifier, derivative information from the system is required, and this is the main disadvantage with this method. Efforts have been made to either eliminate the need for derivative information, or to reduce this requirement. As a result a number of extensions have been made to this approach **(Brdys et al (1986b), Chen (1986), Ellis et al(1988))**.

However, a centralised configuration has some disadvantages. Any optimisation technique is a mathematical procedure, where the rules for evaluating the optimum point are given in the form of an algorithm. This algorithm is then implemented on a computing machine, then if the system dimensions — in terms of

the number of the set-points — is very large, then the computational task will also be very large and complex. Also, if the system is spread over a large area, then the transfer of information to the central point could prove to be difficult and costly. Then, depending on the economics a decentralised control structure could prove to be advantageous.

1.3.2 Decentralised Control Structures.

In a decentralised control configuration, the system to be controlled is treated as being made up of different sub-systems and each sub-system has its own Local Decision Unit (LDU) which defines the set-points for that sub-system. If this decision is taken in isolation, that is, not taking into account the effect it has on other sub-systems, then it may be found that the system is operating not at the system optimum, but some sub-optimal operating point. Such a conflict occurs because the optimum for the individual sub-systems need not be the global optimum. Then, in such a case the control decision has be to made taking into account the effect it has on the other sub-systems, for the aim here is to obtain the global optimum. This, then means that there is a need for some form of coordination between the control decisions taken by the separate local units. This coordination of the LDU's influences their decision making process, and prevents any conflicts between the individual decisions. Thus, due to the decomposition of the global problem into several sub-problems, another problem has to be solved. This is the coordination problem, and it is the main feature of the algorithms which solve the optimising control problem in a decentralised manner.

The decentralised control problem, then, arises when a system is acted upon by a number of different controllers who are not able to communicate with each other on-line or are not in a position to process all the information available to them. However, it is assumed that the decision making is cooperative, that is the controllers may use some a priori rules so that some overall system objective is achieved **(Singh, Titli and Malinowski (1983))**.

Established hierarchical decentralised algorithms use the principles of 'decomposition and coordination' **(Cohen (1978), Singh and Titli (1978), Findeisen et al (1980))**. Here, the optimisation problem is solved using a two level structure, see fig 1.3, where at the lower level the optimisation problem for each sub-system is solved by a LDU and coordination is done at the higher level so that the overall goal of optimising the global problem is achieved. The principle here, is that the objective of the system is modified in such a manner so that when decomposition — into separate sub-problems — of the global problem is made, we take into account

the various conflicts which are likely to take place between the sub-problems. The modification is made by a parameter known as the 'price vector', which is evaluated by the cooridnator. These techniques, however, use fixed models and as has been mentioned earlier, the use of such models in general would not produce optimum results.

However, when these techniques are integrated with the techniques of the modified two-step (MSTP) approach, it is seen that a whole family of optimal algorithms are obtained **(Michalska et al (1985), Brdys and Roberts (1986a), Chen (1986), Abdullah (1988))**. These algorithms are optimal in the sense that an application of any of these to a problem would produce the system optimum. Here, the problem at the higher — coordination — level, is dynamic, in the sense that apart from evaluating a parameter known as the modifier through some simple matrix manipulations, we still have to solve a separate coordination problem which is to evaluate the price vector. This evaluation is done in an iterative manner, and the algorithm used for solving this problem is a simple gradient type algorithm **(Cohen (1978), Singh and Titli (1978), Findeisen et al (1980))**. The algorithms obtained by such an integration are termed 'Dynamic Coordination', **(Ellis et al (1986))**.

On the other hand, if we extend the MSTP algorithm, so that we can solve the hierarchical decentralised problem, we see that the need for solving the additional coordination problem is eliminated **(Ellis et al (1986))**. Here the coordinator is still retained, and its sole task is to evaluate the modifier through some simple matrix manipulations and the coordination is termed passive as opposed to the dynamic nature of the other algorithms. However, there is one drawback with this algorithm, in that it needs two models : an output model and an interaction model. The interaction model is needed to take into account the interactions taking place between the sub-systems. A similar technique was put forward by **Brdys and Roberts (1986a)**, where the need for an additional model, was eliminated. This was made by using the information available from the system more efficiently. In these techniques the task of the coordinator is passive, and they are therefore known as algorthms with passive coordination.

1.4 OBJECTIVES AND SCOPE OF THE THESIS

The case for the development of adaptive model based optimisation techniques is quite evident, as these techniques utilise the advantages of both the direct and the model based methods. It has been found that a simple two-step iterative scheme consisting of :

a) model parameter estimation

b) updating the model using the parameters estimated in step a, and using this updated model to solve the optimisation problem,

will not in general provide the optimum results (**Roberts (1979), Ellis and Roberts (1982)**). The reason for this is that, in such a scheme, the model does not satisfy the sufficiency conditions, and in general, it can be said that a model will not satisfy these conditions, because if it did, it would mean that the system and the model representing it have the same important properties, or even the same structure, which is not realistic.

To overcome these difficulties **Roberts(1979)** proposed a technique known as the modified two-step (MSTP) approach. Here, the features of the simple two-step approach are retained, the difference being that the optimisation problem is modified by a parameter, known as the modifier, which quantifies the violations of the sufficiency conditions. This technique has proved useful in providing optimum solutions to the on-line system optimisation problem (**Roberts (1979), Roberts and Williams (1981), Roberts and Lalui (1982), Ellis and Roberts(1982, 1985)**). The important feature of the method is that the system optimum is always achieved irrespective of whether the system optimisation problem is convex or not (**Brdys and Roberts (1987)**). However, this method has a draw back in that derivative information from the system is needed, in order to evaluate the modifier. These derivatives are obtained using finite difference approximation techniques. Hence the number of simultaneous set-point changes needed so that convergence to the optimum is obtained becomes important. If there are n control set-points, then for every iteration n set-point changes are needed, so that the derivatives can be eveluated at each iteration. Since the optimising control problem is solved in the steady state, this process could become time consuming, as it has to be ensured that, after each set of simultaneous set-point changes, the system attains, a steady-state. Attempts to improve on the performance of the technique in this regard have been made on two fronts, viz:

a) Eliminate the need for evaluating the derivatives

b) Reduce the number of iterations, and hence the number of times the derivatives have to be evaluated.

Towards this end a number of extensions have been made to the MSTP technique. An attempt has been made to eliminate the need for evaluating the derivatives by **Chen (1986)**. Here, a three level structure has been proposed, and it was found that although optimal results are obtained, this scheme is less efficient than the MSTP. For although the derivatives are no longer needed the total number

of set-point changes exceeds the number when using the MSTP approach **(Chen (1986))**.

On the other hand attempts to reduce the number of iterations, and hence the number of simultaneous set-point changes have been found to be more successful. It has been found that it is beneficial if the performance index is augmented by a quadratic penalty function **(Brdys et al (1986b))**. The other method which has proved sucessfull is known as the Approximate Linear Model (ALM-ISOPE) method **(Sheng and Ellis (1985), Kambhampati and Ellis (1987), Ellis et al (1988))**. This method has an advantage in that we do not have to provide the algortihm with a model a priori.

As was mentioned earlier, the MSTP approach in its original form, is suitable for solving the system optimisation problem in a centralised manner. However, for solving the hierarchical decentralised problem the MSTP algorithm has to be modified. This is done, as was mentioned earlier, by integrating the MSTP approach with the principle of decomposition and coordination **(Michalska et al(1985), Brdys and Roberts (1986a), Chen(1986), Abdullah (1988))**. Such an integration gives rise to algorithms which are known as dynamic coordination algorithms **(Ellis et al (1986))**. As a result of such an integration, and depending on the manner in which the information available from the system is utilised, it is seen that a number of algorithms can be developed, all of which are optimal in the sense that an application of them to a given problem provides us with the system optimum. Although these algorithms are developed using the same principles, it is seen that all of them do not have the same efficiency in providing the optimum. However, it has been seen **(Roberts et al (1988))** one particular algorithm, which uses both input and output feed back information is the most efficient. Here, as is the case with the MSTP derivative information is needed so that the problem can be solved and the resulting solution is optimal. In order to reduce this need, new algorithms have been developed. For these algorithms, the performance index is again augmented by a quadratic penalty function **(Roberts et al (1988))**.

Generally, in all ISOPE algorithms, augmented or unaugmented the control set-points are updated using an under-relaxation scheme **(Cohen 1978))**. In such a scheme we change the values of the set-points by a small amount, rather than apply the results of the modified optimisation problem directly, so that the iterative process is stable. Here, we need to select certain parameters, known as the iterative loop gains. These are obtained through trial and error, which in an on-line situation could pose difficulties. In order to overcome these difficulties, **Tatjewski and Roberts(1987a, 1987b)** proposed Newton-like schemes for updating the control set-points, where the mechanism for updating the controls is more precise

and has been found to have good convergence properties.

The aim of this thesis is to undertake a detailed analysis of the algorithms mentioned above. It is expected, that, through such a study we can understand the properties of the algorithms better. The properties of the algorithms can be divided into two categories namely

a) Optimality properties

b) Convergence properties.

A study of the optimality properties, gives us conditions under which the solution, obtained by an application of a given algorithm, would be the same as that of the system. On the other hand, a study of the convergence properties gives an idea of the properties the problem has to have, so that the algorithm can converge to the optimum of the system in the most efficient manner. The result of such a study is that we can get some idea about the applicability of the algorithm. For, if such an understanding is not there, then some times it could prove futile to apply an algorithm to a problem and this problem does not satisfy the conditions for convergence to the optimum.

Apart from studying these properties, some of the other objectives are concerned with understanding what are model-reality differences, the significance of the modifier in the ISOPE algorithms and how far the model-reality differences are reflected in it, and, the effect of augmentations on the properties of the algorithm.

The analysis presented for the centralised algorithms, in chapter 4, answers these questions. A similar analysis is not undertaken for the decentralised case, for clearly, the situation where there are a number of sub-systems, the analysis performed for the centralised case is valid for the decentralised cases also.

Also, it has been mentioned that, the algorithms which solve the hierarchical decentralised optimising control problem can be divided into two groups, namely algorithms with dynamic coordination and those with passive coordination. Then, the question arises which type of coordinations would be suitable and more advantageous to use. Of course some of the advantages are quite apparent, as will be seen in chapter 6, where these two types of algorithms are described. It is then intended to study these two classes, and perfrom simulations so as to make a comparision between them.

A simulation study of the performance of these algorithms is also undertaken. The objective here is to verify the results of the analysis of an algorithm. A simulation study helps us understand some of the properties of the

algorithm, for often it is found that the conditions for convergence and optimality of an algorithm, obtained through an analysis of the algorithm are stricter than required. This could be due to the nature of the algorithm and it may not be possible to develop weak convergence conditions and the conditions obtained will be much stricter than required, **(Brdys et al (1986c))**. Then, the aim of any simulation exercise is :

a) To verify the theoretical results obtained

b) To investigate the algorithm further so that those properties of, the algorithm, which could not be assessed theoretically could be brought out.

In fact, there is one more point to be kept in mind; that is, any proposed algorithm or a technique has to have a practical viability. To test this particular aspect we have to mimic a realistic situation, whereby there would be disturbances in the form of noise, present in the system. All the algorithms must perform satisfactorily in such a situation. However, this particular aspect of a simulation study is beyond the scope of the present thesis.

Another objective is to develop a generalised set of results for all the algorithms which are classified as ISOPE algorithms. For, a remarkable feature of these algorithms is that whether the algorithm solves the problem in a centralised manner or in a decentralised manner, the mode of deriving these algorithms is the same. In all cases we see that not only is the structure of the algorithm similar, but the structure of the modifier is also similar. In fact, as will be evident later on, there a number of such similarities. If such similarities exist then it should be possible to develop a generalised problem, a generalised algorithmic structure, and a set of generalised properties.

Then, to summarise the principal objectives of this research are :

a) To understand what are the model-reality differences.

b) To determine what is the significance of the modifier in the ISOPE algorithms, and how far are the model-reality differences reflected in it.

c) To determine the effect of the augmentation of the performance index on the properties of the algorithm.

d) To determine how to reduce the need for derivatives of the outputs of the real system.

e) To examine the possibilities of generalising the ISOPE algorithms.

Of course the other objective of studying the properties of the algorithms goes without saying, for in order to answer the above questions one has to analyse the properties of the algorithms.

1.5 LAYOUT OF THE THESIS

This thesis consists of ten chapters, which can be divided into four groups. This first group consists of chapters 1 and 2. In chapter 2, the optimising control problem is formulated. Here the problem is formulated for the various ways in which the optimising control problem can be solved :

a) Direct methods

b) Model based methods

c) Adaptive model based methods.

In this chapter, we also present the sufficiency conditions which a model has to satisfy so that the solution of a model based problem can be guaranteed to be the system optimum.

The second group of chapters consists of chapters 3, 4, and 5. In these chapters we describe and study the centralised control structures. In chapter 3, some of the centralised ISOPE algorithms are derived. These algorithms are analysed in chapter 4. In chapter 4, apart from investigating the optimality and convergence properties of these algorithms, an analysis is undertaken to describe the model-reality differences. We also investigate the significance of the modifier on the algorithms, and study the effect of augmenting the performance index by a quadractic penalty function. In chapter 5, a simulation study is undertaken. Here we verify the conclusions of chapters 3 and 4. The simulation is made using two examples, one an abstract example and the other a more realistic example of a fuel gas mixing system which is used by ICI plc.

The third group of chapters consists of chapter 6, 7, and 8. Here the layout is similar to that of the second group. In these chapters the hierarchical decentralised control algorithms are studied. In chapter 6 these algorithms are derived. In chapter 7, the properties of the algorithms are discussed and finally in chapter 8, a simulation study is presented. The simulation study is carried out using abstract examples.

Finally, the fourth group of chapters consists of chapters 9 and 10. In chapter 9, the ISOPE techniques are generalised. The generalisation is made in terms of the problem and algorithmic structure, and a set of generalised convergence conditions is also given. In chapter 10, we discuss the possibilities

of further study, so that the ISOPE techniques can be developed further and refined.

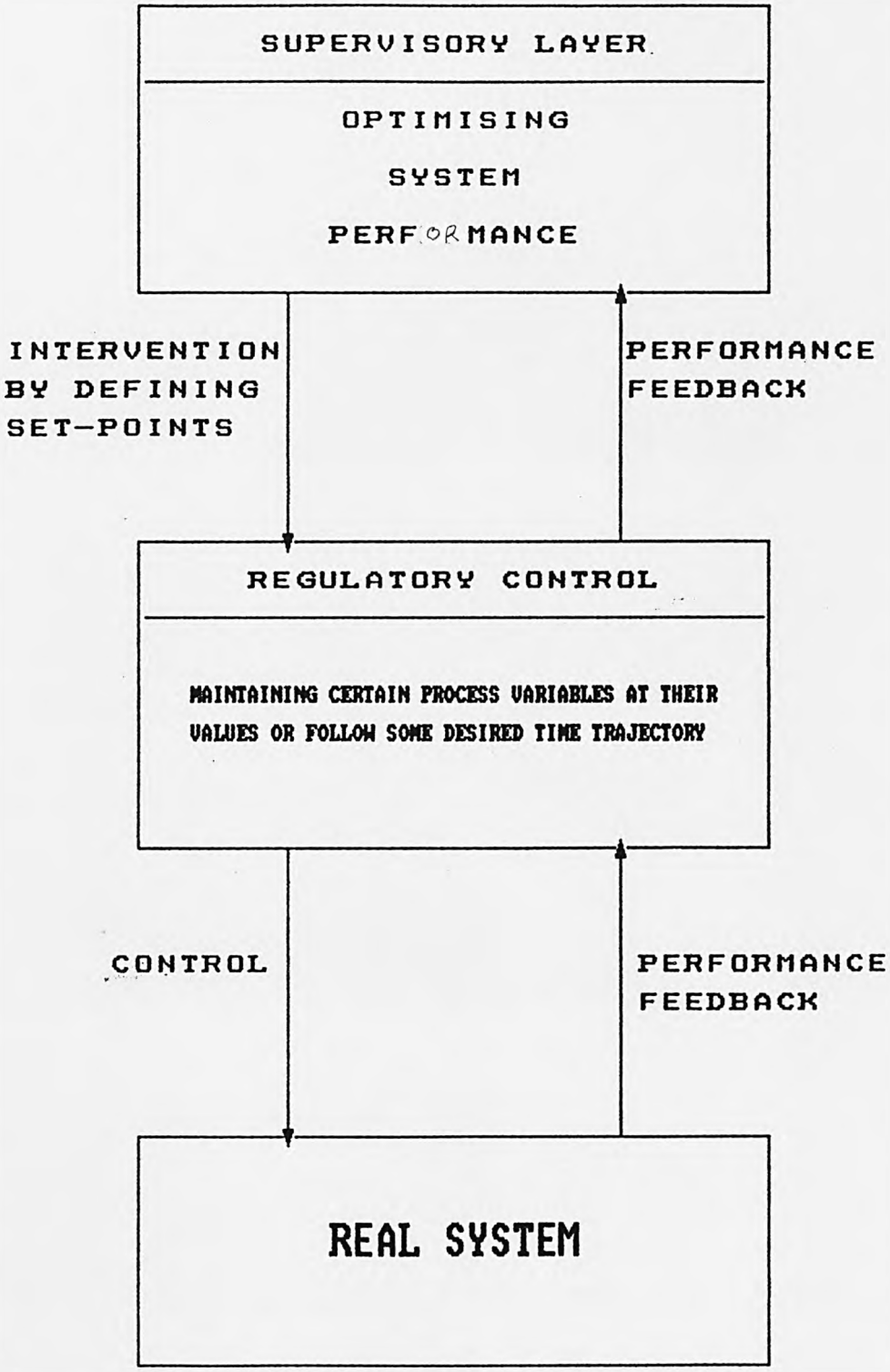


Figure 1.1 Multi-level Model for System Control

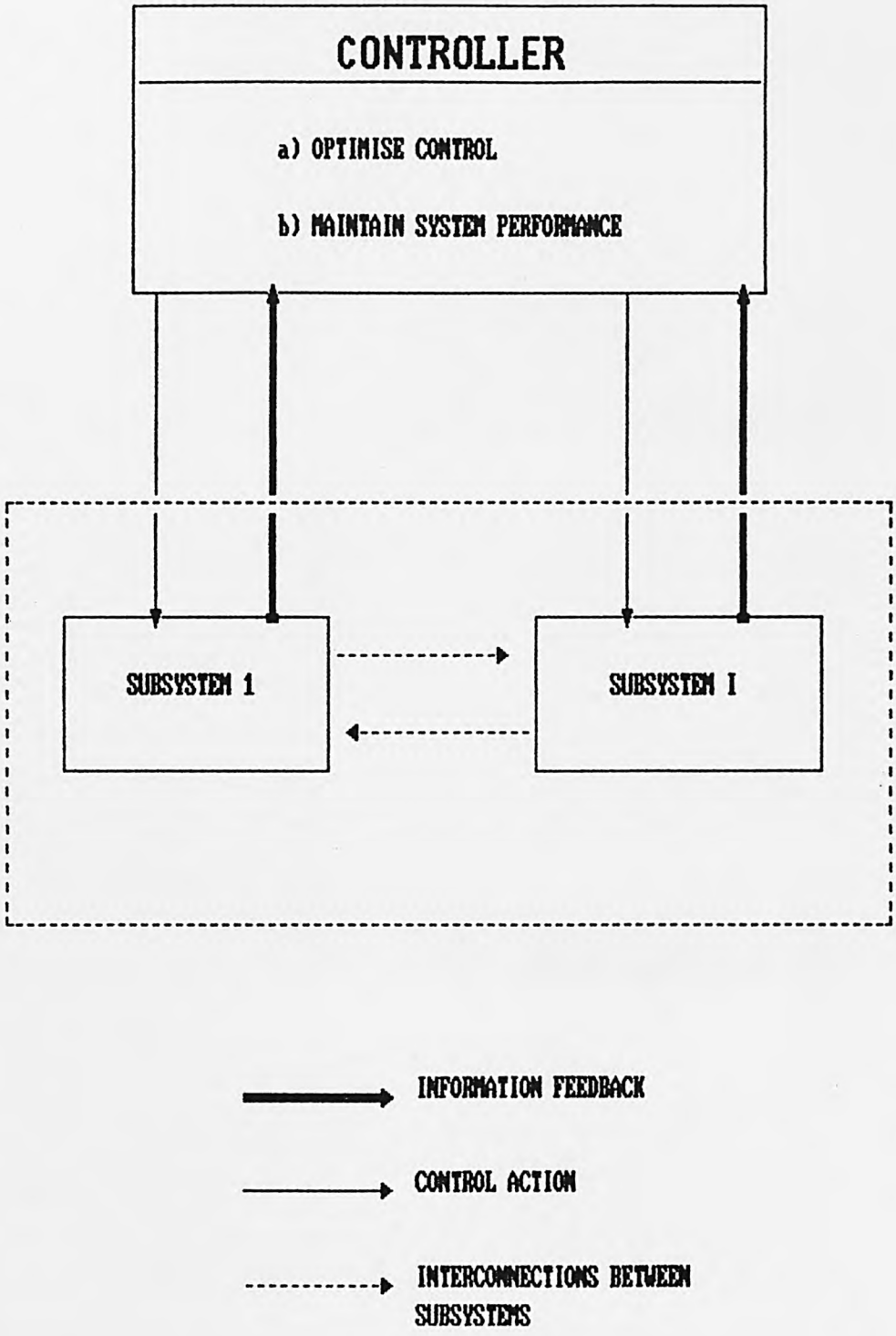


Figure 1.2 Information Exchange and Control Structure in Centralised Control

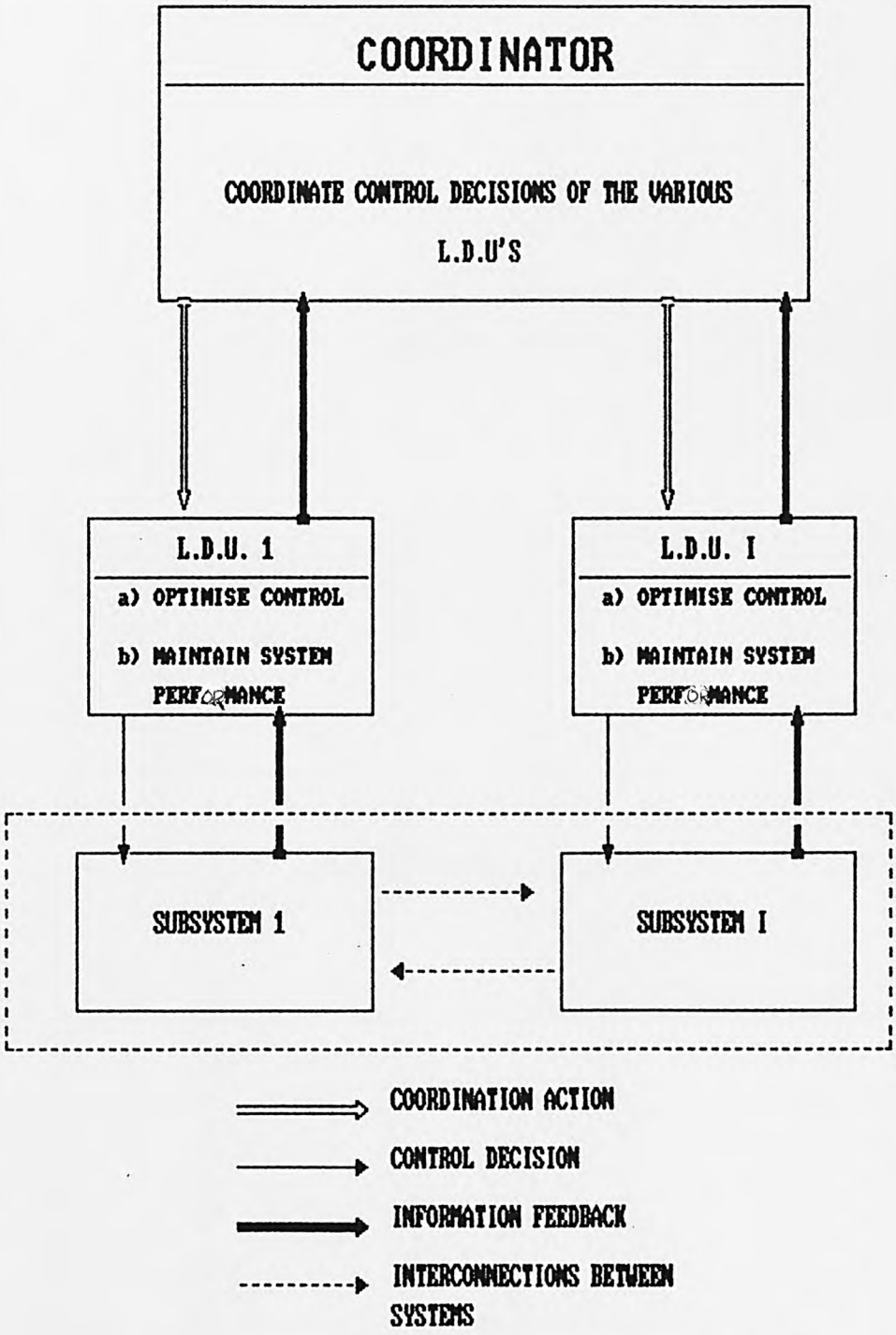


Figure 1.3 Information Exchange and Control Structure in Decentralised Control

CHAPTER 2. THE OPTIMISING CONTROL PROBLEM (OCP)

2.1 INTRODUCTION

As described in chapter 1, the optimising control problem (OCP) can be solved using any of the three methods mentioned, viz:

a) Direct Methods

b) Model Based Methods

c) Adaptive Model Based Methods.

Of these three methods the direct methods and adaptive model based methods are more likely to provide us with the optimum results. The model based methods would provide the optimum only if certain sufficient conditions are fulfilled. These conditions will be discussed in section 2.4. The direct methods, which employ a standard optimisation algorithm and measurements from the system only, are likely to fail to converge when there is noise present and would undoubtedly be inefficient. On the other hand, a purely model based method, besides failing to provide the optimum solution, could prove to be equally inefficient if there are model-reality differences. Adaptive model based methods, on the other hand, are an integration of the direct methods and the model based methods.

Whatever the method one employs to solve the OCP, optimisation is essentially a mathematical procedure. Hence the OCP has to have a mathematical formulation.

In this chapter the OCP will be formulated for a general system consisting of a number of interconnected sub-systems. This is the formulation which is used for decentralised cases, and the problem for the centralised case can be obtained from this by eliminating the interconnections, that is obtaining the global problem. The formulation will be done for both the direct methods and the model based methods. Later these two will be transformed into an adaptive problem along with the associated problem of parameter estimation. The parameter estimation arises due to model-reality differences.

2.2 THE OCP.

We will assume that the controlled system, inclusive of its follow-up controllers, is described in a decomposed way by a set of sub-system input-output mappings (figure 2.1) :

$$F_{*i} : \Re^{ni} \times \Re^{mi} \rightarrow \Re^{mi} ; \quad i \in 1,N$$

$$Y_i = F_{*i}(C_i,U_i)$$

where N is the number of sub-systems

$C_i \in \Re^{ni}$ is the control vector

$U_i \in \Re^{mi}$ is the interaction input vector

$Y_i \in \Re^{mi}$ is the interaction output vector

The sub-systems are interconnected with assumed structure;

$$U_i = H_i Y = \sum_{j=1}^{N} H_{ij}\, Y_j \ , \ i \in 1,N$$

where H_i and H_{ij} are the interconnection matrices.

Globally we can write them as

$$\left.\begin{aligned} Y &= F_*(C,U) \\ U &= HY \end{aligned}\right\} \qquad (2.1)$$

If further we assume that for a given set of controls the set of equations (2.1) have one unique solution, then we can represent the system as a mapping:

$$K_* : \Re^n \rightarrow \Re^m$$

$$Y = K_*(C) \qquad (2.2)$$

The performance of each sub-system is evaluated on basis of a performance index :

$$\tilde{Q}_i : \Re^{ni} \times \Re^{mi} \times \Re^{mi} \ , \ i \in 1,N$$

$$Q_i = \tilde{Q}_i(C_i,U_i,Y_i)$$

and of course subject to a set of constraints

$$G_{ij}(C_i,U_i,Y_i) \leq 0 \quad i \in 1,N \ ; \ j = 1,2,3,.....$$

One of the most important assumptions we make here is that the global performance index is "Additively Separable" **(Singh and Titli (1978), Findeisen et al (1980))** , and hence the global performance index is:

$$Q(C,U,Y) = \sum_{i=1}^{N} \tilde{Q}(C_i,U_i,Y_i) \qquad (2.3)$$

We can now define the Optimising Control Problem (OCP) as:

$$\min_{C,U,Y} \ Q(C,U,Y)$$

s.t

$$\begin{aligned} &Y = F_*(C,U) \\ &U = HY \\ &G(C,U,Y) \leq 0 \end{aligned} \qquad (2.4)$$

In the direct methods, it is this formulation that is used. Of course the mapping F_* is not known and measurements from the system are used.

The formulation of the OCP given by (2.4) is used to solve the problem in a decentralised manner. If however the problem is to be solved in a centralised manner then (2.4) is transformed into:

$$\min_{C,Y} \bar{Q}(C,Y)$$

s.t

$$Y = K_*(C) \tag{2.5}$$

$$\bar{G}(C,Y) \leq 0$$

2.3 MODEL BASED CONTROL PROBLEM (MOCP)

As mentioned earlier the mappings K_* are not known exactly. However, it is possible to develop a set of mathematical relationships; models, to represent the system. These can be represented by the mappings

$$F_i : \Re^{ni} \times \Re^{mi} \quad , i \in 1,N$$

$$Y_i = F_i(C_i,U_i)$$

where Y_i, C_i, U_i are as before.

Globally we can write these as

$$Y = F(C,U)$$

If now we replace the system mappings by the model mappings in the OCP we get the following problem:

$$\min_{C,U,Y} Q(C,U,Y)$$

s.t

$$Y = F(C,U) \tag{2.6}$$

$$U = HY$$

$$G(C,U,Y) \leq 0$$

If as before, we make the assumption that for a given set of controls there is a unique set of outputs, the model can be defined by a mapping:

$$K : \Re^n \rightarrow \Re^m$$

$$Y = K(C)$$

then the centralised model based problem becomes:

$$\min_{C,Y} \bar{Q}(C,Y)$$

s.t

$$Y = K(C) \tag{2.7}$$

$$\bar{G}(C,Y) \leq 0$$

2.4 SUFFICIENT CONDITIONS FOR THE MOCP

In general, the model adopted in the MOCP will not be an accurate description of the system under study. Inaccuracies can be caused due to many inherent deficiences present during the development of the model and often we cannot be aware of the true environmental conditions present. The question then is under what conditions would the MOCP give the optimum of the OCP. Some conditions were given by **Durbeck (1965) and Foord (1974)** and these were later extended by **Ellis and Roberts (1982)**. These conditions, described here are for the overall system (global system) for clearly the situation where there is a number of sub-systems, is still subject to these conditions.

Consider the system optimisation problem given by (2.5). The Lagrangian associated with this problem is defined as:

$$L_*(\,.\,) = \bar{Q}(C,K_*) + \mu^T \bar{G}(C,K_*(C)) \tag{2.8}$$

Then the first order necessary optimality conditions are:

$$\frac{\partial L_*}{\partial C} = \frac{\partial \bar{Q}(C,K_*(C))}{\partial C} + \frac{\partial \bar{Q}(C,K_*(C))}{\partial K_*}\frac{\partial K_*(C)}{\partial C} + \frac{\partial \bar{G}(C,K_*(C))}{\partial C}\mu + \frac{\partial G(C,K_*(C))}{\partial K_*}\frac{\partial K_*(C)}{\partial C}\mu \tag{2.9}$$

$$\mu^T \bar{G}(C,K_*(C)) = 0 \; ; \; \mu = 0 \; \bar{G}(C,K_*(C)) < 0 \tag{2.10}$$

where μ is the Kuhn-Tucker multiplier associated with the constraints G(.).

Also consider the the model based problem given by (2.7). The Lagrangian associated with this problem is:

$$L(\,.\,) = \bar{Q}(C,K(C)) + \eta^T \bar{G}(C,K(C)) \tag{2.11}$$

Then the first order optimality conditions are

$$\frac{\partial L}{\partial C} = \frac{\partial \bar{Q}(C,K(C))}{\partial C} + \frac{\partial \bar{Q}(C,K(C))}{\partial K}\frac{\partial K(C)}{\partial C} + \frac{\partial \bar{G}(C,K(C))}{\partial C}\eta + \frac{\partial \bar{G}(C,K(C))}{\partial K}\frac{\partial K(C)}{\partial C}\eta \tag{2.12}$$

$$\eta^T \bar{G}(C,K(C)) = 0 \; ; \; \eta = 0 \; \bar{G}(C,K(C)) < 0 \tag{2.13}$$

where η is the Kuhn-Tucker multiplier associated with the constraints G().

It is clear, that if the model based optimality conditions given by (2.12) and (2.13) agree with the system optimality conditions (2.9) and (2.10) at the system optimum, then the model based problem will provide the system optimum. This matching is possible if the model outputs match the system outputs and also if the model based derivatives match the system based derivatives at the system optimum.

The sufficient conditions, then for the MOCP to give the optimum of the OCP are:

$$\left.\begin{aligned} K(C) &= K_*(C) \\ \frac{\partial K(C)}{\partial C} &= \frac{\partial K_*(C)}{\partial C} \end{aligned}\right\} \qquad (2.14)$$

at the system optimum.

If the model employed in the MOCP does not satisfy the sufficient conditions of (2.14), then the results obtained cannot be guaranteed to be optimal. Generally, the results will be sub-optimal. Conversely, we can say that the only way to guarantee that the results of MOCP will provide the system optimum is to ensure the satisfaction of the conditions, (2.14). To overcome these difficulties, adaptive model-based strategies are used. In the following section the adaptive model-based problem is formulated.

2.5 ADAPTIVE MODEL BASED PROBLEM (AMOCP)

It has been seen in the previous section that to ensure a model based problem will provide the optimum for the OCP sufficient conditions given by (2.14) should be fulfilled. As it is not always possible for a model to satisfy these conditions and to overcome them adaptive strategies are used. A simple adaptive strategy is to include some free parameters in the model. These parameters are periodically estimated and the model is updated (see figure 2.2). This updated model is then used to solve the problem.

The adaptive model can be represented by a mapping :

$$F_i : \Re^{ni} \times \Re^{mi} \times \Re^{mi} \;,\; i \in 1,N$$

$$Y_i = F_i(C_i, U_i, \alpha_i)$$

where Y_i, C_i, U_i are as before and

$\alpha_i \in \Re^{mi}$ is the parameter estimate vector

Globally we can write these as

$$\left.\begin{aligned} Y &= F(C,U,\alpha) \\ U &= HY \end{aligned}\right\} \qquad (2.15)$$

where H, as before, is the interconnection matrix.

The inclusion of the parameters, α, into the model, leads to an associated problem of parameter estimation. In this thesis a simple parameter estimation procedure is adopted which consists of matching model and system outputs, and the parameter estimation problem is :

$$F(C,U,\alpha) = K_{\mathbf{I}}(C) \dashrightarrow \alpha \tag{2.16}$$

and the AMOCP is then defined as

$$\begin{aligned} &\min_{C,U,Y} \; Q(C,U,Y) \\ \text{s.t} \\ &Y = F(C,U,\alpha) \\ &U = HY \\ &G(C,U,Y) \leq 0 \end{aligned} \tag{2.17}$$

The strategy, then, for adaptive optimisation is to solve problem (2.16) and (2.17) repeatedly until no further improvement in the solution is possible. This type of adaptive strategy is known as a two-step method. More will be said about this later.

It should be stated, however, that the introduction of an adaptive model, still does not ensure the satisfaction of conditions (2.14) and hence does not guarantee the correct system solution.

2.6 CONCLUDING REMARKS

The mathematical formulation of the optimising control has been made. It is seen that the problem deals with the selection of controls (set-points) for a system which satisfy given performance criteria. In practice the determination of controls can be made using measurements directly or by using models. The use of measurements directly involves difficulties such as having to cope with measurements which are contaminated by process noise and also having to decide when the process has reached a steady-state before the measurements can be taken.

The use of system models involves mathematical descriptions of the system concerned, where the results obtained from the model based problem are applied to the system. However, the use of models is still not straight forward as models are likely not to follow the system accurately due to the inherent deficiencies present during the development of the model. The conditions under which it is advantageous to use a purely model-based problem were given in section 2.4.

As it is not always possible to fulfill both these conditions, an adaptive

strategy can be used. In the simple two-step strategy mentioned in section 2.5, only one condition is satisfied. The derivative matching condition is not guaranteed to be satisfied and, as a result, even such a strategy may give sub-optimal results.

To overcome these difficulties, a new adaptive strategy known as Integrated System Optimisation and Parameter Estimation (ISOPE) was proposed by Roberts (1979). It is this algorithm and its various extensions, for both centralised and decentralised control structures, which is the major concern of this thesis.

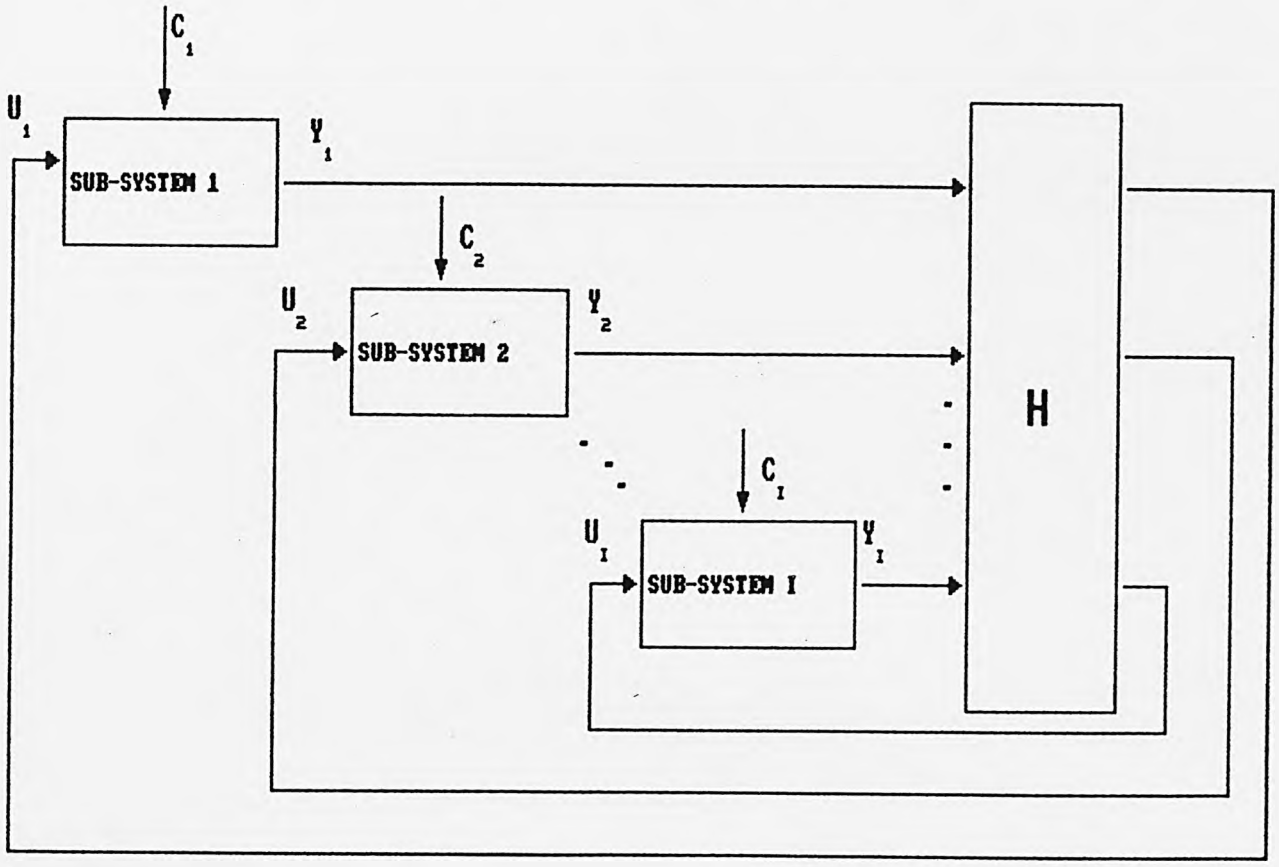


Figure 2.1 A Large Interconnected System

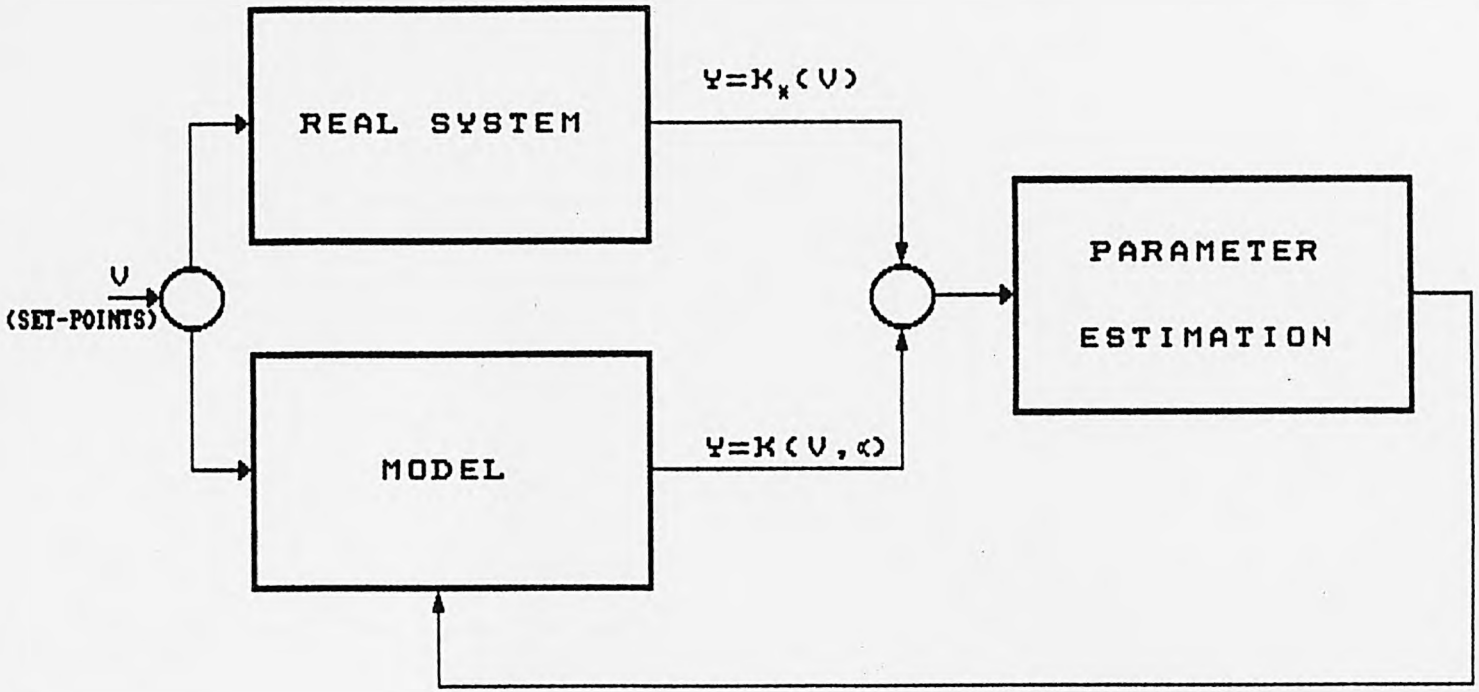


Figure 2.2 Adaptive Model Technique

CHAPTER 3 CENTRALISED CONTROL STRUCTURES

3.1 INTRODUCTION

It has been stated that optimising control deals with the selection of controls for a system which satisfy given conditions; these are the maximisation or minimisation of a given performance index. The control problem then essentially exists in the steady-state, although system dynamics have to be borne in mind, and the controls take the form of set-points. In practice the determination of the controls can be made using models, that is the mathematical descriptions of the system concerned, where the results from the model based problem are applied to the system; or by directly using measurements from the system itself. The use of measurements directly means that every time measurements are required we need to be sure that the steady-state has been attained, and hence the use of steady-state models may be preferable. On the other hand, we require a model to satisfy the sufficient conditions of section 2.4. It is often difficult to satisfy these conditions as there are a number of inherent deficiencies present during the development of the model. Thus, when using a model, we are likely to get sub-optimal system performance.

Thus, there would appear to be a need for adaptive schemes, where the model is updated, through its parameters, taking account of reality conditions. The design of the control system can then be divided into two parts, as shown in figure 3.1. These are :

a) A Model Parameter Estimation Problem

b) A System Optimisation Problem.

Adaptive schemes can be thought of as an integration of the direct methods and the model based methods, In such schemes the optimisation problem is solved using a model where the parameters are periodically updated. Here we see that one of the sufficiency conditions, namely the output matching condition, is fulfilled. In the frequently encountered situation where model-reality differences exist, the problems of parameter estimation and optimisation will interact with each other. The results of the parameter estimation problem – the model parameters – affect the model used in the optimisation problem which, in turn, produces results – set-points – which affect the model parameter estimates. Hence, a number of iterations are required until hopefully convergence is obtained. Such a simple approach is known as the two-step approach, where step 1 is the parameter estimation problem and step 2 is the optimisation problem. This two-step approach

will not, in general, produce optimum results **(Roberts (1979), Ellis and Roberts(1982))** because of ensuring only the satisfaction of the output matching sufficiency condition.

A method intended to decouple these interactions was put forward by **(Roberts (1979))**, where the two problems were integrated into one scheme. Here, a Lagrangian is defined and a set of simultaneous equations are solved repeatedly so that the first-oder optimality conditions are satisfied. The model based problem was modified by a certain parameter, known as the modifier. This procedure was termed the **Modified Two-Step (MSTP)** approach, in the sense that the overall approach remained the same, but the optimisation problem is modified so that the the final solution is optimal. This method comes under the broad class of ISOPE techniques. Since then several studies have been carried out to assess the applicability of this technique **(Roberts (1979), Roberts and Williams (1981), Roberts and Lalui (1982), Ellis and Roberts (1982), Ellis et al (1988), Chen (1986))**. These have demonstrated that the modified two-step approach is superior to the two-step approach in overcoming the model-reality differences and providing optimal solutions.

Another approach which again is an ISOPE technique was put forward by **Sheng and Ellis (1985)**. This approach is a modification of the modified two-step approach and involves system linearization techniques and is known as the **Approximate Linear Model** (ALM-ISOPE) approach. This approach has proved to be successful in providing optimal results as various studies have shown **(Sheng and Ellis (1985), Kambhampati and Ellis (1987), Ellis et al (1986))**.

Apart from these, two other ISOPE techniques have been developed **(Brdys et al (1986b), Tatjewski and Roberts (1987a))** which employ convexification measures. Although convexification of the problem is not needed in general for the centralised cases for convergence to the optimum is always achieved whether the OCP is convex or not **(Brdys and Roberts(1987), Kambhampati and Ellis (1987))**. However, convexification of the problem does lead to increased stability of the iterative process and also speeds it up. Of these, the method given by **Brdys et al(1986b)** retains all the features of the MSTP approach and is known as the **Augmented-ISOPE** (AISOPE) technique. The other, given by **Tatjewski and Roberts (1987a)**, involves more complex mathematical computations and is a Newton-Like algorithm (NL-ISOPE).

In this chapter centralised control structures are described. After decribing the centralised problem, direct methods are discussed. Although, as has been mentioned, such methods are not really suitable for the on-line problem, and

are therefore not dealt with to any extent in the thesis. They are, however, described here for completeness of the approaches to the centralised problem.

Then come the main elements of this chapter : the decription of the modified two-step approach and the approximate linear model approach (ALM-ISOPE). The chapter concludes with a brief presentation of the Augmented ISOPE approach and a Newton-like algorithm.

3.1 THE CENTRALISED CONTROL PROBLEM

The control problem for the centralised case has already been formulated in chapter 2, and is as follows:

$$\min_{C,Y} Q(C,Y)$$

s.t

$$Y = K_*(C) \tag{3.1}$$

$$G(C,Y) \leq 0$$

The aim of the ISOPE methods is to solve (3.1), using the following model based problem:

$$\min_{C,Y} Q(C,Y)$$

s.t

$$Y = F(C,\alpha) \tag{3.2}$$

$$G(C,Y) \leq 0$$

Where the parameters, α, are obtained by solving the following parameter estimation problem:

$$F(C,\alpha) = K_*(C) \rightarrow \alpha \tag{3.3}$$

3.3 DIRECT METHODS

It has already been mentioned, that in the direct methods, measurements from the system are used directly in solving the optimising control problem. For this particular approach one uses the problem as given by 3.1. Here, it can be seen that the mapping K_* is unknown and hence the use of ` measurements. The technique here is to use an optimisation Lagrangian for example the conjugate direction **(Powell (1964))** method in an on-line situation.

3.3.1 Optimisation Methods

There are a number of optimisation algorithms at our disposal **(Fletcher (1981), Gill et al (1981))** which can be used to solve the control problem. These algorithms can be broadly classified as :

a) Gradient (Derivative) based algorithms.

b) Derivative free algorithms.

In general, one would expect the gradient methods to be more effective than the derivative techniques, due to the added information provided. However, in an on–line situation these algorithms are not appropriate. This is because the derivatives are estimated using finite difference approximations and involve perturbing the system and then taking measurements. This whole procedure could be time consuming. Again, the approximations will not be accurate if there is measurement noise present. This means that the optimisation algorithm has to cope with unrepeatable measurements, causing the algorithm to fail to converge.

The derivative free techniques have the advantage that derivatives are not needed by the algorithm. There are a whole range of techniques, derivative free, which one can use in an on-line situation. **Fletcher (1970)**, in his review on function minimisation without evaluating derivatives showed that the method given by **Powell (1964)** and later modified by **Zangwill (1967)**, is one of the most efficient algorithms where derivative information is not used. So, if a direct method was to be used, Powell's method would be the obvious choice and it is this algorithm which is described here.

3.3.2 Powell's Conjugate Direction Method

Most optimisation algorithms, whether gradient based or derivative free, use a direction search approach. In Powell's conjugate direction technique, a search is made for the minimum along conjugate directions.

Consider a quadratic function to be minimised, of type:

$$f(x) = x^T A\, x + B^T x + C \tag{3.4}$$

where $A \in \Re^{n \times n}$ and is positive definite

$$x \in \Re^n$$

and if there is a set of non-zero vectors $\mathcal{G}$ such that :

$$\mathcal{G} = \{ S_i \} \qquad i \in [1,N]$$

covering n-dimensional space, then conjugacy is defined as the property such that

$$S_i^T A S_j = 0 \qquad \forall\ i \neq j \tag{3.5}$$

and S_i and S_j are said to be conjugate directions. Essentially Powell's procedure is one of selecting a set of conjugate directions, $\mathcal{G}$, and searching along each of them until the minimum is obtained.

3.3.2.1 The Algorithm

Each iteration of the procedure commences with a search down n linearly independent directions $E_1, E_2, \cdots, E_n$,starting from the best known approximation to the minimum, x_o. These directions are chosen to be the coordinate directions initially, so that the start of the first iteration is identical to an iteration of the method which changes one parameter at a time. This is later modified to generate conjugate directions by making each iteration define a new direction, E, so that the new set of linearly independent directions are, $E_2, E_3, \cdots, E_n$, E. The way in which E is defined ensures that if a quadratic is being minimised, after k iterations the last k of the n directions chosen for the $(k+1)^{th}$ iteration are mutually conjugate. Thus after n iterations the whole set of directions are mutually conjugate.

An iteration of the procedure is as follows:

Step I : for $r = 1, 2, 3, \cdots, n$ calculate λ_r such that $f(x_{r-1} + \lambda_r E_r)$ is a minimum and define $x_r = x_{r-1} + \lambda_r E_r$

Step II : for $r = 1, 2, 3, \cdots, n-1$ replace E_r by E_{r+1}

Step III : replace E_n by $(x_n - x_o)$.

Step IV : Choose λ so that $f(x_n + \lambda E_n)$ is a minimum and replace x_o by $x_n + \lambda E_n$

3.4 THE MODIFIED TWO-STEP APPROACH (MSTP)

Adaptive model based approaches are an integration of the direct and model based techniques. The disadvantage with the direct technique is that every time the performance index has to be evaluated, measurements from the system are needed. This could be time consuming and often in the presence of noise achieving convergence can be difficult. A purely model based approach will not produce the true optimum unless the model satisfies the sufficiency conditions given in section 2.4. Even after the introduction of an adaptive parameter estimation procedure to give the simple two-step approach, there can be no guarantee of producing optimal results, because the model is unlikely to satisfy the derivative matching condition (see 2.14 of section 2.4). In order to overcome these problems, **Roberts (1979)** proposed a technique, which would produce the optimal solution. The approach is known as the Modified Two-Step (MSTP) approach, in the sense that the overall

approach remains the same as that of the two-step approach but the optimisation problem is modified so that the final result is optimal.

3.4.1 The MSTP Problem

The modified two-step approach uses a problem which is an integration of the two problems of optimisation ((3.2)) and parameter estimation ((3.3)), to obtain the solution of (3.1). The integrated problem is thus:

$$\min_{C,\alpha} \; q(C,\alpha)$$

s.t

$$F(C,\alpha) = K_*(C) \tag{3.6}$$

$$g(C,\alpha) \leq 0$$

where $g(C,\alpha) = G(C,F(C,\alpha))$ and $q(C,\alpha) = Q(C,F(C,\alpha))$

The key step in deriving the algorithm is to integrate the parameter estimation problem and the the optimisation problem, and also introduce a new set of variables, V, into the problem which are defined as

$$V \in \Re^{n_c}, \; V = C \tag{3.7}$$

Now the variables C in the parameter estimation problem are replaced by the new variables V, and the equivalent integrated problem becomes :

$$\min_{C,V,\alpha} \; q(C,\alpha)$$

s.t

$$F(V,\alpha) = K_*(V) \tag{3.8}$$

$$g(C,\alpha) \leq 0$$

$$V = C$$

We now define the Lagrangian, $L(C,V,\alpha,\lambda,\beta,\mu)$, associated with the above problem as :

$$L(.) = q(C,\alpha) + \beta^T(F(V,\alpha) - K_*(V)) + \lambda^T(V - C) + \mu^T g(C,\alpha) \tag{3.9}$$

where β and λ are the Lagrangian multipliers associated with the constraints $F(V,\alpha)=K_*(V)$ and $V=C$ respectively

and μ is the Kuhn-Tucker multiplier associated with the inequality $g(C,\alpha)$

The first order necessary optimality conditions for the existence of a stationary point of (3.9) are :

$$\frac{\partial L}{\partial C} = \frac{\partial q}{\partial C} - \lambda + \frac{\partial^T g}{\partial C}\mu = 0 \tag{3.10}$$

$$\frac{\partial L}{\partial V} = \left[\frac{\partial F}{\partial V} - \frac{\partial K_*}{\partial C}\right]^T \beta + \lambda = 0 \tag{3.11}$$

$$\frac{\partial L}{\partial \alpha} = \frac{\partial q}{\partial \alpha} + \frac{\partial^T F}{\partial \alpha}\beta + \frac{\partial^T g}{\partial \alpha}\mu = 0 \tag{3.12}$$

$$\mu \geq 0 \; ; \; g(C,\alpha) \; = 0 \tag{3.13}$$

$$F(V,\alpha) = K_*(V) \tag{3.14}$$

$$V = C \tag{3.15}$$

If we now solve for λ using (3.11) and (3.12) we get the following

$$\lambda = \left[F'_V(V,\alpha) - K'_*(V)\right]^T \left[F'_\alpha(V,\alpha)^T\right]^{-1} \left[q'_\alpha(C,\alpha) \; + g'_\alpha(C,\alpha)^T \mu\right] \tag{3.16}$$

It should be noted here, that from now on we shall use the following notation to represent the partial derivatives:

$F'_V(V,\alpha) = \frac{\partial F(V,\alpha)}{\partial V}$ and so is the case with the others.

Now for a given C, V, μ, α we can evaluate λ such that conditions (3.11) and (3.12) are satisfied. The condition (3.10) can be satisfied by solving the following minimisation problem:

$$\min_C \; \{q(C,\alpha) - \lambda^T C\}$$

s.t (3.17)

$$g(C,\alpha) \leq 0$$

The condition (3.13) is automatically satisfied by the above. The other conditions, namely (3.14) and (3.15) are also similarly satisfied.

It can be seen, from the above analysis, that we now have an equivalent problem which is composed of three problems which are :

<u>The Parameter Estimation Problem</u>

$$F(V,\alpha) = K_*(V) \dashrightarrow \alpha \tag{3.18}$$

<u>The Modified Optimisation Problem</u>

$$\min_C \; \{q(C,\alpha) - \lambda^T C\}$$

s.t (3.19)

$$g(C,\alpha) \leq 0$$

where λ is as given by (3.16)

The Coordination Problem

$$V = C \tag{3.20}$$

Evaluate the modifier λ

The resultant control structure is shown in figure 3.2. From the manner in which the problem and the structure have been derived, the apparent conclusion is that the structure is hierarchical. However, the whole algorithm can be viewed in an iterative fashion, which is the true interpretation. This is achieved by rationalising the number of variables in the problem **(Ellis et al (1988))**. This is done by eliminating the variables V. Then the Lagrangian

$$L(.) = q(C,\alpha) + \beta^T(F(C,\alpha) - K_*(C)) + \mu^T g(C,\alpha) \tag{3.21}$$

and the necessary optimality conditions for the existence of a saddle point of the above Lagrangian ; are, as before

$$\frac{\partial L}{\partial C} = \frac{\partial q}{\partial C} + \frac{\partial^T g}{\partial C}\mu + \left[\frac{\partial F}{\partial V} - \frac{\partial K_*}{\partial C}\right]^T \beta \tag{3.22}$$

$$\frac{\partial L}{\partial \alpha} = \frac{\partial q}{\partial \alpha} + \frac{\partial^T F}{\partial \alpha}\beta + \frac{\partial^T g}{\partial \alpha}\mu \tag{3.23}$$

$$\mu > 0 \; ; \; g(C,\alpha) = 0 \tag{3.24}$$

$$F(C,\alpha) = K_*(C) \tag{3.25}$$

It can be seen that condition (3.22), using (3.23), can be reduced to the following

$$\frac{\partial L}{\partial C} = \frac{\partial q}{\partial C} + \frac{\partial^T g}{\partial C}\mu - \lambda \tag{3.26}$$

where λ is given by the following

$$\lambda = \left[F'_c(C,\alpha) - K'_*(C)\right]^T \left[F'_\alpha(C,\alpha)^T\right]^{-1} \left[q'_\alpha(C,\alpha) + g'_\alpha(C,\alpha)\mu\right] \tag{3.27}$$

From (3.22) to (3.25) it can be seen, as before, that we have a problem which is composed of the following three components:

The Parameter Estimation Problem

$$F(C,\alpha) = K_*(C) \dashrightarrow \alpha \tag{3.28}$$

The Modified Optimisation Problem

$$\min_C \; \{q(C,\alpha) - \lambda^T C\}$$

s.t (3.29)

$$g(C,\alpha) \leq 0$$

where λ is as given by (3.27)

Notice here that the general structure of the problem is similar to that of the simple two-step approach. In fact, the way the MSTP algorithm functions can be better understood using the above formulation. For the algorithm is basically iterative in nature, and in general when considering centralised systems we see that there is no need for any form of coordination as would be required when the system is decomposed into individual sub-systems. Then, for centralised control the term coordinator is misleading as there is no coordinating function involved in the control structure and does not give a true picture of the iterative process. This rationalised form of the control structure, was given by **Ellis et al (1988)**, and is shown in figure 3.3. However, for the sake of clarity, and for the sake of ease in identifying the on-line and the off-line part of the algorithm, we shall be using the former formulation throughout the thesis.

Whether we take the original form or the rationalised version of the problem structure, it is seen that the resultant structure of the algorithm is the same. Also, due to the modifications introduced into the problem the model need not satisfy the second sufficient condition of derivative matching **(Brdys and Roberts (1987), Chen (1986), See chapter 4 of this thesis)**. This is done by the modifying term λ, which is introduced into the optimisation problem. In fact, it is this term which ensures that an application of this algorithm to a problem yields the true optimum. It will be shown in chapter 4, that, the modifier can be interpreted as following

a) A parameter which quantifies the violations of the sufficiency conditions by the model.

or as

b) A parameter which compensates for the differences in the properties of the model based performance index and the system based performance index.

Whatever interpretation we have given the modifier ultimately, as shall be seen in Chapter 4, they mean the same.

3.4.2 The Algorithmic Structure

We now give the algorithmic structure for the control structure discussed above. Before that let us define the following:

$$\mathcal{C} \triangleq \{ C : g(C,\alpha) \leq 0 \}$$

$$\hat{C} \triangleq \arg \min_{C \in \mathcal{C}} \{ q(C,\alpha) - \lambda^T C \} \qquad (3.30)$$

The algorithmic structure is then as follows:

STEP 0 : Set k = 0

Select initial controls C^o and $\varsigma > 0$ the convergence criteria and $\xi \in \Re^*$, $0 < \xi \leq 1$ where ξ is known as the iterative loop gain

IF the old formulation is used then

set $V^o = C^o$

STEP I

STEP Ia : Apply controls to the system and take measurements to evaluate the parameters α

STEP Ib : Apply further perturbations to the system and take measurements to get a finite difference approximation for K'_*

STEP Ic : Evaluate the modifier λ

STEP II : Solve the modified optimisation problem to obtain $\hat{C}^k$

STEP III : IF the old formulation is used then check

IF $\|V^k - \hat{C}^k\| < \varsigma$ then STOP

else check

IF $\|C^k - \hat{C}^k\| < \varsigma$ then STOP

else continue

STEP IV : IF the old formulation is used then

set $V^{k+1} = V^k + \xi (\hat{C}^k - V^k)$ (3.31)

else

set $C^{k+1} = C^k + \xi (\hat{C}^k - C^k)$ (3.32)

set k ← k+1

GO TO STEP I

Notice the updating scheme, (3.31) or (3.32), used for updating the control set-points. Here the controls are not updated as per the equation $V=\hat{C}$ or $C=\hat{C}$ instead we use the above schemes. This scheme is known as the under-relaxation scheme **(Cohen (1978))**. The idea here is that instead of changing the controls in one go we change the values in a small step. Then the iterative loop gains, ξ, effectively define the step length. The under-relaxation scheme is used so that an element of stability is introduced into the iterative process, for a direct application of the controls obtained from the optimisation problem could have either or both of the following effects:

a) The algorithm may move away from the optimum

b) The algorithm may start to oscillate.

3.5 THE APPROXIMATE MODEL APPROACH (ALM-ISOPE)

The MSTP approach requires the derivatives of the system to evaluate the modifier, so that the modified optimisation problem can be solved. It is seen that the modifier is a term which quantifies the violations of the sufficient conditions by the model. The modifier exists because of the differences in the system and its model. If, however, we use a model, which is linear and of type

$$F(C,\alpha) = WC + \alpha \tag{3.33}$$

where $W \in \Re^{m\times m}$ and $\alpha \in \Re^{m}$

then for such a model to satisfy the sufficiency conditions we see that

$$F'_c(C,\alpha) = W = K'_*(C) \tag{3.34}$$

$$\alpha = K_*(C) - WC \tag{3.35}$$

Now in the modified two-step approach we need the derivatives $K'_*(C)$ for evaluating the modifier λ. However, by using a model of type given by (3.33) and set W as given by (3.34) we obtain a linear model which satisfies the sufficient conditions. This means that in such a case the modifiers, λ, are zero and there is no need to evaluate them and, consequently, the optimisation problem need not be modified.

3.5.1 The ALM-ISOPE Problem

If the approach described above is to be used, we see that the control problem now consists of two parts, as before, and we can refer to these parts as:

a) Model identification

b) Optimisation

Here, model identification replaces the parameter estimation of the modified two-step approach. Of course one could still use the term parameter estimation as we still identify parameters W and α. The difference now is that we develop a completely new model at each iteration. Hence using the term model identification is more appropriate.

So the two problems are:

Model Identification

$$F'_v(V,\alpha) = W = K'_*(V) \tag{3.36}$$

$$\alpha = K_*(V) - WV \tag{3.37}$$

Optimisation Problem

$$\min_{C} \; Q(C,Y)$$

s.t (3.38)

$$Y = WC + \alpha$$

$$G(C,Y) \leq 0$$

3.5.2 The Algorithmic Structure

The control structure for ALM-ISOPE is shown in figure 3.4. The algorithmic structure for ALM-ISOPE remains the same as that given in section 3.4.2. The only difference being that STEPI is now replaced by the following:

STEP I

STEP Ia :Perform perturbations about V^k to get a finite difference approximation for $K'_*(V^k)$ and set it to W

STEP Ib : Evaluate $\alpha = K_*(V^k) - WV^k$

STEP Ic : Set the new model to

$$Y = WC + \alpha$$

All the other steps remain the same.

3.5.3 Interpretation of the Structure

We have seen that the MSTP algorithm introduces a modifier, λ, into the optimisation problem. The modifier quantifies the violations of the sufficient conditions by the model. On the other hand, it has been seen that the sufficient conditions can be used directly to develop a linear model. This model satisfies both of the sufficient conditions at each iteration and hence there is no need to have a term which quantifies the violations.

One can, of course, look at the approach from a different point of view. It can be seen that the ALM-ISOPE tries to linearise the system about a given point. Consider the Taylor series expansion of a function $K_*(V)$ about a point V^k :

$$K_*(V) = K_*(V^k) + K'_*(V^k)\, V^k + \| O \| \tag{3.39}$$

where $\| O \|$ denotes the higher order terms of the expansion.

If now we set

$$\alpha = K_*(V^k) + \| O \| \tag{3.40}$$

we get the expression

$$K_{\#}(V) = K'_{\#}(V^k)\, V^k + \alpha \qquad (3.41)$$

which then is the same as what we are doing in the ALM-ISOPE. Thus near the point V^k the model follows the system in a linear manner and at each iteration we satisfy both the sufficient conditions and at the optimum these are also satisfied.

3.6 OTHER STRUCTURES

It has been shown that the MSTP algorithm will always converge to the true optimum **(Brdys and Roberts(1987))**, whether the OCP is convex or not as long as the model based problem is convex. This is the main advantage with the the MSTP algorithm. However, when large model-reality differences exist, the convergence rate of the algorithm is slow. Also, the iterative loop gains used in the relaxation scheme are found out through a trial and error procedure. This could be very troublesome and a given iterative loop gain could be either very large or to small. A similar problem occurs with the ALM-ISOPE for high non-linear systems. However, with the ALM-ISOPE algorithm there is definitely an advantage in that we do not have to provide the algorithm with a model apriori. Also the model developed by the algorithm would be a better representation of reality at that point and as a result we find that we can use high iterative loop gains with this technique. However, it is possible that with highly non-linear systems, this technique could have the same disadvantages as the MSTP algorithm. To overcome these difficulties two other algorithms have been developed. Both these are an extension of the MSTP algorithm and use convexification techniques, where the performance index is augmented with a quadratic penalty function.

One algorithm known as the Augmented ISOPE (AISOPE) algorithm, was proposed by **Brdys et al(1986b)**. Here all the features of the MSTP algorithm were maintained. The extra feature is that an additional quadratic penalty function was appended to the performance index.

The other algorithm developed by **Tatjewski and Roberts (1987a)** uses the same formulation as AISOPE. Here, however, a major change has been made in the manner in which the control set-points are updated. Instead of using an under-relaxation scheme, a precise Newton-like step is taken, and there is no need to select the iterative loop gains. This has an obvious advantage for the convergence of the procedure.

3.6.1 The Augmented ISOPE (AISOPE)

This algorithm retains all the features of the MSTP algorithm. The additional feature of the algorithm is that the performance index is augmented by

a quadratic penalty function. The analysis for the derivation is the same as that for the MSTP, and hence it is not discussed in detail. The parameter estimation problem remains the same as in MSTP and there is no change is the formula to evaluate the modifier λ. The obvious change occurs in the MBOP, which now becomes:

$$\min_{C} \left\{ q(C,\alpha) - \lambda^T C + \rho \| V - C \|^2 \right\}$$

s.t (3.42)

$$g(C,\alpha) \leq 0$$

where ρ is the penalty factor.

The only benefit one derives from such a formulation is that due to the augmentation the convergence could be faster.

3.6.2. The Newton-Like Algorithm (NL-ISOPE)

It has been mentioned that one of the problems with the algorithms discussed so far is the selection of the iterative loop gains. These gains are arrived at after some trials, and this in an on-line situation could pose some difficulties. The NL-ISOPE algorithm **(Tatjewski and Roberts (1987a))** overcomes this problem by using a precise Newton-like step. The overall problem formulation remains the same as in the AISOPE. The difference is in the manner in which we update the controls.

The scheme for updating controls can be seen as solving the following operator equation:

$$\Psi(V) = V - \hat{C}(V) = 0 \tag{3.43}$$

the solution for the above equation is then as follows

$$\left[I - \hat{C}'(V^k) \right] \left[V^{k+1} - V^k \right] = \left[\hat{C}^{k+1} - V^k \right] \tag{3.44}$$

where

I is an Identity matrix of appropriate dimensions

$$\hat{C}'(V^k) = E \left\{ \rho I + F'(C^k,\alpha) P'' F(C^k,\alpha) \left[F'(V^k,\alpha) - K'_*(V^k) \right] + \left[F'(V^k,\alpha) - K'_*(V^k) \right] P'' F(V^k,\alpha) K'_*(V^k) \right\} \tag{3.45}$$

where

$$E = A^{-1} - A^{-1}(g'_a)^T \left[g'_a A^{-1} g'_a{}^T \right]^{-1} g'_a A^{-1} \tag{3.46}$$

$$A = L''_{cc}(C,\alpha,\mu) + \rho I \tag{3.47}$$

$$L(C,\alpha,\mu) = q(C,\alpha) + \mu^T g(C,U) \qquad (3.48)$$

g_a = The set of constraints which are active

ρ = penalty factor

A detailed derivation of this method is given in Tatjewski and Roberts (1987).

Looking at the formulation one immediately recognises the complexity of the process. This complexity in calculation is compensated for by the fact that the method converges faster.

3.7 CONCLUDING REMARKS

Five different approaches to solve the centralised optimisation problem have been discussed. Of these, the first technique is a direct method and the remainder are adaptive techniques which use models of the system. The direct methods have disadvantages (see chapter 5) and there is a definite advantage in using the adaptive techniques.

Most of the adaptive techniques discussed are extensions of the original MSTP algorithm . Each one tries to improve the performance of the MSTP while retaining the features of this algorithm. Of these the ALM-ISOPE is the algorithm which deviates from the original algorithm in an important way. This is because here a model need not be supplied to the algorithm a priori. Again, as shall be seen in chapter 5, this algorithm functions very well in that very high iterative loop gains can be used, as the model developed by the algorithm is a better representation of the process at the current operating point. This model changes from iteration to iteration.

The AISOPE technique improves upon the rate of convergence of the MSTP. However, this method increases the number of parameters which we have to select. In addition to the iterative loop gains, we have to select appropriate penalty factors. A very high penalty factor would ill condition the problem and a very low factor would not produce any improvement to the iterative process.

The NL-ISOPE algorithm improves on the AISOPE by increasing the off-line calculations. It has been found **(Tatjewski and Roberts (1987a))** that this algorithm has good convergence properties.

However, in this thesis we shall be discussing only four of these methods in detail – the direct method, the MSTP, ALM-ISOPE and the AISOPE algorithms – for the NL-ISOPE algorithm was developed during the time of writing this thesis. In chapter 4 the conditions for convergence of the adaptive algorithms will be

discussed. In chapter 5 a detailed simulation is presented. An examination of what really constitutes model-reality differences is made. It shall be seen that these differences are in agreement with the analysis presented in chapter 4.

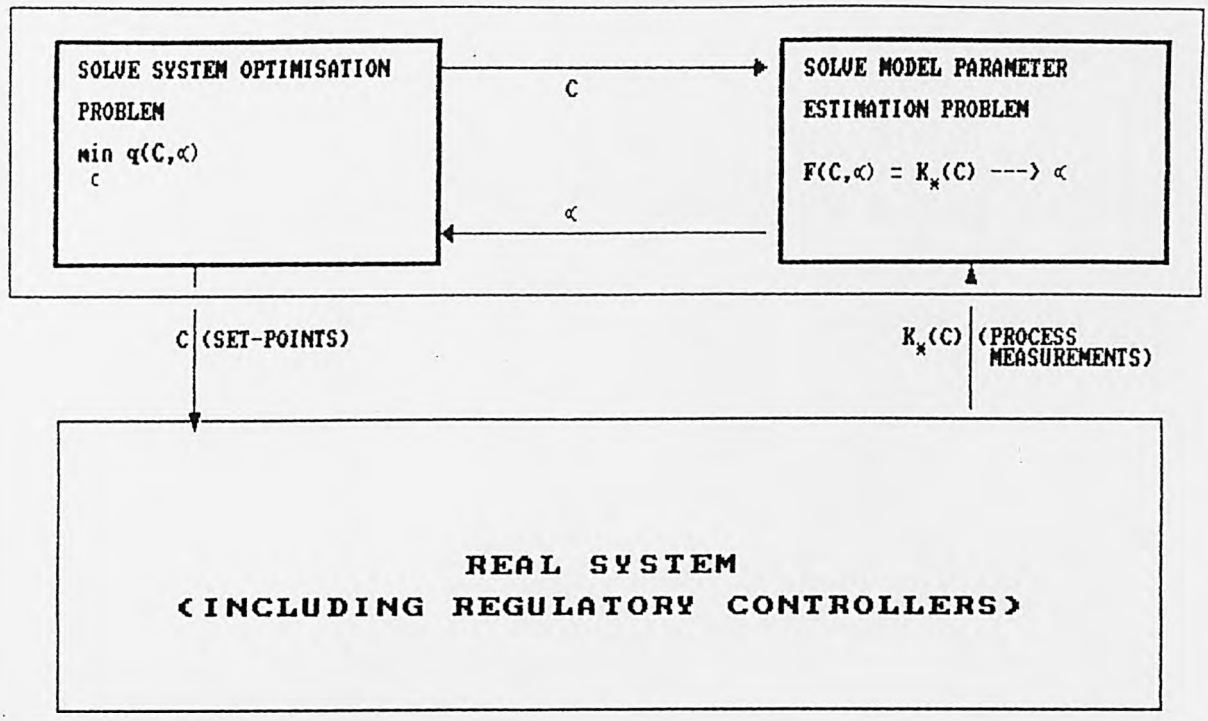


Figure 3.1 The Two Problems of System Optimisation and Parameter Estimation

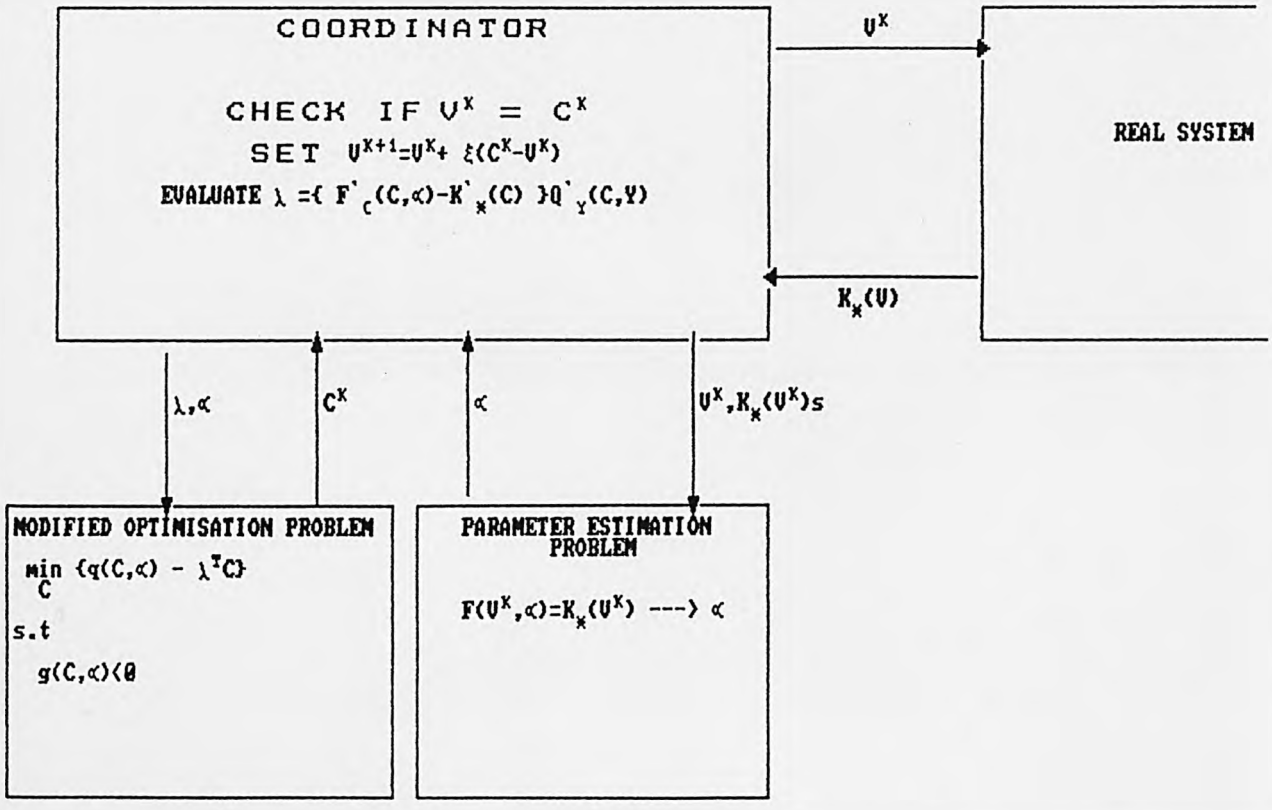


Figure 3.2 The MSTP Structure as Proposed by Roberts

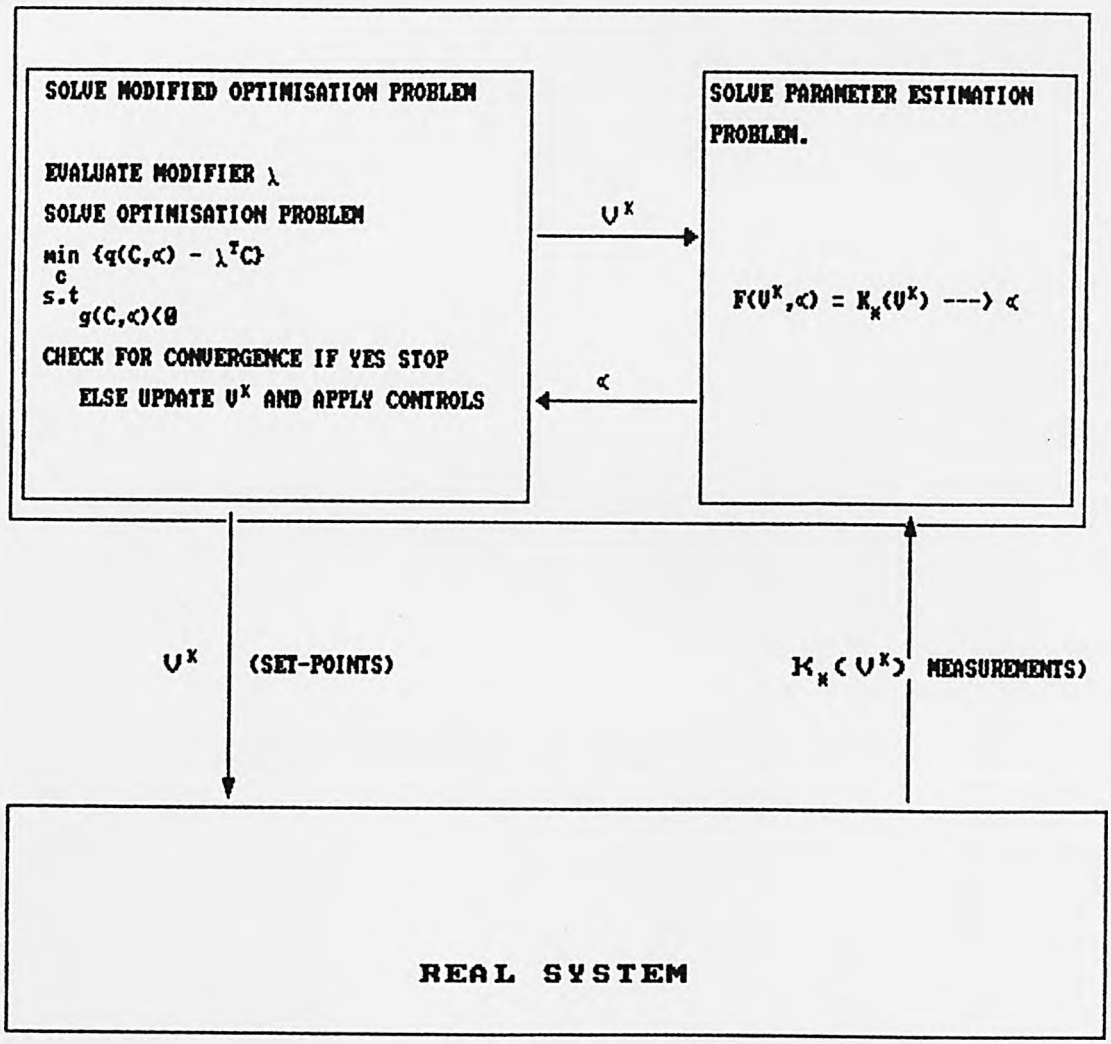


Figure 3.3 The MSTP Structure in an Algorithmic Framework

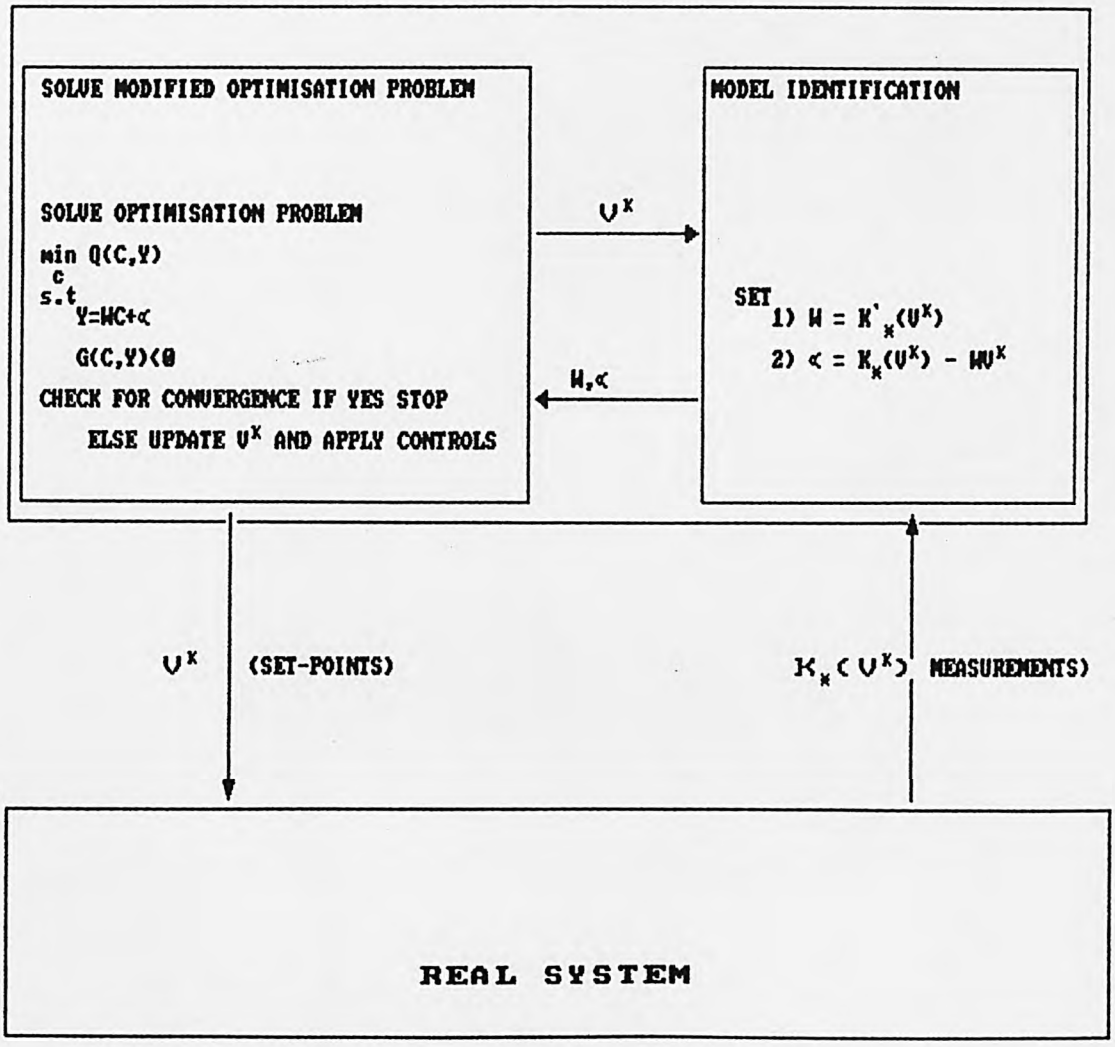


Figure 3.4 The ALM-ISOPE Structure

CHAPTER 4 OPTIMALITY AND CONVERGENCE OF THE CENTERALISED CONTROL STRUCTURES

4.1 INTRODUCTION

In chapter 3 a number of schemes for solving the centralised optimising control problem have been discussed. In this chapter the optimality convergence properties of these algorithms will be presented. A detailed study will be given for the MSTP and ALM-ISOPE algorithms. The analysis for AISOPE is similar to that of the MSTP algorithm. The only difference is that one has to take into account the augmentations done in the AISOPE algorithm. The analysis for the NL-ISOPE algorithm is different from that of the others and can be seen in **Tatjewski and Roberts (1987)**.

Of course, the properties of the MSTP algorithm have been studied by **Brdys and Roberts(1987)**. In their study there was a discussion on the type of model to be used, but this was only in terms of a linear or non-linear model. Here we extend those concepts to study the model structure and to evaluate what really constitute model-reality differences.

An analysis similar to that for MSTP, given by Brdys and Roberts, has been given for AISOPE by **Brdys, et al(1986b)**. Here a detailed analysis of this algorithm is not undertaken. However, a set of conditions for convergence is presented to highlight the effect of the augmentations on the algorithm.

It shall be seen seen that the derivative differences, used in evaluating the modifier, λ, really constitute the model-reality differences. Then a logical extension of this idea is the ALM-ISOPE algorithm, where the algorithm develops a linear model by using the derivatives of the system.

One can conclude that the algorithms which can be classified as ISOPE algorithms are all optimal.This is because of the manner in which thay are derived. In fact these algorithms solve repeatedly a set of simultaneous equations so that the first order necessary optimality conditions for the problem are satisfied. However, for the sake of simplifying further analysis, a brief description of the optimality of these algorithms is undertaken.

4.2 OPTIMALITY

Before proceeding to look into the optimality[1] conditions for the algorithms some definitions are necessary.Let us define the algorithmic solution set Ω as

[1] *By optimality of algorithm, we are referring to a local optimum. For practical purposes this will be a local minimum.*

$$\Omega \triangleq \{ (V) : V \in \mathcal{C} , V = \hat{C}(V,\alpha) \} \tag{4.1}$$

This definition of the solution set is the same for MSTP, AISOPE, NL-ISOPE algorithms. However for the ALM-ISOPE algorithm it is

$$\Omega \triangleq \{ (V) : V \in \mathcal{C} : V = \hat{C}(V) \}$$

$$\mathcal{C} \triangleq \{C : g(C) \leq 0 \}$$

For further analysis the modifier λ has to be simplified further. For simplicity in analysis let us assume that the constraints are independent of the outputs. We have

$$\lambda = [\, F'_C(V,\alpha) - K'_*(V) \,]^T \, Q'_Y(C,F(C,\alpha)) \tag{4.2}$$

$$= Q'_Y(C,F(C,\alpha))F'_C(V,\alpha)^T - Q'_Y(C,F(C,\alpha))K'_*(V)^T$$

now since $V=C$ and $F(C,\alpha)=K_*(C)$ we get from above

$$\lambda = Q'_C(V,F(V,\alpha)) + Q'_Y(V,F(V,\alpha))F'_C(V,\alpha)^T - Q'_C(V,K_*(V)) - Q'_Y(V,K_*(V))K'_*(V)^T$$

$$\Rightarrow \quad \lambda = q'_C(V,\alpha) - q'_*(V) \tag{4.3}$$

Having defined the solution set Ω and simplified the form of the modifier, λ, we can now proceed to give the optimality conditions for the algorithms. The following lemma gives these conditions and is common to all algorithms.

Lemma 4.21

Assume that each point of the set $\mathcal{C}$ satisfies the regularity conditions. Then the point $\bar{V} \in \Omega$ satisfies the Kuhn-Tucker necessary conditions for the OCP.

Proof

The MSTP Algorithm

Since $C \in \mathcal{C}$ we have :

$$\left.\begin{aligned} &q'_C(C,\alpha) - \lambda + g'_C(C,\alpha)^T\eta = 0 \\ &\eta^T g(C,\alpha) = 0 \; ; \; \eta \rangle 0 \; , g(C,\alpha) = 0 \end{aligned}\right\} \tag{4.4}$$

Now, since $\bar{V} = C$ and $F(V,\alpha) = K_*(V)$ and using (4.3) in above we get

$$q'_*(V) + g'_*(V)^T\eta = 0$$

$$\eta^T g_*(V) = 0 \; ; \; \eta > 0 \; , g_*(V) = 0 \qquad \} \tag{4.5}$$

where $q(C,\alpha) \triangleq Q(C,F(C,\alpha))$ and $g(C,\alpha) \triangleq G(C,F(C,\alpha))$ and η are the Kuhn-Tucker multipliers corresponding to the constraints.

It is then clear that (4.5) are the Kuhn-tucker conditions for the OCP so that $\bar{V}$ is the optimum point of the OCP.

The proof for the other algorithms follows the same principles and is not given.

4.3 CONVERGENCE CONDITIONS

The idea behind the study of convergence properties of algorithms, is to get more information regarding the behaviour of the algorithms. That is, under what conditions would a given algorithm be the most efficient. In other words we study what makes an algorithm work as desired and converge to the optimum.

There are a number of techniques which one can use to develop conditions under which an algorithm will converge. One can use some of the Zangwill convergence theorems **(Zangwill (1969))** or use the contraction mapping theorems **(Ortega and Rheinboldt(1980))** or else use some of the techniques of Cohen **(Cohen (1978))**. The strictness of the conditions imposed on the algorithm depends on the technique used to develop the convergence conditions. Of these techniques it can be said the Zangwill's theorems would give the most mild conditions but often it is not possible to use these theorems as a basis for proving convergence. Then one has to resort to other methods. At the other extreme, the use of contraction mapping theorems give strict conditions. However, one can say that the more strict the conditions are, the more the information regarding the algorithm we get. In fact, at times it is even possible to establish the rate of convergence **(Tatjewski and Roberts(1987))**. Another advantage of undertaking this kind of an analysis is that, for the problem we have, one can find out what really constitutes model-reality differences. Are they simply the parameter estimates or is there some other factor which decides the efficiency of the algorithm both in terms of the number of set-point changes and the iterative loop gains. These criteria of deciding the efficiency of the algorithm are discussed in greater detail in chapter 5.

4.3.1 The MSTP Algorithm

Brdys and Roberts(1987) have provided the conditions for and proved the convergence of the MSTP algorithm through Zangwill's theorems. These conditions are mild and do not give any idea as to what properties the model should possess

so that the algorithm is efficient. But the important result obtained through their analysis was that the MSTP algorithm will converge whether the OCP is convex or not. This is a very useful property of the algorithm and makes it invaluable in an on-line situation where one cannot predict the convexity or non-convexity of the OCP.

However, one would like to have some knowledge about the properties to be incorporated into the model so that the efficiency of the algorithm can be improved. This can be done if the conditions given by Brdys and Roberts are strenghtened. An analysis of these conditions would then tell us what constitutes model-reality differences. The following theorem gives conditions, which when analysed would answer the question of model-reality differences.

Theorem 4.3.1.1

Assume that

1. Every point of the set $\mathbb{C}$ satisfies the regularity conditions.
2. The set $\mathbb{C}$ is compact and convex.
3. The functional $q'_{\mathbb{C}}(\,.\,,\alpha)$ is uniformly monotone on $\mathbb{C}$ with some constant $a(\alpha) > 0$ for every $\alpha \in \mathcal{A}$ and also that $q'_{*}(\,.\,)$ is also uniformly monotone on $\mathbb{C}$ with some constant $a^{*} > 0$.
   where $\mathcal{A} = \bigcup_{V \in \Omega} \alpha(V)$
4. There exist such numbers $\bar{A}$ and $\underline{a}$ such that
   $$\bar{A} > A(\alpha) \qquad a(\alpha) \geq \underline{a} > 0 \tag{4.6}$$
   where $A(\alpha)$ is the Lipschitz constant of $q(\,.\,,\alpha)$ on $\mathbb{C}$.
5. The model is selected such that
   $$\bar{A} \leq 2a^{*} - \tfrac{1}{2}A^{*} \tag{4.7}$$
   $$\tfrac{3}{4}A^{*} < \underline{a} \leq A^{*} \tag{4.8}$$
   where A^{*} is the Lipschitz constant of q^{*} on $\mathbb{C}$
6. The class of systems considered is restricted in a manner
   $$a^{*} \geq \tfrac{5}{8} A^{*} \tag{4.9}$$

Then

I. There exists a solution $\bar{V}$ of the OCP

II. There exists such scalars ξ_1 , ξ_2 , such that $0 < \xi_1 < 0.5$, $0.5 < \xi_2 \leq 1$, such that for every $\xi \in (\xi_1, \xi_2)$ the algorithm generates a sequence (V^k) which converges to $\bar{V}$.

Proof

Part I of the theorem follows from the assumption of continuity and differentiablity of $q^*(\,.\,)$ and $q(\,.\,,\alpha)$ on $\mathcal{C}$, which is compact and every point of the set $\mathcal{C}$ satisfies the regularity conditions. Also the functions $q'_C(\,.\,,\alpha)$ and $q'_*(\,.\,)$ are uniformly monotone on $\mathcal{C}$. There is thus a unique point $\bar{V}$ which solves the OCP and satisfies the optimality conditions (Lemma 4.1).

Now consider the k^{th} iteration of the algorithm. From the assumptions made it is seen that, **(Luenberger (1970), Zangwill(1969))**

$$[q'_C(\hat{C}^k,\alpha^k) - \lambda\,](\bar{V} - \hat{C}^k) \geq 0 \tag{4.10}$$

using the modified version of λ given by (4.3), (4.10) can be transformed to the following:

$$[q'_C(\hat{C}^k,\alpha^k) - q'_C(V^k,\alpha^k) + q'_*(V^k)](\bar{V} - \hat{C}^k) \geq 0 \tag{4.11}$$

since $\hat{C}^k \in \mathcal{C}$ and $\bar{V} \in \Omega$ we have

$$[q'_*(\bar{V})](\hat{C}^k - \bar{V}) \geq 0 \tag{4.12}$$

then adding (4.11) and (4.12) we get

$$[q'_C(\hat{C}^k,\alpha^k) - q'_C(V^k,\alpha^k) + q'_*(V^k) - q'_*(\bar{V})](\bar{V} - \hat{C}^k) \geq 0 \tag{4.13}$$

we also have by the assumptions **(Cohen (1978))**

$$q'_C(\hat{C}^k,\alpha^k)(\bar{V} - \hat{C}^k) \leq q(\bar{V},\alpha^k) - q(\hat{C},\alpha^k) - \tfrac{1}{2}a \left\|\hat{C}^k - \bar{V}\right\|^2 \tag{4.14}$$

$$q'_C(V^k,\alpha^k)(\hat{C}^k - \bar{V}) \leq q(\hat{C}^k,\alpha^k) - q(V^k,\alpha^k) - \tfrac{1}{2}a\left\|\hat{C}^k - V^k\right\|^2 + q(V^k,\alpha^k) - q(\bar{V},\alpha^k)$$

$$+ \tfrac{1}{2}\left\|V^k - \bar{V}\right\|^2 \tag{4.15}$$

$$q_*'(V^k)(\bar{V} - \hat{C}^k) \leq q_*(\bar{V}) - q_*(V^k) - \tfrac{1}{2}a^*\left\|\bar{V} - V^k\right\|^2 + q_*'(V^k)(V^k - V^{k+1})$$

$$+ q_*'(V^k)(V^{k+1} - \hat{C}^k) \tag{4.16}$$

and

$$q_*'(\bar{V})(\hat{C}^k - \bar{V}) \leq q_*'(\bar{V})(\hat{C}^k - V^{k+1}) + q_*(V^{k+1}) - q_*(\bar{V}) - \tfrac{1}{2}a^*\left\|V^{k+1} - \bar{V}\right\|^2 \tag{4.17}$$

$$q_*(V^{k+1}) - q_*'(V^k) + q'_*(V^k) \leq \tfrac{1}{2}\,\xi^2 \left\| V^{k+1} - \hat{C}^k \right\|^2 \tag{4.18}$$

and

$$\left[\, q_*'(V^k) - q'_*(\bar{V})\right](V^{k+1} - \hat{C}^k) \leq \tfrac{1}{2}(1-\xi)A^*\left[\left\|V^k - \bar{V}\right\|^2 + \left\|V^k - \hat{C}^k\right\|^2\right] \tag{4.19}$$

Then using (4.14) to (4.19) in (4.13) and after some manipulations we get the following

$$a^*\|V^k - \bar{V}\|^2 - a^*\|V^{k+1} - \bar{V}\|^2 \geq \left[2a^* - \bar{A} + (\xi - 1)\right]\|V^k - \bar{V}\|^2 + \underline{a}\|\bar{V} - \hat{C}^k\|^2$$

$$+ \left[\underline{a} - A^*(\xi^2 - \xi + 1)\right]\|V^k - \hat{C}^k\|^2 \qquad (4.20)$$

From (4.20) it is easily seen that that right hand side of (4.20) will not be negative if

$$2a^* - \bar{A} + (\xi - 1) > 0 \qquad (4.21)$$

$$\underline{a} > 0 \qquad (4.22)$$

$$\underline{a} - A^*(\xi^2 - \xi + 1) > 0 \qquad (4.23)$$

This is ensured by the conditions (4.6) to (4.9) and these are sufficient for the existence of numbers ξ_1, ξ_2 of II. Again condition (4.9) is enough for the existence of the pair $(\underline{a}, \bar{A})$ to satisfy (4.7) and (4.8).

We now define a function T(V) such that

$$T(V) = a^*\|V - \bar{V}\|^2 \qquad (4.24)$$

It is seen from (4.20) that the sequence $\{T(V^k)\}$ is strictly decreasing and is bounded below by zero. Thus, the sequence is convergent and the left hand side tends to zero. Therefore the right hand side which is non negative also tends to zero. We can then conclude that

$$V^k - \bar{V} \dashrightarrow 0 \qquad (4.25)$$

Q. E. D.

Before discussing the results of theorem 4.3.1.1, let us formulate the following lemma. It is similar to that given by **Brady and Roberts (1987)**, and it helps in analysing the results of theorem 4.3.1.1 in terms of

a) The model-reality differences

b) The significance of the modifier λ

<u>Lemma 4.3.1.1</u>

Let the process be described by the following mathematical model

$$F(C,\alpha) = D(\alpha)C + P(\alpha)$$

where $D(\alpha)$ and $P(\alpha)$ are matrix functions of appropriate dimensions.

If the performance index Q(C,Y) is uniformly convex, then $q'_c(C,\alpha)$ is uniformly monotone with a constant $b(\alpha) > 0$ such that

$$b(\alpha) = f_1\left[\Lambda_{min}(D(\alpha)^T D(\alpha))\right]$$

where $\Lambda_{min}(D(\alpha)^T D(\alpha))$ is the least eigenvalue of the matrix $D(\alpha)^T D(\alpha)$.

Proof

By the assumptions of the lemma there is a constant $\sigma > 0$ (Ortega and Rheinboldt (1970)) such that

$$\left[Q'_C(C+h_1, Y+h_2) - Q'_C(C, Y)\right]h_1 + \left[Q'_Y(C+h_1, Y+h_2) - Q'_Y(C, Y)\right]h_2$$

$$\geq \sigma \left[\|h_1\|^2 + \|h_2\|^2\right] \tag{4.26}$$

for every C, $C+h_1 \in \mathbb{C}$ and Y, $Y+h_2 \in \Re^m$

We also have

$$q'_c(C,\alpha) = Q'_C(C,F(C,\alpha)) + Q'_Y(C,F(C,\alpha))F'_c(C,\alpha)$$
$$= Q'_C(C,F(C,\alpha)) + Q'_Y(C,F(C,\alpha))D(\alpha) \tag{4.27}$$

hence we have

$$\left[q'_c(C+h_1,\alpha) - q'_c(C,\alpha)\right]h_1 = \left[Q'_C(C+h_1,F(C+h_1,\alpha)) + Q'_C(C,F(C,\alpha))\right]h_1 +$$

$$\left[Q'_Y(C+h_1,F(C+h_1,\alpha)) + Q'_Y(C,F(C,\alpha))\right]D(\alpha)h_1 \tag{4.28}$$

Now $Y = F(C,\alpha)$ and let $h_2 = D(\alpha)h_1$ and since $F(C+h_1,\alpha) - F(C,\alpha) = D(\alpha)h_1$ we get using (4.28) along with (4.26)

$$\left[q'_c(C+h_1,\alpha) - q'_c(C,\alpha)\right]h_1 \geq \sigma \left[\|h_1\|^2 + \|D(\alpha)h_1\|^2\right] \tag{4.29}$$

$$\geq \sigma \left[1 + \Lambda_{min}(D(\alpha)^T D(\alpha))\right] \|h_1\|^2 \tag{4.30}$$

where $\Lambda_{min}(D(\alpha)^T D(\alpha))$ is the least eigenvalue of the matrix $D(\alpha)^T D(\alpha)$

Now let $b(\alpha) = \sigma \left[1 + \Lambda_{min}(D(\alpha)^T D(\alpha))\right]$

then we have

$$\left[q'_c(C+h_1,\alpha) - q'_c(C,\alpha)\right]h_1 \geq b(\alpha)\|h_1\|^2 \tag{4.31}$$

since $\sigma > 0$ and $\Lambda_{min}(D(\alpha)^T D(\alpha)) > 0$ we have $b(\alpha) > 0$

Q. E. D

This lemma shows that the monotonicity of the function q'_c is a function of the first order derivatives of the model output with respect to the control set-points. This is a very important conclusion. If this is similarly extended to the function q_x', we see that it is also the function of the first order derivatives of the system. We can thus write the following

$$a(\alpha) = f_2(F'_c(C,\alpha)) \tag{4.32}$$

$$a^* = f_b(K_*'(C)) \tag{4.33}$$

Consider assumption 5 of theorem 4.3.1.1, where some conditions were posed on the selection of the model. From these and the above relationships, (4.32) and (4.33), we see that the efficiency of the algorithm depends on the differences between the model based and process based derivatives. The closer these are, the more efficient is the iterative process. In fact this conclusion can also be arrived at from the sufficiency conditions, which the model has to satisfy, given in section 2.4.

In practice there are bound to be model-reality differences, and as a result, one finds that the properties of the model based performance index are different from that of the system based performance index. It was stated in chapter 3 that this difference was compensated for by the modifier λ. In the light of (4.32) and (4.33), we can define the modifier as a function of the differences in the monotone properties of the system based performance index and the model based performance index. Thus, we can write λ as:

$$\lambda = f_3(f_2^{-1}(a(\alpha)) - f_b^{-1}(a^*)) \tag{4.34}$$

and as a result give the modifier two interpretations which are, ultimately, the same except that they are arrived at through different means. These are:

a) A parameter which quantifies the violations of the sufficiency conditions by the model.

b) A parameter which compensates for the differences in the properties of the model based performance index and the real system based performance index.

But whatever the interpretation one gives to the modifier, we see that the derivative differences are of prime importance, and we can conclude that the derivative differences are what really constitute the model-reality differences. The greater these differences the more inefficient the performance of the algorithm will be. In fact, this conclusion can be further verified through the simulation study presented in chapter 5.

4.3.2 The AISOPE Algorithm.

In the previous section, it has been shown that the MSTP algorithm would converge provided the model and the system, under study, satisfied certain conditions. It has also been shown that the MSTP algorithm will converge for any system, provided the model based problem is convex (**Brdys and Roberts(1986)**). However, in order to arrive at some conclusions regarding the efficiency of the

algorithm, and the model-reality differences, the conditions posed have to be stricter than required.

It has also been stated that the modifier is some sort of a compensating factor, and the amount of compensation required decides the efficiency of the algorithm. We can improve the efficiency of the algorithm by reducing the amount of compensation required. This can be done by changing the model so that the model-reality differences are reduced. However, sometimes it is not possible to change a given model. Then one has to resort to other means to improve the overall efficiency. This is done by augmenting the performance index by a quadratic penalty function, as is done in the AISOPE algorithm. This algorithm retains all the features of the MSTP algorithm, the difference being the augmentation. This leads to a slackening in some of the conditions posed in assessing the convergence of the algorithm. The following theorem gives these conditions. They are similar to those given in theorem 4.3.1.1. Here however, the effect of the augmentations is taken into account. The proof for this theorem is similar to that given for theorem 4.3.1.1.

Theorem 4.3.2.1

Assume that assumptions 1, 2, 3, 4, of theorem 4.3.1.1 are satisfied, and additionally assume that

1) The model is selected such that

$$\tilde{A} \leq 2a^* - \frac{A^*}{2} \tag{4.35}$$

$$\frac{3}{4}A^* < \tilde{a} \leq A^* \tag{4.36}$$

where $\tilde{A} \geq \max_{C \in \mathcal{C}} \|q'_C(C,\alpha)\| + 2\rho$ (4.37)

$$\tilde{a} \geq \min A \min \|q''_C(C,\alpha)\| + 2\rho \tag{4.38}$$

2. The class of systems under consideration is restricted such that

$$a^* \geq \frac{5}{8}A^* \tag{4.39}$$

Then the assertions of theorem 4.3.1.1 are valid.

The proof of this theorem follows along similar lines to that given for theorem 4.3.1.1 and hence is omitted here.

Notice that $\tilde{A}$ and $\tilde{a}$ are the Lipschitz and monotone constants of the function $q(C,\alpha) + \rho\|V-C\|^2$ and hence we can modify (4.37) and (4.38) to:

$$\tilde{A} \geq A(\alpha) + 2\rho \tag{4.40}$$

$$\tilde{a} \geq a(\alpha) + 2\rho \tag{4.41}$$

where $a(\alpha)$ and $A(\alpha)$ are as before.

Then analysing the conditions as before we can get the following expression

$$\Gamma = f_4(f_3(f_a^{-1}(a(\alpha)) - f_b^{-1}(a^*)) + 2\rho)$$
$$= f_4(\lambda + 2\rho) \qquad (4.42)$$

Notice the similarity between (4.34) and (4.42). Equation (4.34) is the expression for the modifier and (4.42) is the same but now takes into account the effect of the augmentation. Notice that if the model-reality differences — which are reflected in the properties of the model based performance index and the system based performance index — are large then, this is reflected in λ being large, and also the model may violate the conditions of theorem 4.3.1.1. However, with augmentation it is seen that if the conditions of theorem 4.3.1.1 are violated, we can select a penalty factor, ρ, such that the conditions given by theorem 4.3.2.1 are satisfied. Then, from (4.42) we see the effect this has on the compensation taking place.

It is for this reason that the interpretation that the modifier, λ, is a parameter which compensates for the differences in the properties of the model based and system based performance indices, could be a more accurate definition.

4.3.3 The ALM-ISOPE Algorithm.

The ALM-ISOPE algorithm has been derived, using the sufficiency conditions given in section 2.4. Because we ensure that the sufficiency conditions are met at each iteration, this algorithm proves to be quite successful and relatively more efficient. For this algorithm the conditions given by theorem 4.3.1.1 are valid, only some minor modifications are necessary to take into account the fact that the model is not provided to the algorithm, but the algorithm develops one itself.

Theorem 4.3.3.1

Assume that

1. **Every point of the set $\mathbb{C}$ satisfies the regularity conditions.**
2. **The set $\mathbb{C}$ is compact and convex.**
3. **The functional $q'_C(\,.\,)$ is uniformly monotone on $\mathbb{C}$ with some constant $\underline{a} > 0$.**
4. **The iterative loop gains are selected such that the algorithm develops a model such that**

$$\bar{A} \leq 2a^* - \tfrac{1}{2}A^* \qquad (4.43)$$

$$\tfrac{3}{4}A^* < \underline{a} \leq A^* \qquad (4.44)$$

where $\bar{A}$ is the Lipschitz constant of q on $\mathbb{C}$ and a^*, A^* are as before

5. **The class of systems considered is restricted in a manner**

$$a^* \geq \frac{5}{8} A^* \qquad (4.45)$$

Then

I. There exists a solution $\bar{V}$ of the OCP

II. There exists such scalars ξ_1 , ξ_2 , such that $0 < \xi_1 < 0.5$, $0.5 < \xi_2 \leq 1$, such that for every $\xi \in (\xi_1, \xi_2)$ the algorithm generates a sequence (V^k) which converges to $\bar{V}$.

The proof of this is again similar to that given for theorem 4.3.1.1 and is not given.

It should be noted here that, given the formulation of the ALM-ISOPE, one could be under the impression that the properties of both the OCP and the model based problem should be the same, and as a result the algorithm should converge in one iteration. However, this is not the case, because the method is a form of system linearization about a given point. If, however the system is highly non-linear the linear approximation need not be an accurate description of the system since then the higher order derivatives may have a larger role to play. This, then would lead to a difference in the properties of the two problems. Hence, it is likely that the algorithm performs more than one iteration. However, it has been found that this algorithm approaches the optimum much faster, as shall be seen in the following chapter, than the other algorithms.

For the ALM-ISOPE algorithm, the conditions for convergence given by theorem 4.3.3.1 are strict. It is possible to develop conditions which are weaker, and these are similar to those given by Brdys and Roberts for the MSTP **(Brdys and Roberts (1987))** and use Zangwill's **(Zangwill(1969))** Convergence Theorem A as the basis for proving the convergence. Before formulating these conditions let us define the mapping of the algorithm. We shall first define the mapping for each step of the algorithm, and the mapping of the whole algorithm is then a composition of the seperate mappings **(Zangwill (1969))** :

Step I : is directly described by the system input-output mapping.

Step II: Let $\hat{C}(V) \triangleq \arg \min_{C \in \mathcal{C}} [Q(C,F(C)]$

Then step II can be described as

' For a given V^k find C^k such that $C^k \in \hat{C}(V^k)$ '

Step IV: is concerned with the updation of the control set-points that is

$$V^{k+1} = V^k + \xi^k (C^k - V^k)$$

We shall assume that the iterative loop gains change at each iteration, this means that every time the controls are updated we select some sort of an optimum gain, so that the iterative

process is stable.

where ξ^k is the iterative loop gain such that $\tau \leq \xi^k < B(V)$

where $\tau > 0$, b(V) is an appropriately defined function of V.

Then the mapping is deifned by the following

$\mathcal{C} \times (\mathcal{C} - \mathcal{C}) \ni (V,d) \dashrightarrow \upsilon(V,d) \in 2^{\Re^n \times \Re^n}$ and

$$\upsilon(V,d) = \left\{ V + \xi d : \tau \leq \xi < B(V) \right\}$$

Then step IV can be described as :

' For given V^k and C^k find V^{k+1} such that

$$V^{k+1} \in \upsilon(V^k, C^k - V^k)'$$

Then the k^{th} iteration can be described as finding V^{k+1} such that $V^{k+1} \in \Pi(V^k)$, where Π is an algorithmic mapping defined by

$\mathcal{C} \ni V \dashrightarrow \Pi(V) \in 2^{\mathcal{C}}$

$\Pi = \upsilon \tilde{\varphi}$

$\tilde{\varphi}$ is a point-to-set map which associates each point $V \in \mathcal{C}$ a set

$$\left\{(V, C-V) \in \mathcal{C}\times(\mathcal{C}-\mathcal{C}) : C \in \hat{C}(V)\right\}$$

Theorem 4.3.3.2

Assume that

1) The set $\mathcal{C}$ is bounded.

2) The functions G are convex and continuous.

3) The function $q_{\mathcal{K}}'$ is Lipschitz continuous on $\mathcal{C}$ with a constant $\delta > 0$, i.e;

$$\left\| q_{\mathcal{K}}'(C+h_1) - q_{\mathcal{K}}'(C) \right\| < \delta \left\| h_1 \right\| \qquad (4.46)$$

4. The function $q'_c()$ is uniformly monotone on $\mathcal{C}$ with a constant b(V) > 0, i.e;

$$\left[q'_c(C+h) - q'_c(C) \right] h \geq b(V) \|h\|^2 \qquad (4.47)$$

and b(V) is upper semi-continuous on $\mathcal{C}$ and $\inf_{V \in \mathcal{C}} b(V) > 0$

5. The iterative loop gains are such that

$\tau \leq \xi \leq B(V)$

$\tau > 0$ and B(V) is an appropriately defined function of V

Then

I If $B(V) = \min \left\{ 1, \dfrac{2 \inf_{V \in \mathcal{C}} b(V)}{\delta} - \epsilon \right\}$ (4.48)

and $\tau$and ϵ are selected such that

$$1 \geq \tau > 0 \, , \, \epsilon > 0$$

and

$$\tau + \epsilon \leq \frac{2 \inf_{V \in \mathcal{C}} b(V)}{\delta} \tag{4.49}$$

then the algorithmic mapping is well defined on $\mathcal{C}$.

II. There is atleast one subsequence $\{V^k_i\}$ of the sequence $\{V^k\}$ generated by $V^{k+1} \in \Pi(V^k)$, and the limit point of $\{V^k_i\}$ belongs to the set Ω, where $\Pi(V^k)$ is the algorithmic mapping

III. Each V^k satisfies real system constraints and the following is satisfied

$$q_{\Xi}(V^{k+1}) \leq q_{\Xi}(V^k) \text{ if } V^k \in \Omega \text{ for all } k = 0, 1, 2, \cdots$$

Proof

Due to assumptions 1 and 2 the set $\mathcal{C}$ is compact and convex. Due to assumption 4 the functions

$$\mathcal{C}X\mathcal{C} \in (C,V) \dashrightarrow q(C,V) \in \Re^1$$

are continuous on $\mathcal{C}X\mathcal{C}$ and $\mathcal{C}$ is compact. Hence the mapping $\hat{C}(V)$ is well defined. Because of the monotonicity assumption made in 5, $\hat{C}(V)$ consists of a single point for every $V \in \mathcal{C}$. Thus, $\hat{C}$ is a point-to-point continuous mapping, **(Hogan(1973))**. The compactness of $\mathcal{C}$ and the continuity of $\mathcal{C}$ implies that $\hat{C}$ is closed on a compact set. Thus, the accessory mapping $\tilde{\Psi}$ is closed on $\mathcal{C}$.

Next, consider the mapping υ. Since we have $\inf_{V \in \mathcal{C}} b(V) > 0$, and τ, ϵ satisfy the inequality of I, then it is easy to verify $\upsilon(.)$ is well defined. Now, we show that $\upsilon(.)$ is closed at every point $(V,d) \in \mathcal{C}X(\mathcal{C}-\mathcal{C})$ such that $d=0$.

Let $\mathcal{C} \ni V \dashrightarrow \inf_{V \in \mathcal{C}} b(V) \in \Re^1$

and $\upsilon(V^k, d^k) \in y^k = V^k + \xi^k d^k \dashrightarrow_{k \to \infty} y$

Therefore for a very large k, we have

$$\xi^k = \frac{\|y^k - V^k\|}{\|d^k\|} \dashrightarrow_{k \to \infty} \frac{\|y - V\|}{\|d\|} = \xi$$

thus $y = V + \xi \, d$

since $\tau \leq \xi^k \leq B(V^k)$ and due to assumption, $B(V)$ is upper semi-continuous, then

$$\tau \leq \xi^k \leq \limsup_{k \to \infty} B(V^k) \leq B(V)$$

therefore $y \in \upsilon(V, d)$ and consequently υ is closed.

Then to recap we have a set $\mathcal{C}$ which is compact, and thus $\mathcal{C}-\mathcal{C}$ is also compact and consequently $\mathcal{C}\times(\mathcal{C}-\mathcal{C})$ is also compact. We can then say that the algorithmic mapping Π is closed at every point which is not a solution. We see that $B(V) \leq 1$ and $\tau > 0$, and $\mathcal{C}$ is convex, then $\Pi(V) \in 2^{\mathcal{C}}$.

Then we can state that all the points generated by the algorithm are in a compact set $\mathcal{C}$ and the algorithmic mapping is closed outside an algorithm solution set.

We now prove the final assertion. Due to assumptions 3 and the convexity of $\mathcal{C}$, we have **(Kantorovich and Akilov (1964))** :

$$q*(V) - q*(C) \geq q'*(V)\,(V - C) - \frac{\delta}{2}\|V - C\|^2 \tag{4.50}$$

Now, we have $C = V + \xi\,(\hat{C}(V) - V)$ and $\tau \leq \xi \leq B(V)$ where

therefore

$$q*(V) - q*(C) \geq \xi q*'(V)(V - \hat{C}(V)) - \frac{\xi^2\delta}{2}\|V - \hat{C}(V)\|^2 \tag{4.51}$$

Now from assumption 4 and the convexity of $\mathcal{C}$ **(Luenberger (1970))**

$$q'_{\mathcal{C}}(\hat{C}(V))\left[V-\hat{C}(V)\right] \geq 0 \tag{4.52}$$

and we also have

$$\left[q'_{\mathcal{C}}(V) - q'_{\mathcal{C}}(\hat{C}(V)\right](V - \hat{C}(V)) \geq b(V)\|V-\hat{C}(V)\|^2 \tag{4.53}$$

Due to the definition of the model we have

$$q'*(V) = q'_{\mathcal{C}}(V) \tag{4.54}$$

Then using (4.52), (4.53), (4.54) we get

$$q'*(V)(V-\hat{C}(V)) \geq b(V)\|V-\hat{C}(V)\|^2 \tag{4.55}$$

Using this in (4.51) we get

$$q*(V) - q*(C) \geq \left[\xi\, b(V) - \frac{\xi^2\delta}{2}\right]\|V - \hat{C}(V)\|^2$$

$$\geq \tau\left[b(V) - \frac{\xi\delta}{2}\right]\|V - \hat{C}(V)\|^2 \tag{4.56}$$

since $\tau > 0$ and $\tau \leq \xi \leq B(V)$ and I we have

$$\xi \leq B(V) \leq \frac{2}{\delta}\inf_{V \in \mathcal{C}} b(V) - \epsilon \leq \frac{2}{\delta}\, b(V) - \epsilon \tag{4.57}$$

$$\Rightarrow \frac{\xi\delta}{2} \leq b(V) - \epsilon\delta$$

$$\Rightarrow b(V) - \frac{\xi\delta}{2} \geq \epsilon\delta \tag{4.58}$$

Using this (4.58) in (4.56) we get,

$$q^*(V) - q^*(C) \geq \tau\delta\varepsilon \left\| V - \hat{C}(V) \right\|^2 \qquad (4.59)$$

ans since $\tau > 0$, $\delta > 0$, $\varepsilon > 0$, and $\left\| V - \hat{C}(V) \right\|^2$

then we have

$$q^*(V) \geq q^*(C)$$

$$\Rightarrow \qquad q^*(V^{k+1}) \leq q^*(V^k)$$

Q. E. D

It was stated that the MSTP algorithm will converge to the optimum, whether the OCP is convex or not **(Brdys and Roberts(1987))**. In fact, this is also true for the ALM-ISOPE, or for that matter any of the centralised ISOPE algorithms. It should be noted here that the only condition we impose on the system based problem is that the performance index be continuous. This is a standard assumption or else it would not be possible to establish the convergence of the algorithm. A stricter condition was posed on the model based performance index, in that it had to be monotone. This can be ensured always if the original performance index, Q(C,Y) is convex. This can be seen from the result of lemma 4.3.1.1 where the assumption of monotonicity was proved using a linear model.

4.4 CONCLUDING REMARKS

This chapter dealt with three aspects of the centralised adaptive algorithms, viz

a) Under what conditions does a given algorithm converge?

b) What constitutes the model-reality differences?

c) What is the significance of the modifier?

The convergence conditions had already been given for the MSTP and the AISOPE algorithms. An analysis of these conditions does not answer the remaining questions. To do this the conditions for convergence had to be made stricter. In general one can say that the stricter the conditions one imposes the more information we can get about the algorithm. Eventually, it may also be possible to come to some conclusions about the rate of convergence, as would be possible if the contraction mapping theorems of **Ortega and Rheinboldt(1970)**, as was done by **Tatjewski and Roberts (1987a)** for the NL-ISOPE algorithm.

From the analysis presented in this chapter we can conclude that

a) The derivative differences, $\left[F'_c(V,\alpha) - K'_*(V)\right]$ constitute the model-reality

differences

b) The modifier can be interpreted in either of the following two ways

i) A parameter which quantifies the violations of the sufficiency conditions by the models

or

ii) A compensator which compensates for the differences in the model based performance index and the system based performance index.

These conclusions are of prime importance for, having once understood the model-reality difference, we can then ensure that the model has the necessary characteristics so that the performance of the algorithm is efficient. And hence the smaller the model-reality differences the more efficient is the performance of the algorithm. Then, the logical extension of this is the ALM-ISOPE algorithm, where at the current point there are no model-reality differences and hence should be more efficient than the MSTP algorithm. Also, understanding the function of the modifier has enabled us to understand the effect of the augmentations, and as a result the value of penalty factor ρ, on the functioning the algorithm. It is in this context that we define the modifier as a compensating parameter rather than a parameter which quantifies the violation of the sufficiency conditions by the model.

In the next chapter we present a simulation study, so that the conclusions of this chapter can be verified. The simulation study is carried out using two example systems. One, an abstract example and the other, a more realistic example, a fuel gas mixing system, consisting of a vapouriser and gas mixer.

CHAPTER 5. SIMULATION RESULTS FOR THE CENTRALISED CONTROL STRUCTURES.

5.1 INTRODUCTION

In the preceeding chapters, techniques for solving the on-line optimising control problem were introduced. In chapter 3, both direct methods and adaptive, model-based methods were discussed, along with some of the advantages of using adaptive techniques relative to both the direct methods and fixed-model based methods. In chapter 4 the convergence properties of the algorithms were presented and the analysis was carried out in terms of :

a) The model-reality differences

b) The effect of and the properties of the modifier λ

In this chapter the conclusions of chapters 3 and 4 will be verified through a simulation study, on two examples namely

a) Example 1 is an abstract system.

b) Example 2 is a Butane Fuel Gas mixing system.

The programs used for the computer simulation can be seen in appendix — A.

5.2 CRITERIA FOR DECIDING THE EFFICIENCY OF ALGORITHMS.

One of the common methods of assessing the performance of iterative algorithms is to compare the number of iterations needed by each algorithm to provide the required solution. However, this criterion is not a suitable one for algorithms which are to be implemented on-line. Take for example, the direct technique, here we find that although the number of iterations are few, the number of simultaneous set-point changes are very large. This is because we have to perform linear searches and every time the performance of the system has to be evaluated there is a need for measurements from the system. Thus, it is the number of set-point changes which is important here, for the numerical computations take very little time compared to the time required for the system to settle down to a steady-state.

If we were just comparing the direct technique with an adaptive algorithm then the comparision of the set-point changes would be sufficient . However, if on the other hand, a comparison of two adaptive algorithms, of the relative efficiency of an algorithm using two different models is made, then this criterion

is not sufficient. For example, if we have a given system, its model and performance index, then the iterative behaviour of each algorithm is bound to be different because of the nature of the modifications made to the problem or the properties of the models as the case may be. The iterative behaviour of algorithms can be observed by paying attention to :

a) The stabilty of the algorithmic/iterative process.

b) The number of set-point changes.

Both these features are interconnected. The stability of the algorithmic process can be defined by the iterative loop gains used in the relaxation schemes. A high gain signifies a high degree of stabilty of the algorithm for that example and would result in a reduced number of set-point changes. A small gain value indicates the reverse. The values of these gains depend on the model-reality differences. One immediately realises that the larger the model-reality differences then the smaller the gains to be used and hence the larger the number of set-point changes. If a very large gain is used, in order to reduce the set-point changes, then it is possible that the algorithm would move away from the solution or would tend to oscillate about the solution and hence a compromise has to be found between the desire to reduce the number of set-point changes and the stability of the algorithm. Hence, we can say that both the iterative loop gains as well as the set-point changes are characteristics of the algorithm. Thus, when comparing two adaptive algorithms, the criteria one has to take into account are the number of setpoint changes and the value of the gains.

5.3 EXAMPLES

5.3.1 Example 1

The first example is an abstract one and has 5 control set-points, and 3 outputs. For purposes of simulation the real process is represented by a system of equations:

$$\begin{bmatrix} 1 & -(2 + 0.15C_1) & 0 \\ -1.2 & 1.0 & 3.0 \\ 1.0 & -1.0 & 1.0 \end{bmatrix} \begin{bmatrix} Y_{1*} \\ Y_{2*} \\ Y_{3*} \end{bmatrix} = \begin{bmatrix} 1.3C_1 - C_2 \\ C_3 - C_4 + C_4^2 \\ 2.0C_4 - 1.25C_5 + 0.25C_4C_5 \end{bmatrix}$$

Performance index:

$$Q = (Y_{1*} - 1.0)^2 + 2.0(Y_{2*} - 2.0)^2 + (Y_{3*} - 3.0)^2 + \sum_{i=1}^{5} C_i^2$$

For this system 3 different models were used namely :

a) Linear Model I

$$\begin{bmatrix} 1 & -2 & 0 \\ -1 & 1 & 3 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} C_1 - C_2 \\ C_3 - C_4 \\ 2.0C_4 - C_5 \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

b) Non-Linear Model I

$$\begin{bmatrix} 1 & 3 & 0 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} C_1 - C_2 \\ C_3 - C_4^2 \\ 2.0C_4 - C_4C_5 \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

c) Non-Linear Model II

$$\begin{bmatrix} 1 & -(1.5 + 0.1C_1) & 0 \\ -1 & 1 & 6 \\ 0.9 & -1 & 1 \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} C_1 - C_2 \\ C_3 - C_4^2 \\ 6.0C_4 - C_4C_5 \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

The simulations were carried out using all the models, from the same starting conditions which is:

$$C = \{ C_{ij} \} \quad ,i = 1,2 \ \wedge \ j = 1,2,3$$

$$= \{ 0, 0, 2, 2, 0 \}$$

5.3.2. Example 2

The second example is a fuel gas mixer system, operated by ICI plc, and is shown in figure 5.1. This is a part of a large olefine plant, and as can be seen, the system effectively consists of two sub-systems, which are treated as a whole. The

systems are :

a) The Vapouriser, which receives liquid butane to be converted into vapour

b) The Gas Mixer, where the Butane is mixed with Hydrogen, Methane, Ethane and Propane each of which is present in the flows Q_6, Q_7, Q_8, Q_9 and which enter as constant flows in fixed proportions to provide a fuel gas mix to be delivered to a furnace.

The aim of the regulatory control in the system is to supply the furnace fuel gas in the pressure range 60 psi – 90 psi. Two secondary level control loops for steam condensate and liquid butane also exist. In addition, two relief valves are connected to the gas mixer so that fuel gas can be rapidly taken from or added to the gas mixer during unexpected conditions.

The real system is represented by the following relations. For the vapouriser, the heat transferred from the steam to the butane is considered to be proportional to the differences in temperature between the steam and butane and is also proportional to the areas of the tubes in the steam side and the butane side. This area is assumed to be proportional to the difference in the levels of the steam condensate and the liquid butane. Thus, the relations are (for notations and symbols see appendix - B),

$$Q = K_q\,(T_{st} - T_{vap})\,(L_{vap} - L_{st}) \tag{5.1}$$

$$q_{st} = \frac{Q}{h_{fgw}} \tag{5.2}$$

Then a mass balance between the liquid butane inlet and outlet flow gives

$$\frac{dM_{vap}}{dt} = q_{45} - q_{12} \tag{5.3}$$

The energy balance for the vapouriser, and pressure/temperature relations, provide the following:

$$\frac{dT_{st}}{dt}(M_s C_{ps} + M_{vap} C_{pb})\frac{dT_{vap}}{dt} = Q - h_{fgb} q_{12} - q_{45}(T_{vap} - T_{45})C_{pb} \tag{5.4}$$

$$= \frac{(q_{st} - q_{cond})}{A_w \rho_w}$$

$$P_{vap} = 2.2166\ T_{vap} - 53.38 \tag{5.5}$$

Similarly, for the gas mixer concentrations, where the mixer is treated as a single volume, the mass balances and the internal energies of the gas are :

$$x_2^{\,i} = \frac{M_2^{\,i}}{\sum_{j=1}^{5} M_2^{\,j}} \qquad i=1,5 \tag{5.6}$$

$$\frac{dM_2}{dt} = x \begin{bmatrix} q_6 \\ q_7 \\ q_8 \\ q_9 \\ q_{12} \end{bmatrix} - (q_{23} + q_{27}) x_2 \tag{5.7}$$

$$\frac{dE_2}{dt} = q_{12}C_P T_{vdP} + \sum_{i=1}^{5} q_i h_i - (q_{23} + q_{27})T_2 \, x_2{}^1C_P \tag{5.8}$$

$$T_2 = \frac{E_2}{M_2{}^1C_v} \tag{5.9}$$

$$P_2 = \frac{(M_2{}^1r_2)}{V_2} \tag{5.10}$$

$$\rho_2 = \frac{(\sum_{i=1}^{5} M_i)}{V_2} \tag{5.11}$$

$$q_{12} = K_{12}(P_{vdP} - P_2) \tag{5.12}$$

$$q_i = k_i\sqrt{\rho_i \Delta P} \tag{5.13}$$

where x_2 , M_2 , and E_2 denote the concentration, mass, and internal energy of the gas mix in the mixer.

These equations, together with the three control loops which consists of three term controllers, the measurement transducers and valve actuators provide the process description. The measurement transducers and actuators were modelled as first order lags.

Equations (5.1) to (5.13) represent the mapping K_* described earlier, where the output vector Y_* is given by

$$Y_* = \{ q_{cond} , q_{12} , q_{23} \} \tag{5.14}$$

and the control vector is given by

$$C = \{ L_{st} , L_{vdP} , P_2 \} \tag{5.15}$$

The performance of the system is based on the following index

$$Q = 1200(L_{vdP} - 0.15)^2 + 250(L_{st} - 0.3)^2 + 0.15(P_2 - 70)^2 + 4q_{12} + 10q_{cond} - 15q_{23} + 128 \tag{5.16}$$

The above system was simulated using two models. The first model had a similar structure to that of the above and consisted only of the mass balance and heat transfer equations. The difference here was that the constants T_{st} , T_{vap} , h_{fgb} , h_{fgw} and C_{pb} varied by 5% from those of the process. Such variations are likely to occur in practice due to scaling, heat losses and other factors. The parameter estimation procedure is carried out in accordance to the relationships already established, and the following were the parameters estimated:

$$\alpha = \{ K_q , K_{12} , (q_{23} - q_{12}) \} \tag{5.16}$$

The second model used for this system was a linear model of type

$$Y = W C + \alpha \tag{5.17}$$

where $W \in \Re^{3X3}$ has the following elements

$$W = \begin{bmatrix} -8.0 & 8.0 & -4.0 \\ -8.0 & 8.0 & -4.0 \\ -1.0 & 1.0 & -0.001 \end{bmatrix} \tag{5.18}$$

Here the parameters α are different from those given by (5.16) and represent the differences between the real outputs and the model outputs and are

$$\alpha = (\alpha_1 , \alpha_2 , \alpha_3) \tag{5.19}$$

From the above model one immediately realises that the effective gradients of the model are constant throughout the algorithmic procedure and the physical structure of this model is totally different from that of the real system.

5.4 SIMULATION RESULTS

Both the two examples described above were simulated using the PRIME 750 computer and where necessary the NAGF (Numerical Algorithms Group) Library was used.

5.4.1 Example 1

The first example was simulated using the three models already mentioned. In all the cases the initial conditions were (0, 0, 2, 2, 0).

Table 5.1 summarises the results of noise free simulations using Powell's conjugate direction optimisation technique. This was used directly, since whenever the performance index was to be evaluated, system measurements were used. The simulations confirmed the earlier conclusions that this method could prove to be

difficult to implement as the number of simultaneous set-point changes required is very large —2204 in this example. This number is prohibitive and the whole process could take a very long time as in reality allowance has to be made for the process to sufficiently settle. Also, if there is any presence of measurement noise then the whole algorithm is likely to fail to converge. Also in the same table a summary of the results obtained using a fixed model is given. The degree of sub-optimality when using the fixed model, which in this case was 9.5% , is clearly model dependent. This clearly proves the case for using model based adaptive techniques.

The results obtained using the MSTP algorithm are shown in Table 5.2. This table summarises results using the various models after convergence was obtained.Table 5.3 gives the summary of the results obtained using the ALM-ISOPE algorithm. Here of course, as mentioned earlier, we do not have to provide an a priori model to the algorithm as it develops a linear model on its own. The relative rate of convergence using the various models with the MSTP algorithm and the ALM-ISOPE algorithm can be seen in Figure 5.2 where the variations at each iteration in the objective function are plotted. Figure 5.3 shows the variations in the parameters estimated for the three models at each iteration. Finally, Figures 5.5 to 5.7 show the variations in the derivative differences ($F'_c - K_*$) with each model and Figure 5.8 is a consolidation of these three figures and Table 5.4 gives these differences when final convergence is obtained.

5.4.2 Example 2

The initial conditions for the second system were

$$C = \{ 0.06, 0.08, 60.5\} \tag{5.20}$$

and , at all times after any changes in controls, sufficient time was allowed for a new steady-state condition to come into being before measurements were taken.

The final converged results of the simulations for the various techniques are shown in Tables 5.5 to 5.7. In all cases, the relief valve system never came into operation. Table 5.5 summarises the results obtained using the direct technique and as with the previous example it is clearly seen that the number of simultaneous set-point changes are high. Table 5.6 gives the summary of the converged results using the two models and finally Table 5.7 is for the ALM-ISOPE algorithm. Figure 5.9 shows the variations of the objective function using the two models with the MSTP and the ALM-ISOPE algorithms. Figure 5.10 is similar to that of Figure 5.8 in that it shows the variations of the derivative differences at each iteration using

the two models.

Table 5.7 summarises the results obtained using the AISOPE algorithm with the first example and the Non-linear Model I.

5.5 DISCUSSION OF THE RESULTS

One of the primary aims of any simulation exercise is to verify the theoretical results obtained. It was mentioned in the previous chapter that the derivative differences ($F'_c - K_*$) consitute and define the model-reality differences more than just the free parameters which are estimated through solving the parameter estimation problem.

Take example 1. Here we find that although the parameter estimates obtained using Linear Model I when convergence is obtained, see Table 5.2, do not differ greatly from those obtained using the Non-linear model II, the number of set-point changes made using Linear model I is much less than those for the second model, and so also with Non-linear model II. If, now, we compare the derivative differences (Table 5.4 and Figures 5.5 to 5.8) we see that there is a large variation in the derivative differences. Using linear model I the differences are smaller than those when using the other models. This can be seen further with the results obtained with the second example. The derivative differences, for the fuel-gas mixing, using the linear model were smaller than those when using the model where the physical structure of the model was the same as that of the system (figure 5.10). It was stated in section 5.2 that the iterative loop gains define the stability of the algorithmic process. This can be verified from the results obtained, where we find that the value of the iterative loop gains decreases as the model-reality differences increase.

We can then conclude that the smaller the derivative differences the better the rate of convergence and the better are the stabilty properties of the algorithmic process. This is further reinforced by the value of gains which could be used using various models with the two examples. It can be seen that the smaller the derivative differences the higher the gains we could use, and hence the greater is the stability of the algorithmic procedure.

If, we then extend this concept, as mentioned in chapter 4, that the smaller the derivative differences the better the iterative process then using the ALM-ISOPE algorithm we should get the best results —both in terms of the rate of convergence, and hence the number of set-point changes, and the values of the iterative loop gains. This inference is confirmed through the simulation study carried out on the two examples. It is seen that with the first example we could use

unity gain and the algorithm converged in 5 iterations. And with the second example the gains used were again unity and the algorithm converged in 5 iterations.

In the last chapter we gave two interpretaions to the modifier λ, namely that we either interpret it as

a) A parameter which quantifies the violations of the sufficiency conditions (see section 2.4)

or

b) As a parameter which quantifies the differences in the properties of the model based performance index and the process based performance index.

The simulations carried out with the MSTP and the ALM-ISOPE algorithms verify the first interpretation. The second one can be verified using the results obtained with the AISOPE algorithm.

If we refer to Table 5.7, it can be seen that augmenting the performance index by a quadratic penalty function, the algorithm behaves in a better fashion. We see that the number of iterations are reduced and higher iterative loop gains can be used. We see that the model used (Non-linear model I) has not been changed in any manner at all and hence the derivative differences remain the same. Then using the penalty function has some effect on the algorithm. It was shown in the last chapter that the modifier λ can be written as a function of $a(\alpha)$ and a_* :

$$\lambda = f_1(a(\alpha)) - f_2(a_*)$$

This was for the MSTP and the AISOPE algorithms. However with the AISOPE algorithm we now have to add the effect of the penalty function. If, then, by Λ we represent the modified version of λ taking into account the penalty function then we get the following

$$\Lambda = f_1(a(\alpha)) - f_2(a_*) + 2\rho$$

We then see that the penalty factor ρ in fact augments the model based performance index so that its monotone number and the Lipschitz constant (see Theorem 4.3.2.1) are closer to that of the process based performance index. Hence, we can also interpret the modifier in the second way.

5.6 CONCLUDING REMARKS

From the results of chapter 4 and those obtained in this chapter, we can conclude the following :

a) that, apart from the estimated parameters, the differences between the

model based derivatives and the process based derivatives represent the model-reality differences.

b) that the modifier λ quantifies the differences between the purely model based optimisation problem and the process based optimisation problem.

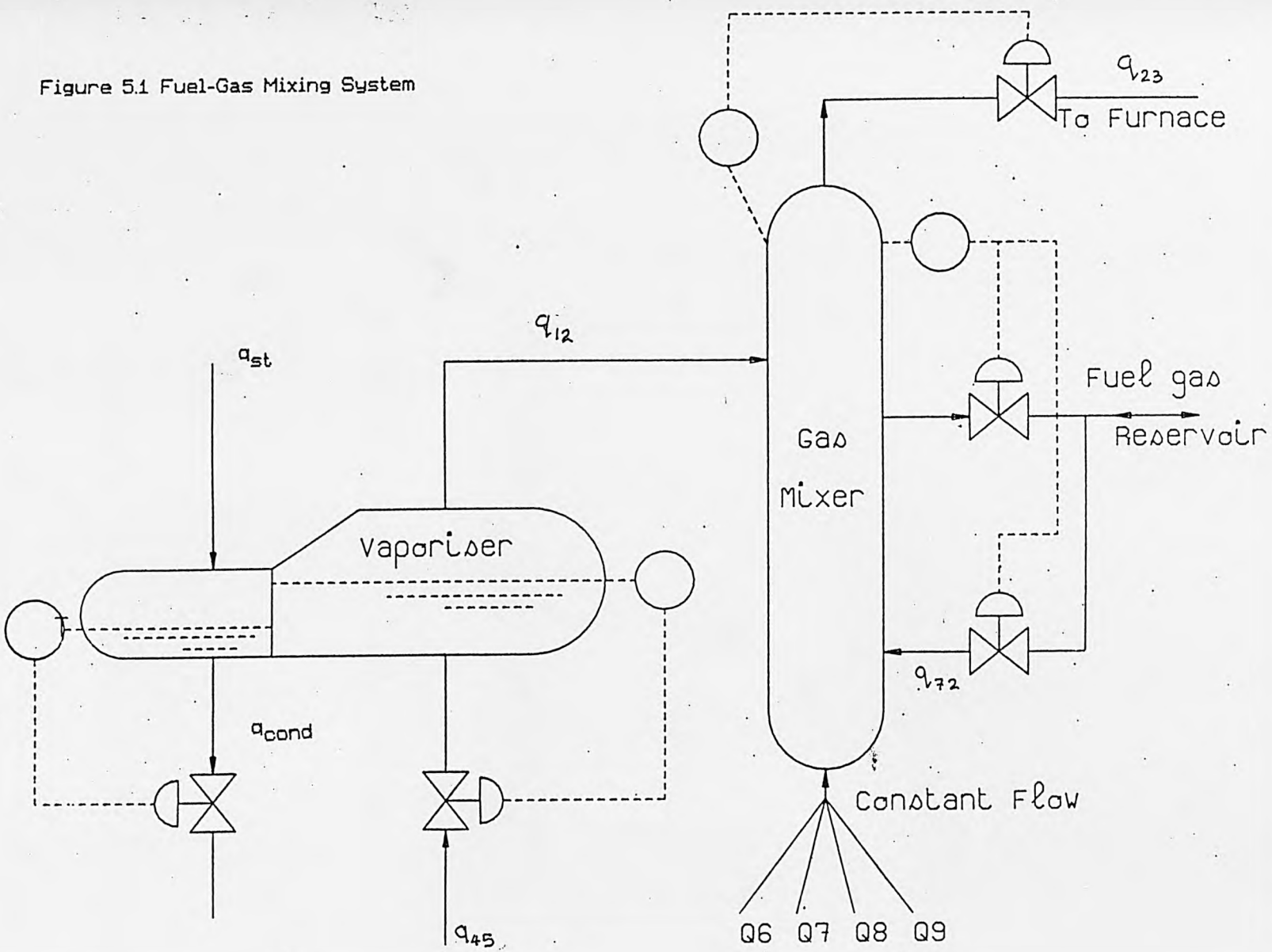
c) that, even though the model may resemble the process physically (ref: example2, model I) a model whose derivatives are closer to that of the process gives better results, even if structurally incorrect.

d) that the stability of the iterative process depends on the model-reality differences, and hence on the derivative differences.

From conclusions (a) and (c), we can draw a subsidiary conclusion that if the derivatives of the model are the same as those of the process, then there is structural equivalence between the two.

So far, we have studied the centralised control structures. In the following chapters some decentralised control structures are also studied and later some generalisations are made for both the centralised and decentralised control structures.

Figure 5.1 Fuel-Gas Mixing System



	MODEL	SYSTEM
FUNCTION VALUE	9.74041	8.899984
ITERATIONS	11	28
SET POINT CHANGES	NIL	2204
SUB-OPTIMALITY %	+9.5%	0%

Table 5.1 Results of Model and Direct Optimisation (Example 1)

	LINEAR MODEL	NON-LINEAR MODEL I	NON-LINEAR MODEL II
PARAMETERS			
α_1	-0.288050	-0.287674	0.545366
α_2	0.362133	0.297724	-0.208757
α_3	0.118695	0.573585	0.343833
FUNCTION VALUE	8.899984	8,900207	8.901966
ITERATIONS	27	150	240
SET POINT CHANGES	162	900	1440
ITERATIVE LOOP GAINS	0.4	0.2	0.1

Table 5.2 Results of Modified Two-step Approach Using Different Models (Example 1)

FUNCTION VALUE	8.899983
ITERATIONS	5
SET POINT CHANGES	25
ITERATIVE LOOP GAINS	1.0

Table 5.3 Results of Approximate Linear Model Approach (Example 1)

	LINEAR MODEL	NON-LINEAR MODEL I	NON-LINEAR MODEL II
$\frac{\partial y_1}{\partial c_1} - \frac{\partial y_{1*}}{\partial c_1}$	0.504893	0.5044164	-1.7655088
$\frac{\partial y_1}{\partial c_1} - \frac{\partial y_{2*}}{\partial c_1}$	0.5798188	0.5796372	-1.4792441
$\frac{\partial y_3}{\partial c_1} - \frac{\partial y_{3*}}{\partial c_1}$	0.0752295	0.0752208	-0.0302481
$\frac{\partial y_1}{\partial c_2} - \frac{\partial y_{1*}}{\partial c_2}$	0.0134622	0.0135680	1.857903
$\frac{\partial y_2}{\partial c_2} - \frac{\partial y_{2*}}{\partial c_2}$	-0.0358647	-0.0357536	1.6250462
$\frac{\partial y_3}{\partial c_2} - \frac{\partial y_{3*}}{\partial c_2}$	-0.0493269	-0.0493216	0.0424178
$\frac{\partial y_1}{\partial c_3} - \frac{\partial y_{1*}}{\partial c_3}$	-0.0270148	-0.0270400	-0.9853911
$\frac{\partial y_2}{\partial c_3} - \frac{\partial y_{2*}}{\partial c_3}$	-0.0033655	-0.0033920	-0.7627929
$\frac{\partial y_3}{\partial c_3} - \frac{\partial y_{3*}}{\partial c_3}$	0.0236493	0.0236480	0.0491888
$\frac{\partial y_1}{\partial c_4} - \frac{\partial y_{1*}}{\partial c_4}$	0.3744024	0.6807456	3.6513370
$\frac{\partial y_2}{\partial c_4} - \frac{\partial y_{2*}}{\partial c_4}$	0.1201814	0.2733653	3.0626024
$\frac{\partial y_3}{\partial c_4} - \frac{\partial y_{3*}}{\partial c_4}$	-0.1277567	0.2091646	0.5815669
$\frac{\partial y_1}{\partial c_5} - \frac{\partial y_{1*}}{\partial c_5}$	0.2518604	1.6443657	1.5300769
$\frac{\partial y_2}{\partial c_5} - \frac{\partial y_{2*}}{\partial c_5}$	0.1634940	0.8596336	0.7614149
$\frac{\partial y_3}{\partial c_5} - \frac{\partial y_{3*}}{\partial c_5}$	0.1462462	0.3781696	0.3615372

Table 5.4 Derivative Differences When Convergence is Obtained (Example 1)

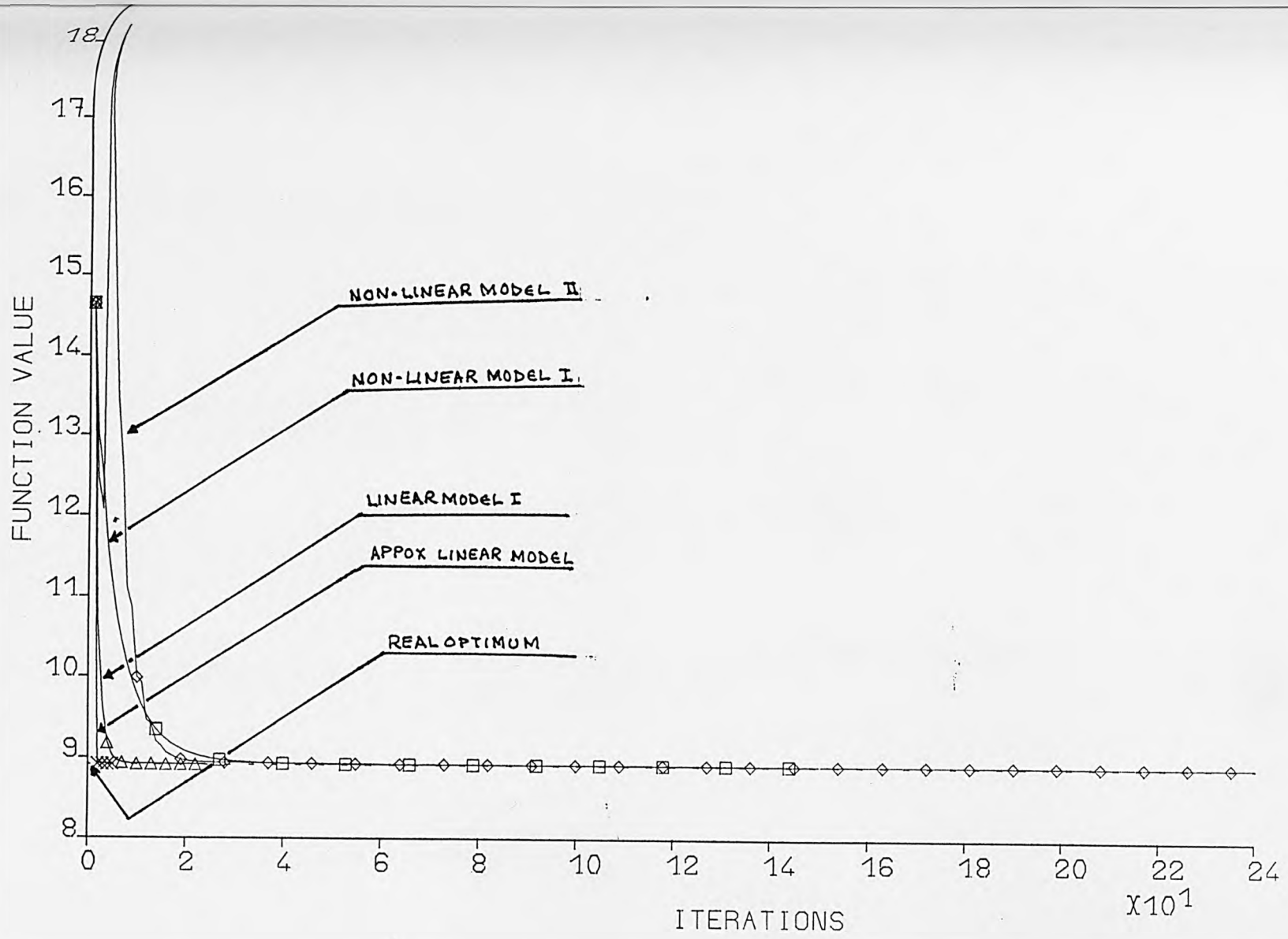


Figure 5.2 Performance Index Value Vs Iterations (Example 1)

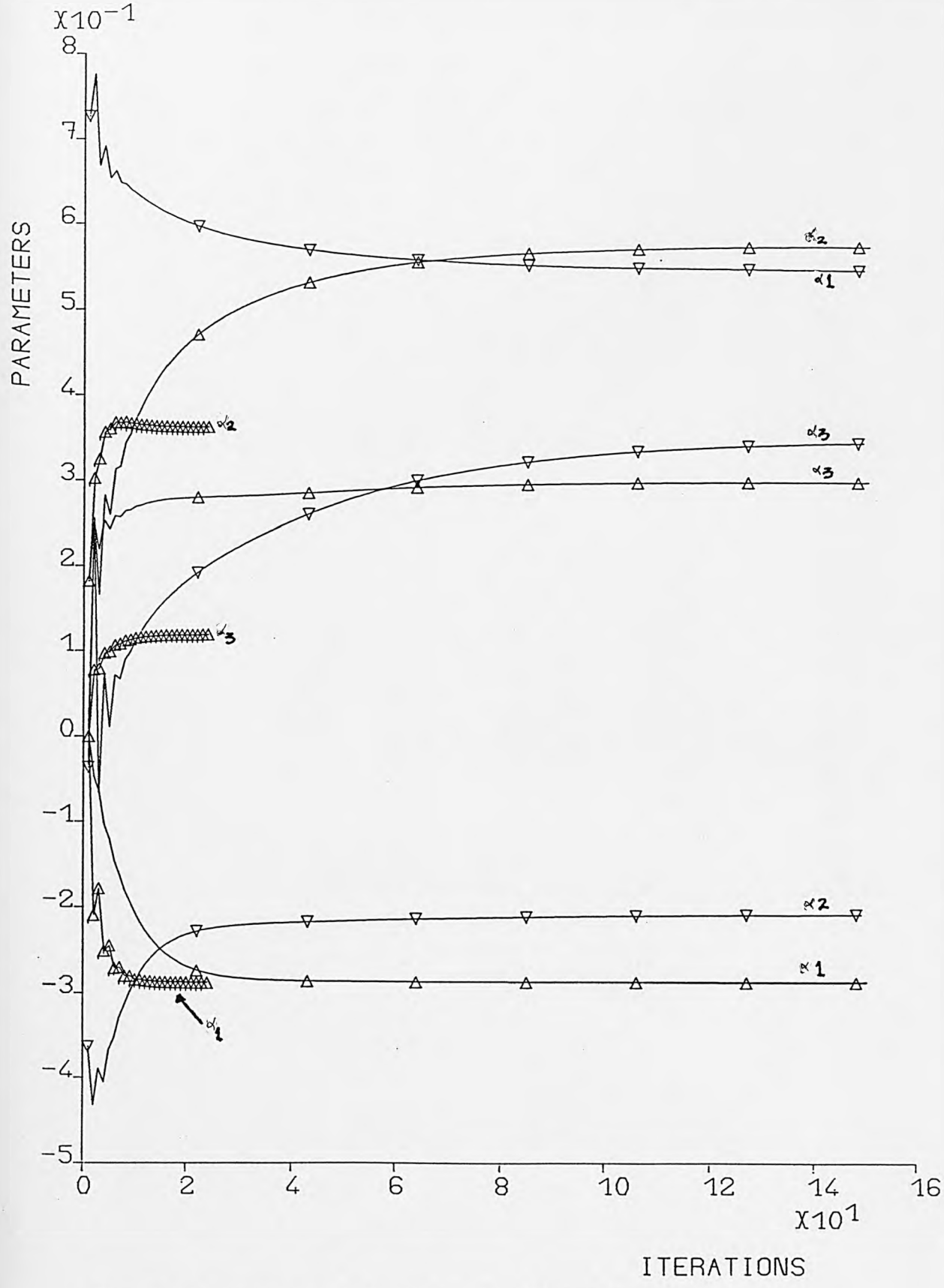


Figure 5.3 Parameters Estimated Vs Iterations (Example 1)

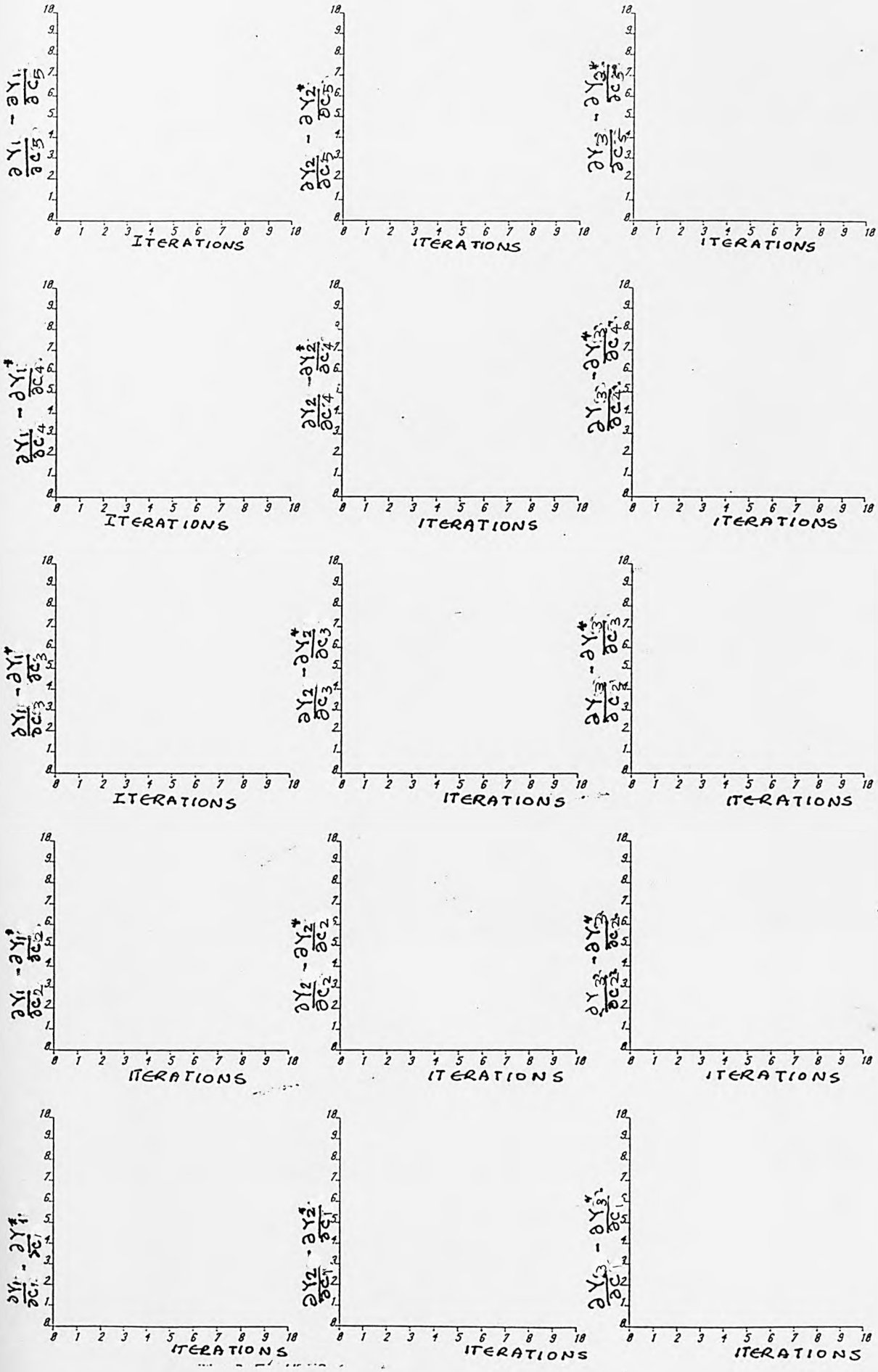


Figure 5.4 Key to Figures 5.5 — 5.8

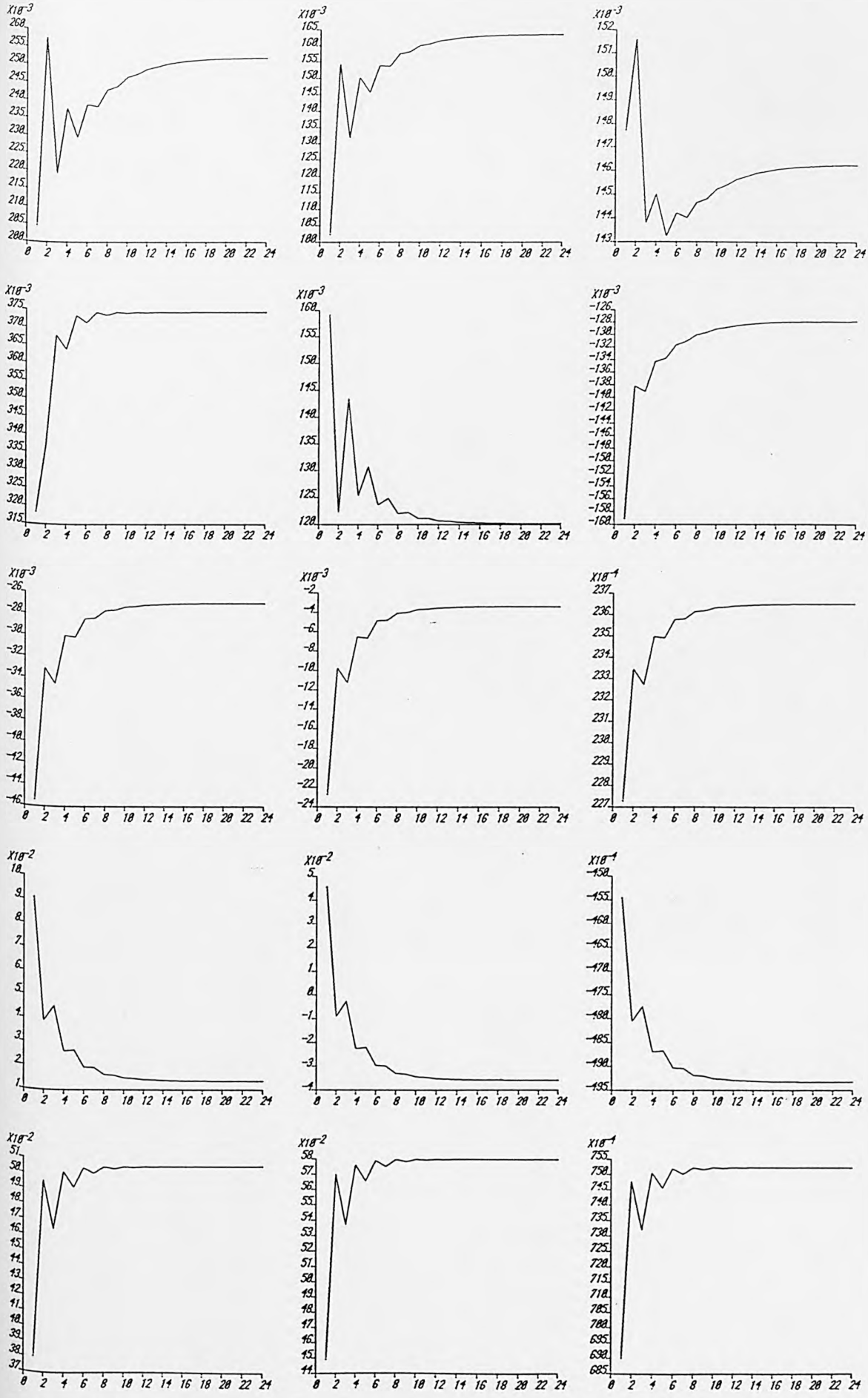


Figure 5.5 Derivative Differences Vs Iterations (Example 1; Linear Model)

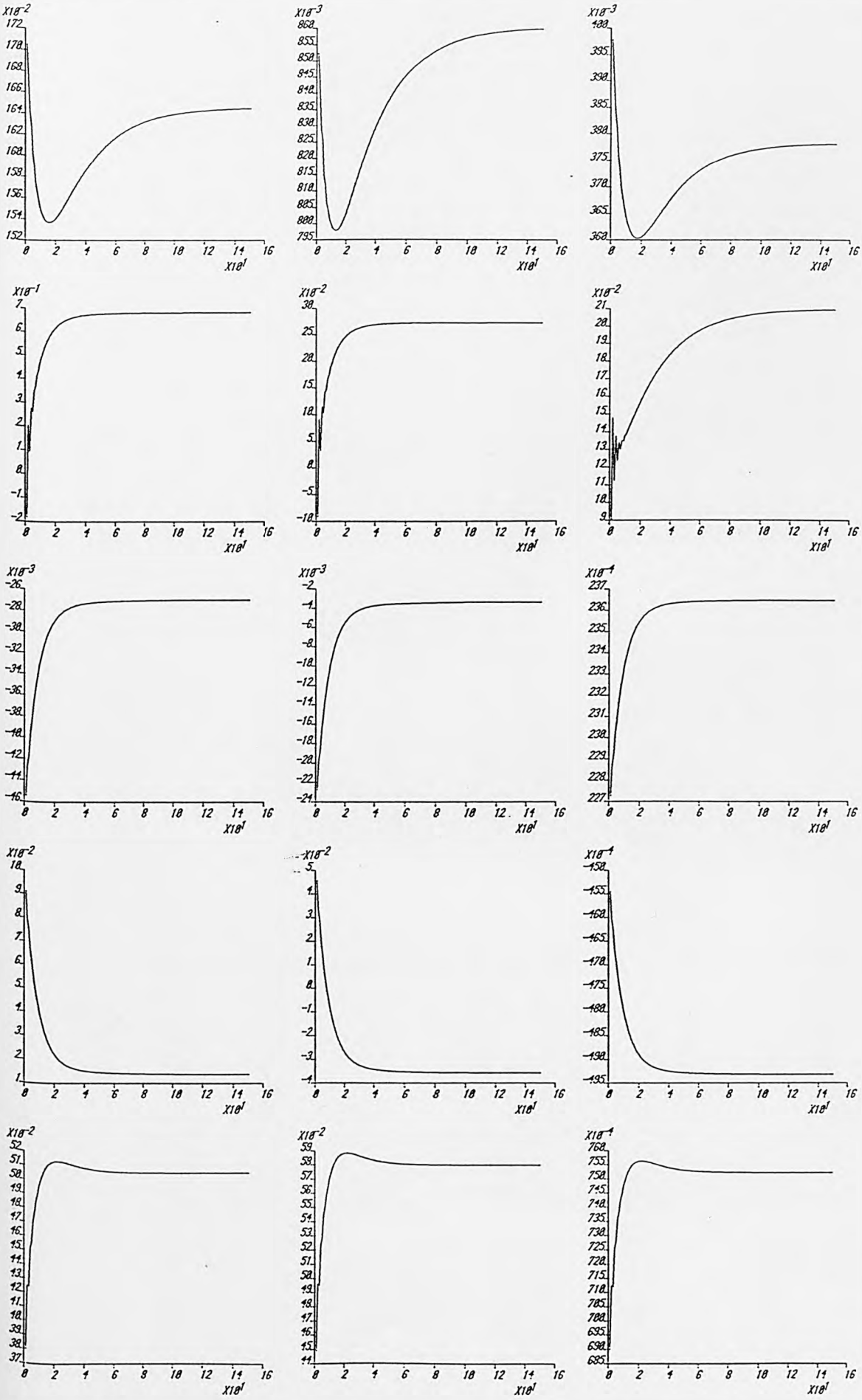


Figure 5.6 Derivative Differences Vs Iterations (Example 1; Non-Linear Model I)

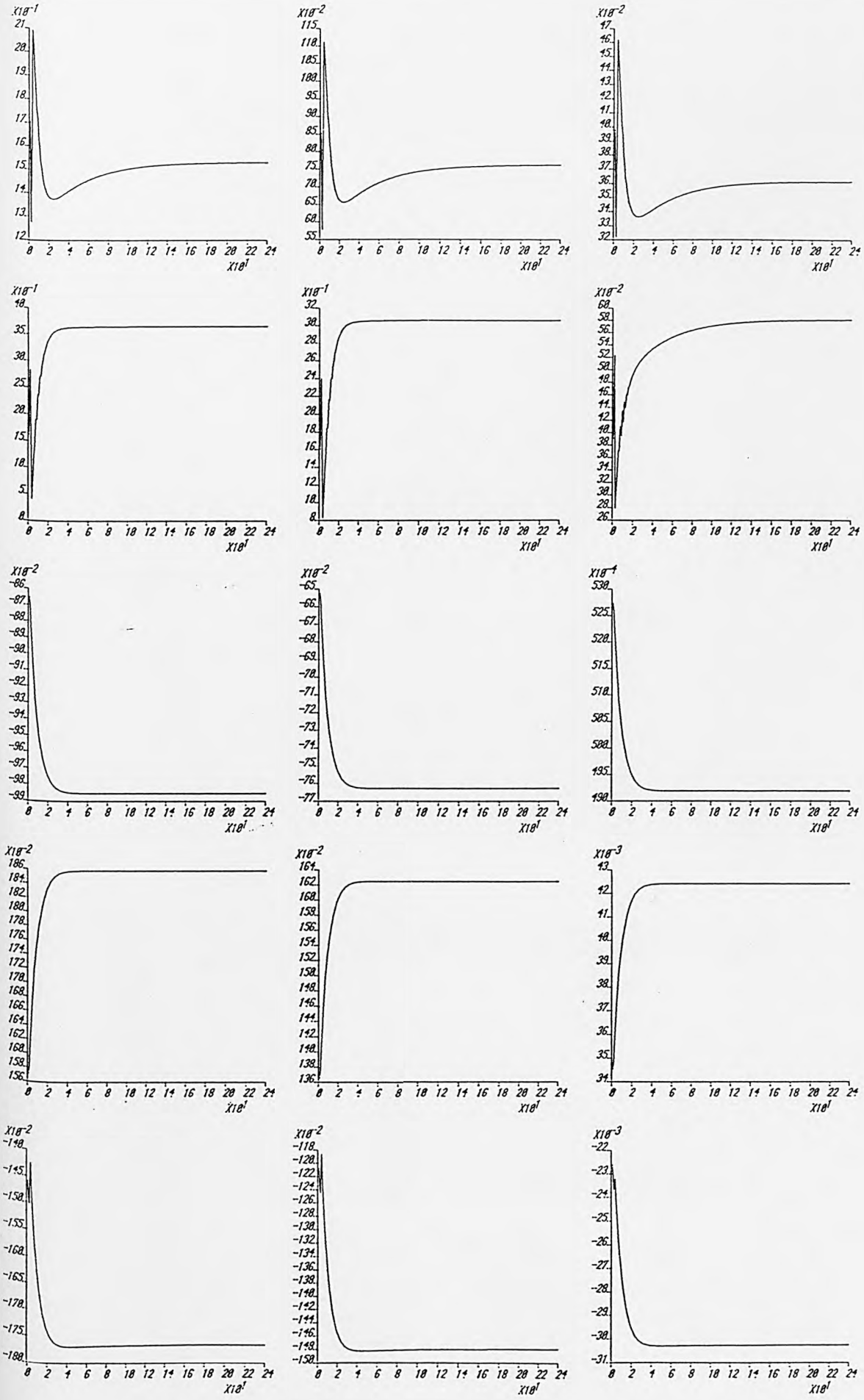


Figure 5.7 Derivative Differences Vs Iterations (Example 1; Non-Linear Model II)

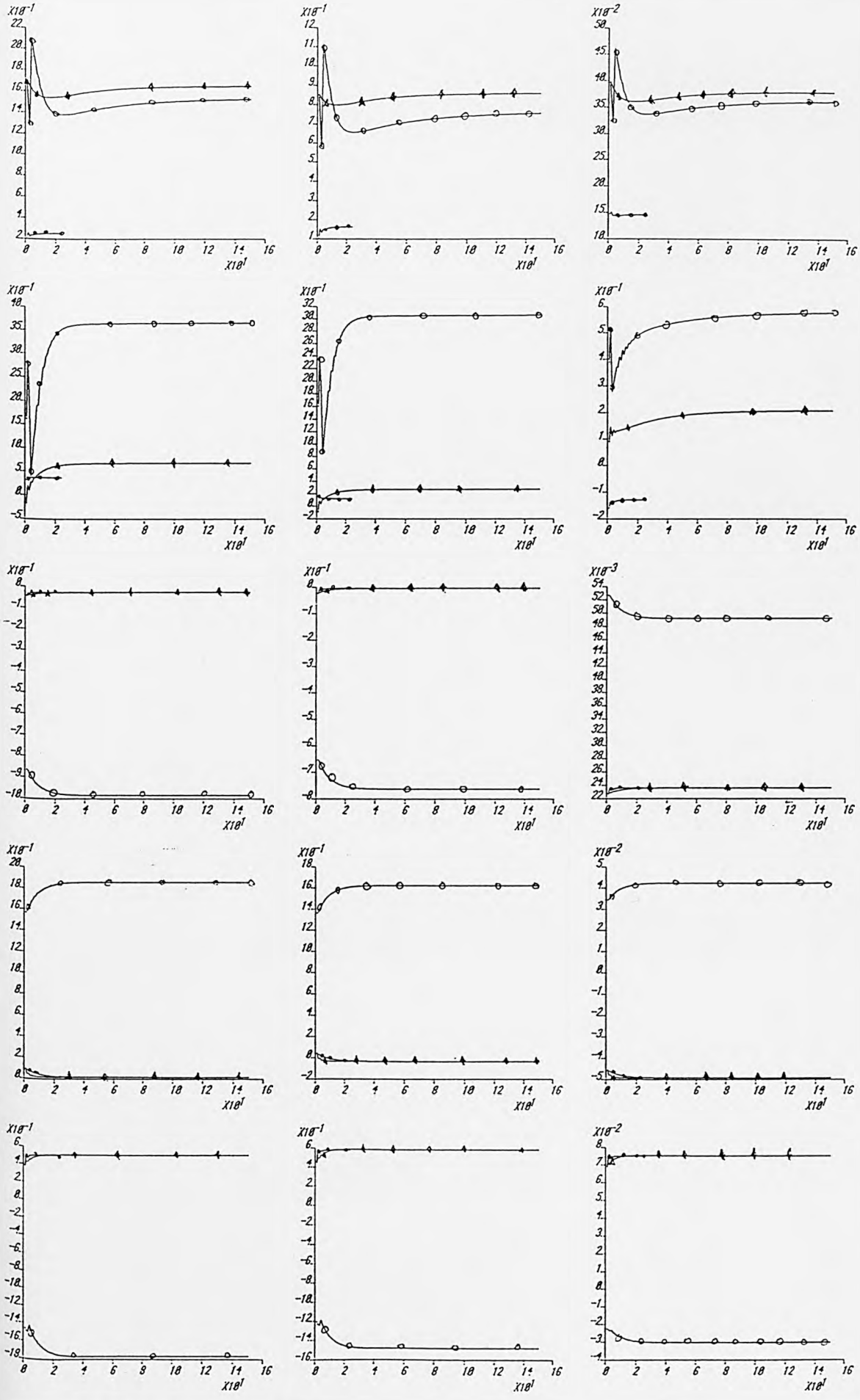


Figure 5.8 Consolidation of figures 5.5 to 5.7 (

FUNCTION VALUE	-199.4
ITERATIONS	5
SET POINT CHANGES	356

Table 5.5 Converged Results of Direct Optimisation (Fuel Gas System)

	NON-LINEAR MODEL	LINEAR MODEL
PARAMETERS		
α_1	12.95	-10.10013
α_2	3.567	9.02210
α_3	5.764	0.99239
FUNCTION VALUE	-199.4	-199.4
ITERATIONS	150	7
SET POINT CHANGES	600	28
ITERATIVE LOOP GAINS	0.1	0.9

Table 5.6 Converged Results of MSTP Using Different Models (Fuel Gas System)

FUNCTION VALUE	-199.4
ITERATIONS	5
SET POINT CHANGES	15
ITERATIVE LOOP GAINS	1.0

Table 5.7 Converged Results of ALM-ISOPE (Fuel Gas System)

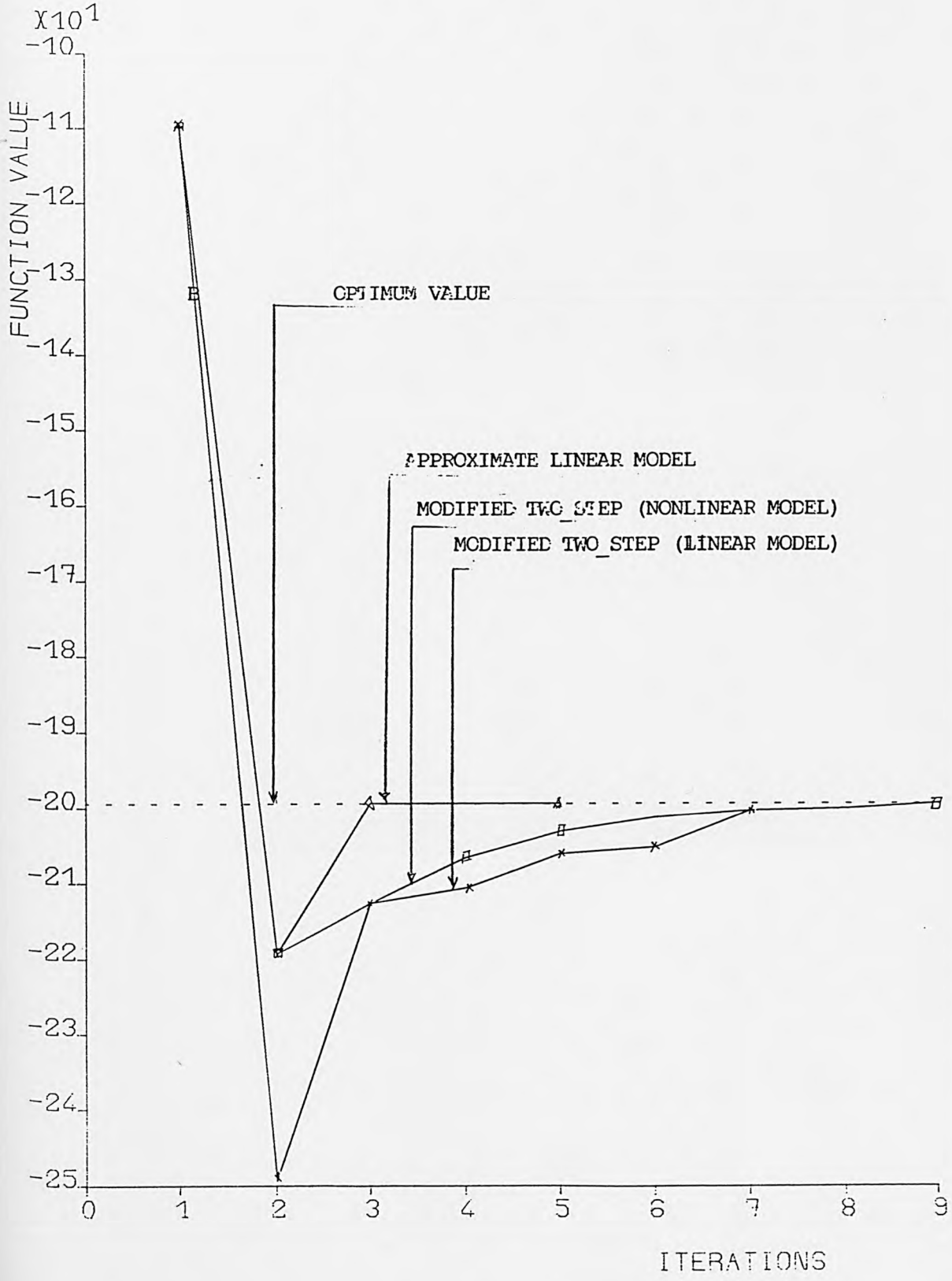


Figure 5.9 Performance Index Value Vs Iterations (Fuel-Gas Mixing System)

Non-Linear Model

Linear Model

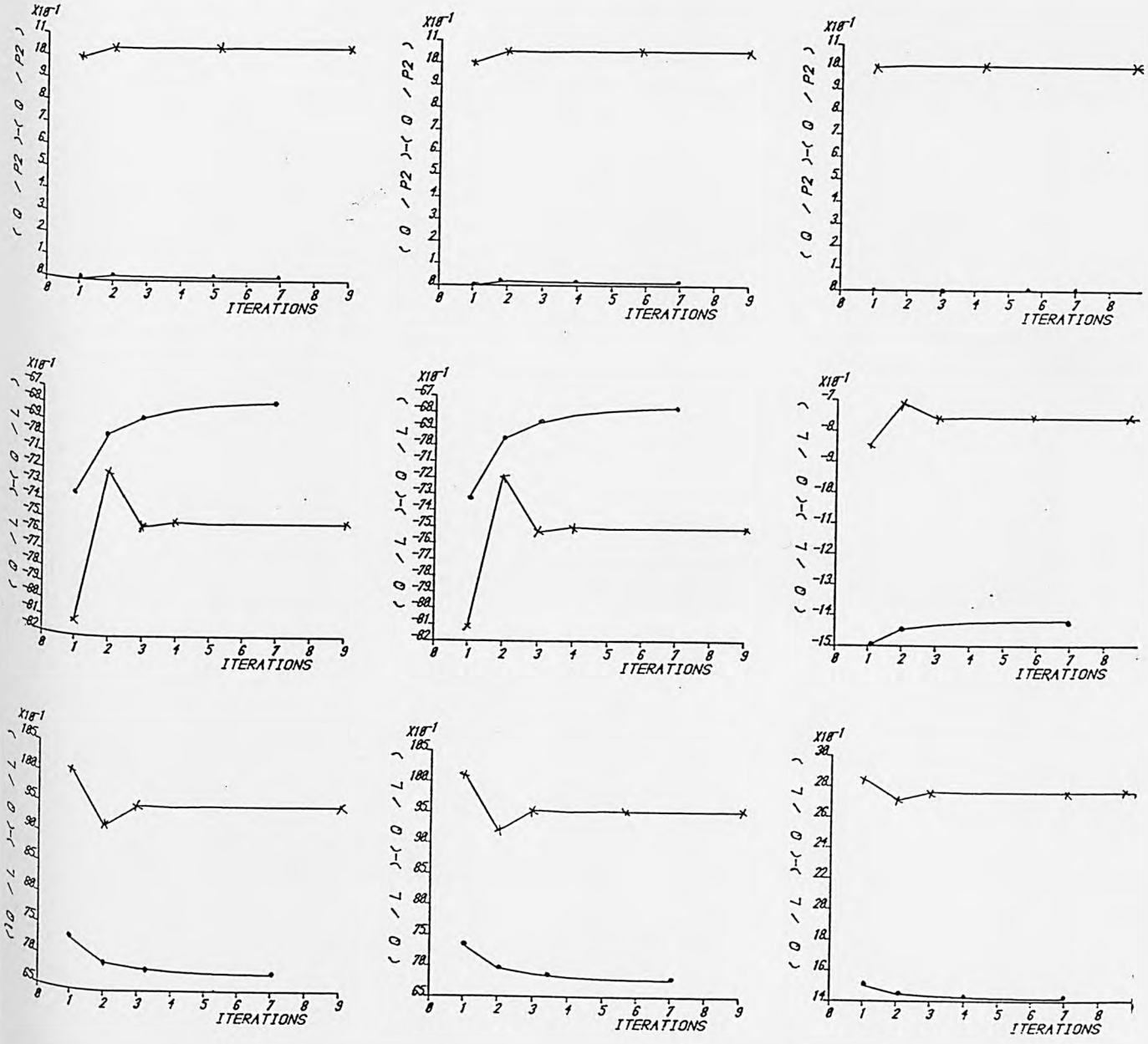


Figure 5.10 Derivative Differences Vs Iterations (Fuel-Gas Mixing System)

	LINEAR MODEL	NON-LINEAR MODEL I
FUNCTION VALUE	8.899	8.999
ITERATIONS	24	75
SET POINT CHANGES	144	450
ITERATIVE LOOP GAINS	0.5	0.4
PENALTY FACTOR ρ	2.0	2.0

Table 5.8 Results of the AISOPE Method (example 1)

CHAPTER 6 DECENTRALISED CONTROL STRUCTURES

6.1 INTRODUCTION

So far, we have studied centralised control structures and their associated algorithms. In centralised control all the information needed to make a control decision is taken to and processed at a central location. However, there are essentially two difficulties in solving problems of large scale systems using the techniques developed so far. The first is numerical, and arises due to high dimensionality caused by an increase in system dimensions. That is, there are too many control decisions to be made. The second is that there may be a lack of centrality in a system. Then, consider a system having the following characteristics:

a) A large system consisting of a number of sub-systems and a large number of controls (high dimesionality).

b) The whole system is spread over a large area. That is, there is a geographical dispersion of the sub-systems so that it is not possible to bring all the available information to a central point so that a control decision can be taken.

It may then be necessary to process the information, so that a control decision can be taken, locally. This means that we need to establish a decentralised control structure, whereby each sub-system has its own local decision unit taking the control decision locally. Of course this decision has to take into account the effect it has on the other sub-systems. This, then means that there a need for some form of coordination between the decisions taken by the separate local units. The coordination influences these decisions and effectively prevents conflicts between the individual decisions.

The lack of centrality arises in many systems, such as the gas and water systems, large industrial systems distributed all over the world, where due to the geographical dispersion, the cost of centralising the task of taking a control decision may become prohibitive. Thus, sometimes it may be found that establishing a decentralised control strategy may be the only answer to effectively control the system.

The decentralised control problem, then, arises when a system is acted upon

by a number of different controllers which are not able to communicate with each other on-line or are not in a position to process all the information available to them. However, it is assumed that the decision making is cooperative, that is the controllers may use some a priori rules so that some overall system objective is achieved **(Singh, Titli and Malinowski (1983), Findiesen et al (1980))**.

Established algorithms for solving the decentralised problems **(Findiesen et al (1980), Singh and Titli (1978),** Cohen (1978)) are based on 'decomposition and coordination' techniques. These use fixed models, and in general, have been found to produce sub-optimal results. These techniques when integrated with the ISOPE (MSTP) technique produce a whole family of algorithms which are all classified as ISOPE algorithms **(Brdys and Roberts (1986), Michalska et al (1985), Ellis et al (1986), Lin et al (1987a), Tatjewski et al (1987)** .

The ISOPE algorithms for solving the decentralised control problem can be classified as :

a) Dynamic Coordination algorithms **(Brdys and Roberts(1986a), Michalska et al (1985), Lin et al (1987a))**.

b) Passive Coordination algorithms **(Ellis et al(1986), Kambhampati and Ellis (1987))**.

This classification comes about because of the type of coordination task involved in each structure. Algorithms which use the techniques of 'decomposition and coordination' in conjunction with the ISOPE techniques have dynamic coordination. This is because they have to solve an additional coordination problem in an iterative manner so that the final solution is optimal. This additional problem arises because of the manner in which the problems, for these algorithms, have been formulated. This, then, leads to the necessity of solving the dual associated with it. It is in this sense that we term these algorithms dynamic **(Ellis et al (1986))**. Of course, all these algorithms do not have the same behavioural pattern, that is the convergence rate is different. Of these algorithms the one selected for study here is the algorithm which uses both input and output feedback information **(Brdys and Roberts (1986a), Chen (1986))** as this algorithm has been seen to be the most efficient so far **(Roberts et al (1988))**.

It was noted in the previous chapter, concerning the centralised algorithms, that there is a problem in selecting appropriate iterative loop gains. These gains are usually selected on a trial and error basis. However, **Tatjewski and Roberts(1987a)** developed an algorithm which uses a precise Newton-like step for updating the controls. The whole control structure for this algorithm and the

problem formulation remains the same. Only the control updating mechanism changes. This, of course, leads to an increase in the off-line calculations but there is an advantage here in that the algorithm is not very sensitive to the augmentation of the performance index nor is it sensitive to any non-linearities, present in the system. **Tatjewski and Roberts (1987b)** have developed an algorithm, similar to the centralised Newton-like algorithm, for solving the decentralised problem. This algorithm retains most of the basic features of the dyanamic coordination algorithm mentioned above. The difference is in the control updating mechanism.

If we extend the MSTP algorithm so that the decentralised problem can be solved, we find that there is no additional coordination problem to be solved. The sole task of the coordinator is to evaluate the modifier, λ, and this is done using elimentary matrix operations. This, then, would avoid the need for sophisticated mathematical computations to solve the coordination problem. **Ellis et al (1986)** gave an algorithm which incorporated these concepts. However, there was one drawback with this algorithm. It needed two models : an output model and an interaction model. The interaction model is needed to take into account the interactions taking place between the sub-systems. This additional requirement can be avoided if we make use of the information from the system more efficiently. Here, we can use the interaction input measurements directly in the optimisation problem. This has a two fold effect. Namely, the number of independent variables in the problem is reduced and therefore the numerical complexity. Also there is no coordination problem to solve. **Brdys and Roberts(1986a)** were the first to give an algorithm which uses these concepts. However, the interpretation they gave for their resulting control structure is that it was completely decentralised with information exchange between the sub-systems. Such an interpretation would be difficult to implement on-line. If, however, we now add a coordinator, whose task it is to evaluate the modifier and pass the information to the individual decision units, it would then be easy to implement the algorithm on-line. Here, we see that the coordinators task is passive.

In this chapter, we shall describe dynamic and passive forms of the ISOPE algorithms. In the dynamic case, the algorithm will be described with relaxation and Newton-like schemes for updating the controls.

It should be noted here that for these algorithms we shall not make any distinction between the augmented case and the unaugmented case. This is so because augmentation does not make any difference to the analysis, the control structure obtained or the algorithmic structure. All one has to do, to obtain the unaugmented case is to set the penalty factor ρ equal to zero and leave everything else the same. The analysis of the properties of these algorithms and

the simulation results is presented in the following chapters.

6.2 THE DECENTRALISED ALGORITHMS

In chapter 2 we made the assumption, while formulating the OCP, that the performance index is additively separable. In fact, it is this assumption that enables us to develop decentralised control structures. For, when the global performance index is split into different sub-problems, each sub-problem corresponds to each sub-system. This particular property has been used by **Singh and Titli (1978) and Findeisen et al (1980)** to develop a whole range of algorithms. However, these algorithms use fixed models, and in general would not produce optimum results. It has, however, been found that optimal results can be obtained when we combine the concept of the MSTP with these algorithms **(Brdys and Roberts(1986a), Chen (1986), Abdullah (1988))**. When the combination is made, it has been seen that a whole family of algorithms are obtained, each of which produce optimal results. We have already classified these algorithms as algorithms with dynamic coordination and algorithms with passive coordination.

To recap then, see chapter 2, the OCP is as follows:

$$\begin{aligned} &\min_{C,U,Y} \; Q(C,U,Y) \\ &\text{s.t} \\ &\quad Y = F_*(C,U) \\ &\quad U = HY \\ &\quad G(C,U,Y) \leq 0 \end{aligned} \tag{6.1}$$

where $Q(C,U,Y) = \sum_{i=1}^{N} Q_i(C_i, U_i, Y_i)$, and the adaptive model based problem, where the problems of system optimisation and parameter estimation are integrated into one problem is then as follows :

$$\begin{aligned} &\min_{C,U,Y} \; Q(C,U,Y) \\ &\text{s.t} \\ &\quad Y = F(C,U,\alpha) \\ &\quad F(C,U,\alpha) = K_*(C) \\ &\quad U = HY \\ &\quad G(C,U,Y) \leq 0 \end{aligned} \tag{6.2}$$

The separability of the problem into N different sub-problems, corresponding to N different sub-systems, is still maintained even while using the problem given by (6.2). The detailed description of this separation is described later. It is this problem which is used in describing the three algorithms in this chapter.

6.2.1 Algorithms with Dynamic Coordination

Here in this section, the algorithms with dynamic coordination are derived. It has already been mentioned that the difference between the relaxed algorithm and the Newton-like algorithm is only in the mechanism for control updating. Hence, we shall derive the general control structure for both the algorithms, together and later describe the updating scheme in detail.

Notice that in the problem described in equ (6.2) only the output feedback information is used in the problem. If we use the total system information available to us, we can transform the above problem into the following **(Brdys and Roberts (1986a))**:

$$\begin{aligned} &\min_{C,U,Y} \; Q(C,U,Y) \\ &\text{s.t} \\ &\quad Y = F(C,U,\alpha) \\ &\quad U = HY \\ &\quad F(C,HK_{\sharp}(C),\alpha) = K_{*}(C) \\ &\quad G(C,U,Y) \leq 0 \end{aligned} \tag{6.3}$$

We now have to transform this problem into its equivalent expanded form, by introducing the relevant new variables and augmenting the performance index with the relevant quadratic penalty functions. Then if, as before, we introduce new variables $V \in \Re^n$, $V = C$ into the problem, and append the performance index by the following function, $\rho \left\{ \|V-C\|^2 + \|U-HK_{\sharp}(V)\|^2 \right\}$, we get the following equivalent problem:

$$\begin{aligned} &\min_{C,U,V,\alpha} \; q(C,U,\alpha) + \rho \left\{ \|V-C\|^2 + \|U-HK_{\sharp}(V)\|^2 \right\} \\ &\text{s.t} \\ &\quad U = HY \\ &\quad F(V,HK_{\sharp}(V),\alpha) = K_{*}(V) \\ &\quad g(C,U) \leq 0 \\ &\quad V = C \end{aligned} \tag{6.4}$$

where as before $q(C,U,\alpha) = Q(C,U,F(C,U,\alpha))$ and $G(C,U) = g(C,U)$

An analysis of the first order optimality conditions for (6.4), in a similar manner to chapter 3, **(Brdys and Roberts (1986a))** would give an equivalent problem which is composed of the following three sub-problems:

Parameter Estimation Problem (DCPEP)

$$F(V,HK_{\sharp}(V),\alpha) = K_{\sharp}(V) \dashrightarrow \alpha \tag{6.5}$$

Modified Optimisation Problem (DCMOP)

$$\min_{C,U} \left\{ q(C,U,\alpha) - \lambda^T C + P^T(U - HF(C,U,\alpha) + \rho \left\{ \|V-C\|^2 + \|U-HK_{\sharp}(V)\|^2 \right\} \right\}$$

s.t (6.6)

$$g(C,U) \leq 0$$

where λ is the modifier

The vector $P \in \Re^m$ is known as the price vector, which is obtained by solving the dual of (6.4) in the coordinator and compensates for the interactions between the various sub-systems.

Coordination Problem (DCCP)

$$U = HF(C,U,\alpha) \tag{6.7}$$

$$V = C \tag{6.8}$$

Evaluate the modifier λ, given by

$$\lambda = [\ F'_V(V,HK_{\sharp}(V),\alpha) - K'_{\sharp}(V)\ ]^T\ [\ Q'_Y(V,HK_{\sharp}(V),F(V,HK_{\sharp}(V),\alpha) - H^T P\ ] \tag{6.9}$$

6.2.1.1 Decomposability and the resultant control structure

Now, let us consider the decomposability of the DCMOP. The function which is to be optimised consists of four different components. Let us then consider each of these parts and consider the decomposability of each:

a) The function $q(C,U,\alpha)$ is decomposable into the N different sub-functions, because of the assumption of additive separabilty.

b) The function $\lambda^T C$, is decomposable, and can be written as follows

$$\lambda^T C = \sum_{i=1}^{N} \lambda_i C_i \tag{6.10}$$

c) Similarly the function $P^T(U - HF(C,U,\alpha))$ is also decomposable. This can be written as

$$P^T(U - HF(C,U,\alpha)) = \sum_{i=1}^{N} \left(P_i U_i - \sum_{j=1}^{N} H_{ji} F(C,U,\alpha) \right) \tag{6.11}$$

d) Finally the last function, which is a quadratic funciton is easily decomposable, for

$$\rho\left\{\|V-C\|^2 + \|U-HK_*(V)\|^2\right\} = \rho\left\{\sum_{i=1}^{N}\left\{(V_i - C_i)^2 + (U_i - H_iK_*(V))^2\right\}\right\} \qquad (6.12)$$

Of course the parameter estimation problem is also separable into its different problems, each corresponding to a sub-system. Then the problem for the i^{th} Local Decision Unit (LDU) , for a given V_i, α_i, and λ_i, is as follows:

Parameter estimation problem:

$$F_i(C_i,U_i,\alpha_i) = Y_{*i} \qquad (6.13)$$

The modified optimisation problem:

$$\min_{C_i,U_i}\left\{q_i(C_i,U_i,\alpha_i)-\lambda_iC_i+P_iU_i-\sum_{j=1}^{N}H_{ji}F(C,U,\alpha)+\rho\left\{(V_i - C_i)^2 + (U_i - U_{*i})^2\right\}\right\}$$
$$\text{s.t} \quad g_i(C_i, U_i) \leq 0 \qquad (6.14)$$

The overall control structure is shown in figure 6.1

6.2.1.2 The algorithmic structure

So far the problem formulation is common to both the relaxed and the Newton-like algorithms. However, now the difference in these algorithms occurs because of the way in which the controls and the price vectors are updated. However, the general algorithmic and control structures remain the same for both the algortihms. The algorithmic structure then is as follows:

STEP I : Select C^0, U^0, P^0

select $\varepsilon > 0$,where ε is the accuracy needed.

Set $k \leftarrow 0$

Set $V^k = C^k$

STEP II : Apply the controls V^k to the system and take measurements and evaluate the parameters α

STEP III : Apply further perturbations about V^k and take measurements to obtain finite difference approximations to $K'_*(V)$

STEP IV : Evaluate the modifier given by (6.9)

STEP V : Solve the DCMOP to obtain $\hat{C}^{k+1}$ and $\hat{U}^{k+1}$

STEP VI : Check IF $\hat{U}^{k+1} = HF(\hat{C}^{k+1},\hat{U}^{k+1},\alpha^k)$ and $\|V^k - \hat{C}^{k+1}\| < \varepsilon$ THEN STOP

ELSE

Set $k \leftarrow k+1$

Set $V^k = \hat{C}^k$ (6.15)

Update price vector P^{k+1}

GO TO STEP II

6.2.1.3. Relaxed Algorithm

If the algorithm is implemented in the manner described above, it is likely that the iterative process will be unstable (See Chapter 5). In order to introduce an element of stability into the algorithm, a scheme known as under-relaxation **(Cohen (1978))** is used. Here, instead of going the whole length in updating controls, only a small step is taken. Then (6.15) is replaced by the following:

$$V^k = V^{k-1} + \xi_V(\hat{C}^k - V^{k-1}) \tag{6.16}$$

where $\xi_V \in \Re^n$ and $0 < \xi_V \leq 1$ and is termed the iterative loop gain.

The interactions are taken into account by the price vector which needs to be updated by the coordinator at the end of every iteration. Here we use a simple gradient type mechanism to do so :

$$P^k = P^{k-1} + \xi_P\left(\hat{U}^k - HF(\hat{C}^k,\hat{U}^k,\alpha^k) \right) \tag{6.17}$$

where $\xi_P > 0$ and $\xi_P \in \Re^m$

6.2.1.4 Newton-Like Algorithm

In the relaxed algorithms the selection of the scalars ξ_V and ξ_P is done by trial and error. This, then, would pose a problem while implementing the algorithm on-line. The Newton-like algorithm does away with the need for selecting these scalars by defining an exact step lenght to take. Of course the computational complexity would increase, but this is offset by the fact **(Tatjewski and Roberts (1987b))** that this algorithm converges at a faster rate then the other algorithm. Also, the algorithm is less sensitive to the non-linearities in the system. There is a definite advantage if the performance index is augmented by a quadratic penalty function. However, we cannot define the exact value for the penalty factor ρ. It is found that as the value of this factor increases the performance is definitely bettered but beyond a certain value the performance of the algorithm deteriorates. The Newton-like algorithm solves this problem, for whatever be the value of the penalty factor the performance of the algorithm does not deteriorate.

The updating of both the controls and the price vector can be viewed as solving the following operator equation:

$$\Psi(V,P) = \begin{bmatrix} V - \hat{C}(V,P) \\ U - HF(\hat{C}(V,P),\hat{U}(V,P)\hat{\alpha}(V)) \end{bmatrix} = 0 \tag{6.18}$$

let $t(C,U,\alpha) = U - HF(C,U,\alpha)$

Then the Newton-like updation scheme for solving (6.18) would be:

$$\begin{bmatrix} V^{k+1} \\ P^{k+1} \end{bmatrix} = \begin{bmatrix} V^k \\ P^k \end{bmatrix} - \left[\Psi'(V^k,P^k)\right]^{-1} \begin{bmatrix} V^k - \hat{C}(V^k,P^k) \\ t(\hat{C}(V^k,P^k),\hat{U}(V^k,P^k),\hat{\alpha}(V^k)) \end{bmatrix} \tag{6.19}$$

provided $\Psi'(V^k,P^k)$ is non-singular.

For purposes of notational simplicity, $\hat{C}(V^k,\hat{P}^k)$ is replaced by $\hat{C}$; $\hat{U}(V^k,P^k)$ by $\hat{U}$ and $\hat{\alpha}(V^k)$ by $\hat{\alpha}$

$$\Psi'(V,P) = \begin{bmatrix} [I \quad O_{nxr}] - \hat{C}' \\ t'_x(\hat{X},\hat{\alpha})\hat{X}' + t'_x(\hat{X},\hat{\alpha})\begin{bmatrix} \hat{\alpha}' & O_{rxr} \end{bmatrix} \end{bmatrix} \tag{6.20}$$

where $X = (C,U) = (C_1,U_1,C_2,U_2,\cdots\cdots,C_N,U_N)$. It is important to note the ordering here.

Let

$$D_* = F'_V(V,HK_*(V),\alpha) - K'_*(V) \tag{6.21}$$

$$Z_* = (V_1,U_{*1},\cdots\cdots,V_N,U_{*N})$$

$$Z'_* = \begin{bmatrix} I \vdots O_{n_1x(n-n_1)} \\ U'_{*1} \\ \vdots \\ O_{n_Nx(n-n_N)} \vdots I \\ U'_{*N}(V) \end{bmatrix} \tag{6.22}$$

$$U_* = HK_*(V)$$

$$E_\rho = A^{-1} - A^{-1}(g'_a)^T\left[g'_aA^{-1}g'_a{}^T\right]^{-1}g'_aA^{-1} \tag{6.23}$$

g_a is the set of constraints which are active.

$$A = L''_{xx}(X,\alpha,P,\mu) + \rho I \tag{6.24}$$

$$L(X,\alpha,P,\mu) = q(C,U,\alpha) + P^T t(C,U,\alpha) + \mu^T g(C,U) \tag{6.25}$$

$$\bar{\lambda} = (\lambda_1, 0_{1\times m_1}, \cdots\cdots, \lambda_N, 0_{1\times m_N}) \tag{6.26}$$

Now, for sake of notational simplicity we shall denote $\hat{X}(V,P)$ by $\hat{X}$ and $\hat{\alpha}(V)$ by α.

The first order optimality conditions for DCMOP are

$$\begin{aligned} q'_x(X,\alpha)^T + \rho(X-Z_*) + t'_x(X,\alpha)^T P - \bar{\lambda} + g'_a{}^T \mu = 0 \\ g_a = 0 \end{aligned} \tag{6.27}$$

<u>Theorem 6.2.1.4.1</u>

Assume that

(1) The strict complementarity condition is satisfied at the optimal point $\hat{X}=(\hat{C}, \hat{U})$ of the OCP

(2) The constraints $G(X) \leq 0$ are regular at the optimal point.

Then, provided ρ is sufficiently large, there is a neighbourhood $N(\hat{V}, \hat{P})$ of the point $(\hat{V}, \hat{P})$ such that for each $(V, P) \in N(\hat{V}, \hat{P})$ the solution of the set of equations (6.27) is a locally unique minimising point-multiplier pair $(\hat{X}, \hat{\mu}_a)$ for the optimisation problem (6.6). Moreover $\hat{X}(.)$ is continuously differentiable, and

$$\hat{X}' = E_\rho\left\{ \rho Z'_* + \bar{\lambda}'_r - \left[q''_{x\alpha}(\hat{X},\alpha) + \sum_{j=1}^{m} P_j (t'_x(\hat{X},\alpha)'_\alpha \right] \alpha' \;\vdots\; -t'_x(\hat{X},\alpha)^T + \bar{\lambda}_P \right\} \tag{6.28}$$

<u>Proof</u>

The Jacobian matrix of the set of equ (6.27) with respect to X and μ_a as dependent variables, is

$$J(X,\alpha,P,\mu) = \begin{bmatrix} L''_{xx}(X,\alpha,P,\mu) + \rho I & g'_a{}^T \\ g'_a & 0 \end{bmatrix} \tag{6.29}$$

where all the components of the vector μ which are not contained in μ_a are equal to zero. Let $|\nu_*|$ be the smallest eigenvalue of the matrix $L''_{xx}(\hat{X},\alpha,P,\hat{\mu})$. Then it can easily be shown that the Jacobian $J(X,\alpha,P,\mu)$ is non-singular provided $\rho > |\nu_*|$. The Jacobian is non-singular iff the set of equations $J(\hat{X},\alpha,P,\hat{\mu})=0$ has only the zero solution $s = 0$. Suppose $s \neq 0$. Then denoting $s = (t,w)$, we have

$$At + g'_a{}^T w = 0 \tag{6.30}$$

$$g'_a t = 0 \tag{6.31}$$

Suppose $t \neq 0$, then multiplying (6.30) by t^T and using (6.31) we get

$$t^T A t = 0$$

which is a contradiction, since $\rho > |\nu_*|$ and thus the matrix A is positive definite.

Thus $t = 0$, then $g'_a{}^T w = 0$ also implies that $w = 0$. Hence the Jacobian $J(X,\alpha,P,\mu)$ is non-singular.

We can now apply the implicit function theorem to (6.27), and restricting the analysis to such a neighbourhood of $\hat{X}$ that changes of activity of the constraints do not occur. Thus, there are neighbourhoods $N_1(\hat{V}, \hat{P})$ and $N(\hat{X}, \hat{\mu}_a)$ such that $(\hat{X}(V,P),\hat{\mu}_a(V,P)) \in N(\hat{X}, \hat{\mu}_a)$ if $(V,P) \in N_1(\hat{V}, \hat{P})$ and the mappings $\hat{X}(.)$, $\hat{\mu}(.)$ are unique, continuous and continuously differentiable in $N_1(\hat{V}, \hat{P})$. Since all elements of the matrix $L''_{xx}(.,.,.)$ are continuous functions of (V, P) in $N_1(\hat{V}, \hat{P})$, then so also are its eigenvalues. Let $N(\hat{V}, \hat{P}) = \text{int } \overline{N}$, where $\overline{N} \in N_1(\hat{V}, \hat{P})$ is some closed subset satisfying $(\hat{V}, \hat{P}) \in \text{int } \overline{N}$, and denote

$$\overline{\rho} = \max_{1 \le j \le n} \left\{ \min_{(V, P) \in \overline{N}} | \nu_j(V,P) | \right\}$$

where $\nu_j(V, P)$, $j=1, 2, 3, \cdots, n$, are eigenvalues of $L''_{xx}(.,.,.)$. Then, for $\rho > \overline{\rho}$, the Hessian matrix A of the performance index is positive definite at each $(\hat{X}, \hat{\mu}_a)$, $(V, P) \in N(V, P)$, and the second order sufficient optimality conditions are satisfied at these points.

It follows from the implicit function theorem that

$$\begin{bmatrix} \hat{X}' \\ \hat{\mu}_a' \end{bmatrix} = -J(\hat{X},\alpha,P,\hat{\mu}).$$

$$\begin{bmatrix} \left[q''_{x\alpha}(\hat{X},\alpha) + \sum_{j=1}^{m} P_j (e'_x(\hat{X},\alpha)^T{}_j)'_\alpha \right] \alpha' - \rho Z'_* - \bar{\lambda}'_v & e'_x(\hat{X},\alpha)^T - \bar{\lambda}_P' \\ 0 & 0 \end{bmatrix} \quad (6.32)$$

Evaluating the inverse of the Jacobian in terms of A and g_a we get (6.29).

Q. E. D.

We can simplify (6.27) further. Now we have

$$q'_x(X,\alpha) = Q'_x(X,F(X,\alpha)) + Q'_Y(X,F(X,\alpha))F'_x(X,\alpha) \quad (6.33)$$

then

$$q''_{x\alpha}(X,\alpha) = Q''_{x\alpha}(X,F(X,\alpha))F'_\alpha(X,\alpha) + F'_x(X,\alpha)^T Q''_{YY}(X,F(X,\alpha))F'_\alpha(X,\alpha) + \sum_{j=1}^{m} Q'_Y(X,F(X,\alpha))_j (F'_x(X,\alpha)^T{}_j)'_\alpha \quad (6.34)$$

$$\text{also } \lambda'_P = -D_*{}^T H^T \quad (6.35)$$

Now

$$\lambda'_v = D_*{}^T \left[Q''_{YC} + Q''_{YU} H K_*(V) + Q''_{YY} K'_*(V) \right] + \sum_{j=1}^{m} (Q'_Y - P^T H)_j (D_*{}^T{}_j)'_\alpha(\alpha') + \sum_{j=1}^{m} (Q'_Y - P^T H)_j (D_*{}^T{}_j)'_V \quad (6.36)$$

$$= \Lambda_1(V) + \Lambda_2(V,P) + \Lambda_3(V,P) \tag{6.37}$$

where the arguments of Q have been omitted for clarity. Now to evaluate (6.37) second derivative information is needed. Thus there is a need for more information than what is already available from the ISOPE structure. This is not reasonable , and hence $\hat{X}'$ is approximated to $\tilde{X}'$ by eliminating the term which requires the second derivatives.

Let us denote

$$\bar{\lambda}'_p = \mathfrak{C}_n(\lambda'_p) \tag{6.38}$$

$$\bar{\lambda}'_v = \mathfrak{C}_m(\lambda'_v) \tag{6.39}$$

where $\mathfrak{C}_n$, and $\mathfrak{C}_m$ are appropriately defined matrix operators. Also noting that the model is contructed such that

$$F(X,\alpha) = f_1(X) + f_2(\alpha)$$

and using the above, we get the following approximation for $\hat{X}'$

$$\tilde{X}'(V,P) = E_\rho\Big\{\rho Z'_* + \mathfrak{C}_n\Big(D_*^T(Q''_{YC} + Q''_{YU}HK'_*(V) + Q''_{YY}K'_*(V))\Big) + \Big(Q''_{XY} + F'_X(C,U,\alpha)^T Q''_{YY}\Big)D_* : t'_X(C,U,\alpha) - \mathfrak{C}_n(D_*^T H^T)\Big\} \tag{6.40}$$

Then

$$\tilde{\Psi}'(V,P) = \begin{bmatrix} I_n & 0_{nxm} \\ HD_* & 0_{mxm} \end{bmatrix} + \begin{bmatrix} \tilde{C}' \\ e'_x\tilde{X}' \end{bmatrix} \tag{6.41}$$

where $\tilde{\Psi}'(V,P) \simeq \Psi'(V,P)$

It is clearly seen that the amount of information needed from the system is identical to that needed by the relaxed algorithm. Here, however, this information is used more efficiently and this leads to an increase in the amount of off-line calculations. However, as will be seen in chapters 7 and 8, there are three definite advantages, which are

a) The updating scheme is near-precise and the convergence is much quicker

b) It is known that when adding the quadratic penalty function, the properties of the algorithm are vastly improved **(Tatjewski et al(1987))**. However there is no exact scheme whereby the penalty factors, ρ, can be selected. It is seen that as this factor increases in magnitude the performance of the algorithm is improved. But increasing the value of

ρ beyond a certain value the performance deteriorates (ill-conditioning of the problem). Using the the Newton-like scheme it is seen that the algorithm is less sensitive to the penalty factor.

c) The sensitivity to large non-linearities in the system is reduced. This is important in an on-line situation where we do not have an exact knowledge of the non-linearities.

6.2.2 Algorithm with Passive Coordination

In the preceeding sections, control structures with dynamic coordination were described. In deriving those structures we used the interaction input information which is fed back, in evaluating the parameters, α. If now we were to use this information more extensively, as was done by **Brdys and Roberts (1986a)** and **Kambhampati and Ellis (1988)** we get a structure where the coordination problem is very much simplified. This simplification comes about because we have eliminated the need to evaluate the price vectors, and also because at the lower level of the local decision unit the number of independent variables in reduced. The simplification in the coordination problem comes about because when using the feedback information more extensively, we see that we can eliminate one of the constraints, namely

$$U = HY \tag{6.42}$$

It is because of this constraint that the nature of the coordination is dynamic. Then, it is natural that if this constraint is eliminated then the resulting coordinating strategy would be passive. Let us then formulate the problem accordingly. Adding the additional variables $V \in \Re^n$, V=C, and appending the performance index with the quadratic penalty function, we get:

$$\min_{C,V,\alpha} \left\{ q(C,HK_*(V),\alpha) + \rho\|V - C\|^2 \right\}$$

s.t

$$\begin{aligned} &F(V,HK_*(V),\alpha) = K_*(V) \\ &g(C,HK_*(V),\alpha) \leq 0 \\ &V = C \end{aligned} \tag{6.43}$$

Then an analysis of the first order necessary optimality **(Brdys and Roberts (1986a))** conditions for the above would yield, as before, a problem which consists of three sub-problems which are

Parameter Estimation Problem (PCPEP)

$$F(V,HK_*(V),\alpha) = K_*(V) \dashrightarrow \alpha \tag{6.44}$$

Modified Optimisation Problem (PCMOP)

$$\min_{C,U} \left\{ q(C,HK_*(V),\alpha) - \lambda^T C + \rho\|V - C\|^2 \right\}$$

s.t (6.45)

$$g(C,HK_*(V)) \leq 0$$

where λ is the modifier

Coordination Problem (PCCP)

$$V = C \tag{6.46}$$

Evaluate the modifier λ, given by:

$$\lambda = [\ F'_V(V,HK_*(V),\alpha) - K'_*(V)\ ]^T\ [\ Q'_V(V,HK_*(V),F(V,HK_*(V),\alpha)\ ]$$

$$- [q'_V(C,HK_*(V),\alpha) + g'_V(C,HK_*(V))^T \mu\ ] \tag{6.47}$$

where μ are the Kuhn-Tucker multipliers corresponding to the inequality constraints.

The algorithmic structure is exactly similar to that given in section 6.2.1.1 The only difference is that we do not have to update the price vector.

Notice the obvious differences between the two problems ((6.5) to (6.9) and (6.44) to (6.47)). We see that in the PCMOP a term is missing. This is the term which in the DCMOP takes into account the various interactions between the subsystems. Here then it can be seen that we use the interaction measurements directly in the optimisation problem, and assume that they are constant for a given iteration. It can easily be seen that the PCMOP is also decomposable into N different sub-problems. The resultant control structure in shown in figure 6.3.

6.3 CONCLUDING REMARKS

In this chapter some decentralised algorithms were discussed. The algorithms discussed covered the two classifications of the decentralised ISOPE algortihms: the dynamic coordination and the passive coordination techniques. Both these types have their advantages and disadvantages. Using the passive coordination technique we see that the method is simple and does not need the sophisticated approach of the dynamic coordination technique. The point to remember here is that there is an increased reliance on the system measurements which, when noise is present, would definitely pose difficulties.

The dynamic coordination technique, on the other hand, does not face these problems. However, it is more complicated than the passive algorithm. However, its advantage is that it could be more robust in the presence of noise.

In the next chapter we shall study the properties of these algorithms. The optimality of these algorithms is beyond doubt, and the conditions are the same as those for the centralised algorithms. We shall also look into the convergence properties of these algorithms. In chapter 8 simulation results are presented. Finally, in chapter 9, a generalisation of these algorithms is made.

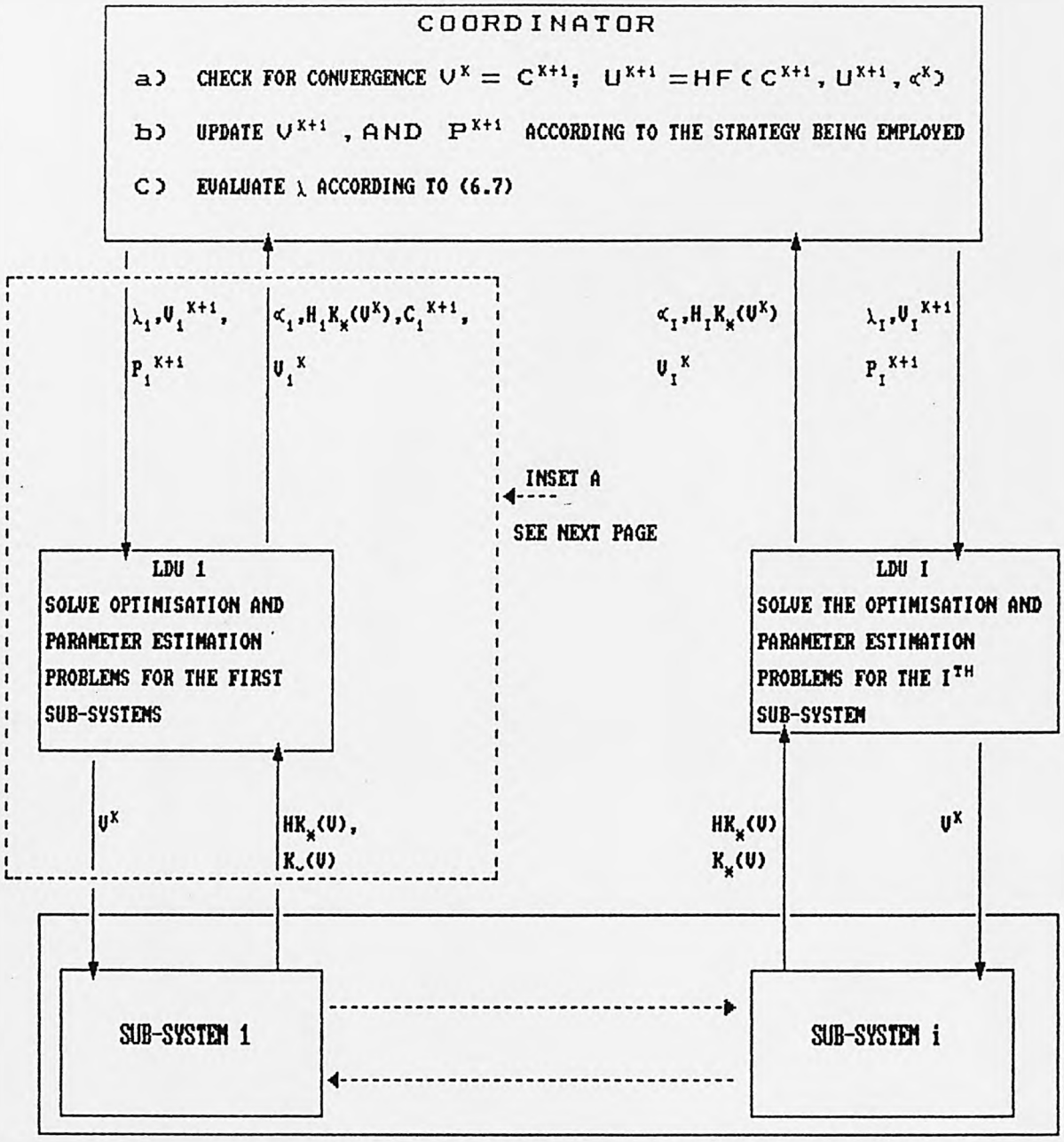


Figure 6.1 Control Structure for the Algorithms with Dynamic Coordination

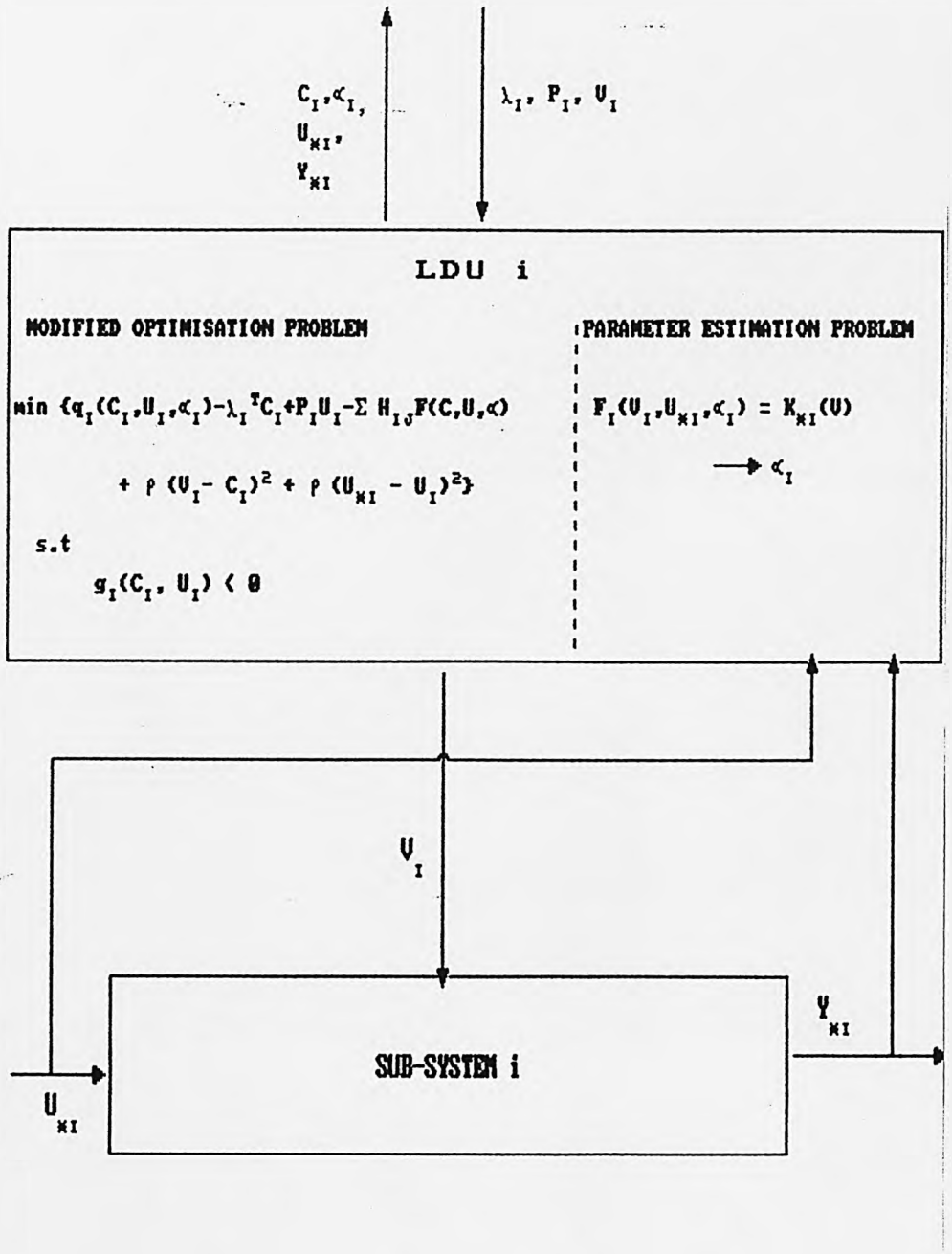


Figure 6.2 Inset A from Figure 6.1

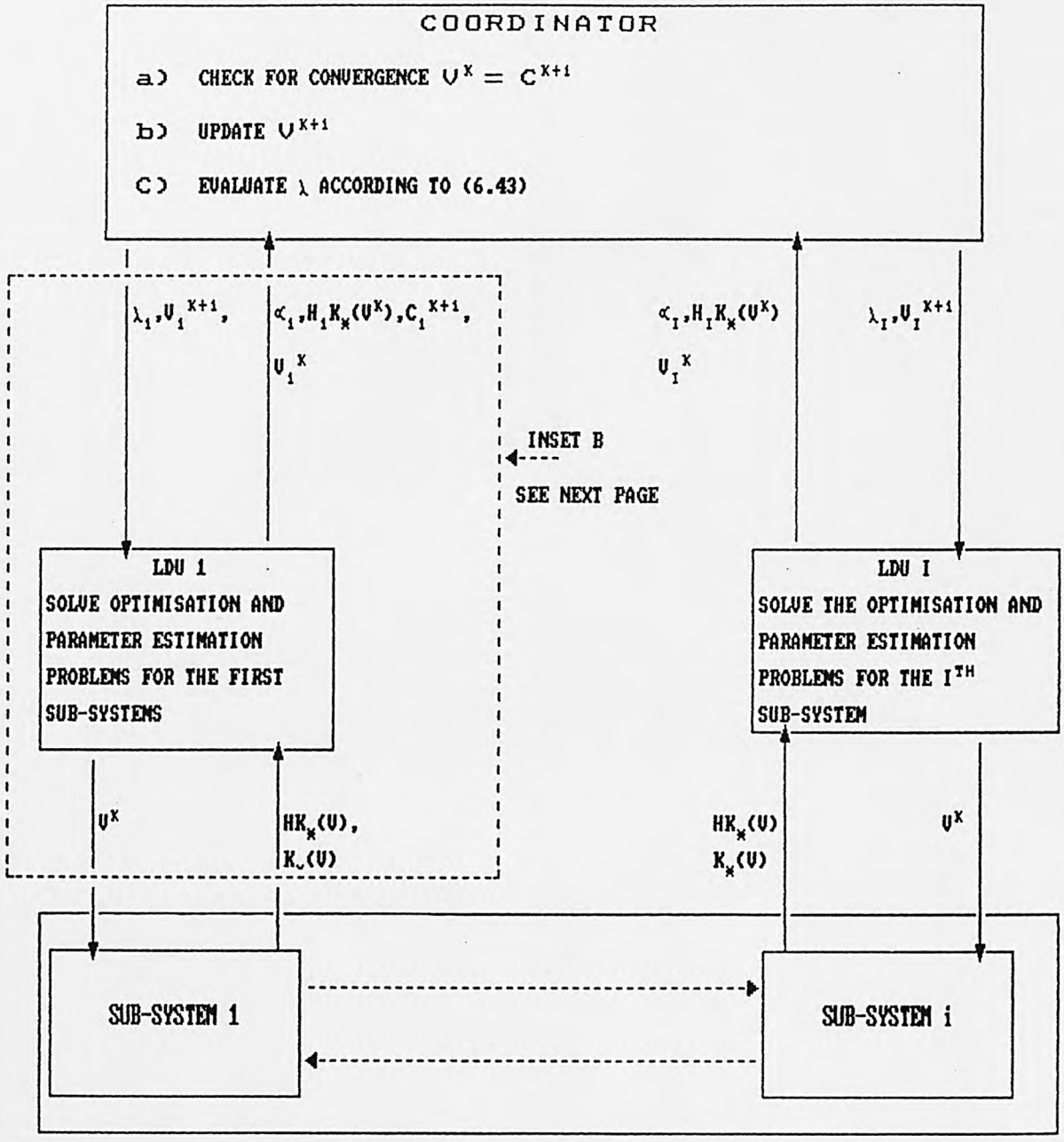


Figure 6.3 Control Structure for the ALgorithms With Passive Coordination

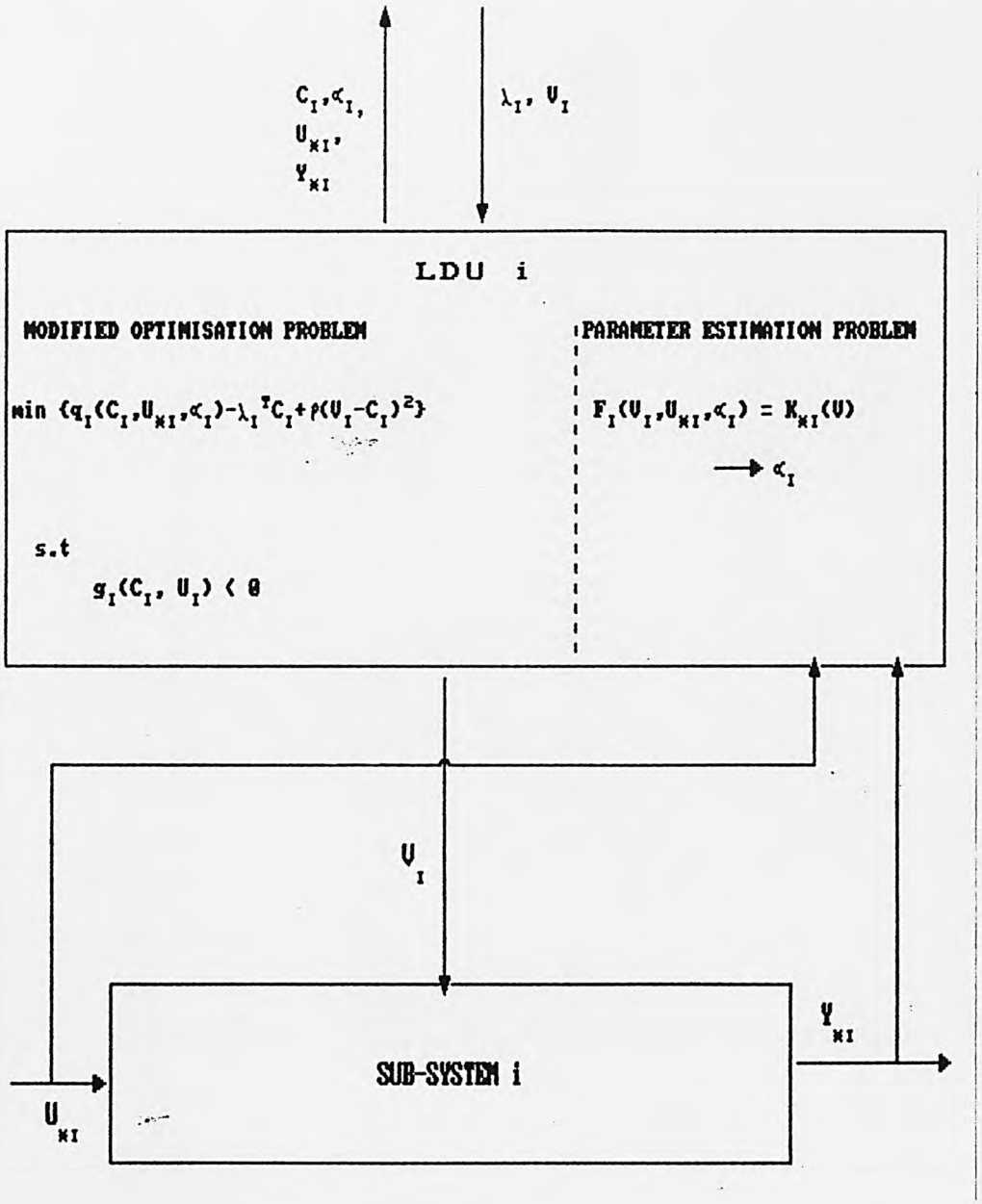


Figure 6.4 Inset B from Figure 6.2

CHAPTER 7 OPTIMALITY AND CONVERGENCE OF THE DECENTRALISED ALGORITHMS

7.1 INTRODUCTION

In the previous chapter algorithms for solving the decentralised control problem were discussed. In this chapter these algortihms will be analysed and their properties discussed. It is an inherent property of the algorithms, classified under ISOPE, that they will provide optimal solutions, because of the manner in which they are derived. The algorithms are obtained after an analysis of the first-order optimality conditions of the relevant problem. This problem is equivalent to the OCP and hence the result obtained from these is bound to be optimal. The other property, which is discussed, is the convergence of these algorithms. In fact, the question that shall be put is what are the requirements that we have to put on the problem so that the algorithms converge to the optimum in the most efficient manner.

Here, we shall not be discussing the relaxed form of the dynamic algorithm, as it has been seen to be difficult to prove the convergence of this algorithm. However, it has been seen that this algorithm converges to the optimum very efficiently **(Chen (1986), Abdullah (1988))**. However, the convergence of the Newton-like form of the dynamic algorithm will be discussed. It will be seen that we can give a measure of the the rate of convergence for this algorithm. The reason for this is that the updating scheme employed, enables us to use the contraction mapping theorems of **Ortega and Rheinboldt (1970)**

The passive coordination technique is also analysed. The analysis includes the optimality and the convergence conditions of the algorithm. As has been mentioned earlier, the optimality properties of all the ISOPE algorithms are the same. This is because of the manner in which the algorithms are derived. Again, as was analysed in chapter 5, the modifier, λ, can be reduced to the difference of the model based performance index and the system based performance index. This reduction is then used in proving the optimality and assessing the convergence properties of the algorithm.

7.2 OPTIMALITY

Here we shall assess the optimality[1] of the algorithms. In fact the optimality of these algorithms is beyond doubt, as they are all derived using the first-order necessary optimality conditions of their respective problem formulations, which

[1] *By optimality of algorithm, we are referring to a local optimum. For practical purposes this will be a local minimum.*

are all equivalent to the OCP. However, the study of the optimality of these algorithms helps us in analysing the properties of the algorithm further. Here, we shall be studying the optimality conditions for the passive coordination technique, as the properties of dynamic coordination algorithm have already been studied **(Abdullah(1988))**. Before proceding further let us make some definetions.

Let Ω be the solution set which contains all the points (V), which are the solution to the OCP; such that

$$\Omega \triangleq \left\{ (V) : (V, HK_*(V)) \in \mathcal{CU},\ V = \hat{C}(V,\alpha) \right\}$$

where

$$\mathcal{CU} \triangleq \left\{ (C,U) : G(C,U) \leq 0 \right\}$$

As was seen in chapter 4 the modifier λ, if we assume that it is autonomous and dependent on V, can be reduced in terms of the model based and system based performance indices. Hence λ can be written as

$$\lambda = q'_c(V, HK_*(V), \alpha) - q'_*(V) \tag{7.1}$$

We are now in a position to verify the optimality of the algorithm. The conditions are similar to those given in chapter 4.

Lemma 7.2.1

Let $\mathcal{C} \triangleq \left\{ (C) : g(C, HK_*(V)) \leq 0 \right\}$

Assume that each point of the set $\mathcal{C}$ satisfies the regularity conditions. Then the point $V \in \Omega$ is the solution of the OCP, and satisfies the Kuhn-Tucker necessary optimality conditions for the OCP.

Proof

Now let $\hat{C}(V,\alpha) \in \mathcal{C}$; then we have

$$\left.\begin{aligned} &q'_c(\hat{C}(V,\alpha), HK_*(V), \alpha) - \lambda + \mu^T g'_c(\hat{C}(V,\alpha), HK_*(V)) = 0 \\ &\mu > 0\ g(\hat{C}(V,\alpha), HK_*(V)) = 0;\ \mu^T g(\hat{C}(V,\alpha), HK_*(V)) = 0 \end{aligned}\right\} \tag{7.2}$$

Then using (7.1) and the fact that $V = \hat{C}(V,\alpha)$ in the above we have

$$\left.\begin{aligned} &q'_*(V) + g'_*(V)^T \mu = 0 \\ &\mu^T g_*(V) = 0;\ \mu > 0\ g_*(V) = 0 \end{aligned}\right\} \tag{7.3}$$

which are the Kuhn-Tucker conditions for V to be the solution of the OCP.

Q. E. D

Notice the similarity , between the conditions lemma 4.2.1 poses and those that lemma 7.2.1 poses. In fact these conditions are valid for all the algorithms which are derived using the ISOPE techniques. The optimality is ensured by the

modifier, λ, which in all cases can be reduced in terms of the model based and system based performance indices. Then, using the first order conditions for optimality of the modified optimisation problem, along with the reduced definition of λ, we get the optimality conditions for the OCP.

7.3 THE CONVERGENCE CONDITIONS

It was mentioned in section 7.2 that the optimality conditions for the ISOPE algorithms are similar. This property stems from the fact that the method of deriving these algorithms is similar for all the algorithms. In fact this is also true of the convergence conditions for these algorithms. This shall be seen in chapter 9, where a generalised set of conditions are provided for the algorithms which use the under-relaxation scheme.

Here, in this section we shall discuss the convergence conditions of the dynamic coordination Newton-like algorithm and the passive coordination algorithm. The methods employed to do this, vary for the two algorithms. For the dynamic Newton-like algorithm we use some of the techniques used by **Ortega and Rheinboldt (1970)**. This is facilitated by the fact that the updating scheme is structured in such a manner that the use of these methods is convenient. On the other hand, for the passive coordination algorithm we employ the techniques already used in chapter 4. We shall then, first discuss the properties of the Newton-like algorithm and then go on to the passive coordination algorithm.

7.3.1 The Newton-like Dynamic Coordination Algorithm

Using the same notation as in chapter 6, the following theorem gives the conditions under which the Newton-like algorithm converges. It should be noted here that due to the manner in which the algorithm has been derived, and the updating procedure employed, we can use the contraction mapping theorems and establish some sort of a measure for the convergence rate of the algorithm.

Theorem 7.3.1.1

Assume that the assumptions of theorem 6.2.2.1are satisfied, and that the matrix $\Psi'(\hat{V},\hat{P})$ is non-singular.

Then, for every ρ such that theorem 6.2.2.1 holds, there exists a **neighbourhood $N_\rho(\hat{V},\hat{P})$ of the optimal control-multiplier pair $(\hat{V},\hat{P})$ such that if $(V^o,P^o) \in N_\rho(\hat{V},\hat{P})$ then provided the model used is sufficiently accurate the sequence**

(V^k,P^k) generated by the algorithm remains in $N(\hat{V},\hat{P})$ and converges linearly to $(\hat{V},\hat{P})$, with the R-convergence factor δ satisfying

$$\delta \leq \left\| I - \tilde{\Psi}'(\hat{V},\hat{P})^{-1}\Psi'(\hat{V},\hat{P}) \right\|_S \tag{7.4}$$

where $\| \cdot \|_S$ denotes the spectral norm.

Proof

An iteration of the Newton-like algorithm can be described as follows

$$\tilde{\Psi}'(V^k,P^k)\begin{bmatrix} \tilde{V}^{k+1} - V^k \\ P^{k+1} - P^k \end{bmatrix} = -\begin{bmatrix} V^k - \hat{C}(V^k,P^k) \\ \hat{U}(V^k,P^k) - HF(\hat{X}(V^k,P^k),\alpha) \end{bmatrix}$$

$$= H(V^k,P^k) \tag{7.5}$$

Using similar arguments as **Tatjewski and Roberts (1987a)**, it is sufficient to show that $\| H'(\hat{V},\hat{P})\|_S$ is less than one. Then, using an argument similar to that of **Ortega and Rheinboldt (1970, section 10.2.1)** it can be shown that $H(.,.)$ is differentiable at $(\hat{V},\hat{P})$ and

we have

$$H'(\hat{V},\hat{P}) = I - \tilde{\Psi}'(\hat{V},\hat{P})^{-1}\Psi'(\hat{V},\hat{P}) \tag{7.6}$$

Then we have to show that

$$\| H'(\hat{V},\hat{P})\|_S \leq \left\| I - \tilde{\Psi}'(\hat{V},\hat{P})^{-1}\Psi'(\hat{V},\hat{P}) \right\|_S \tag{7.7}$$

Let us denote by ν_i the eigenvalues of $\tilde{\Psi}'(\hat{V},\hat{P})^{-1}\Psi'(\hat{V},\hat{P})$, $i = 1, \cdots, n$, then

$$\left\| I - \tilde{\Psi}'(\hat{V},\hat{P})^{-1}\Psi'(\hat{V},\hat{P}) \right\|_S = \max_{1\leq i\leq n} | 1 - \nu_i | \tag{7.11}$$

Then it is sufficient to show that $|1-\nu_i| < 1$. Then using similar arguments as **Tatjewski and Roberts (1987)**, we can easily show that if the model is accurate, the closer is the matrix $\tilde{\Psi}'(\hat{V},\hat{P})^{-1}\Psi'(\hat{V},\hat{P})$ to I and that the algorithm is convergent.

Q. E. D

The above theorem gives some useful properties of the algorithm. It states that the smaller the model-reality differences the better is the local convergence of the algorithm – regardless of the value of ρ, which is easily confirmed through simulations. On the other hand, with a precise non-linear model, which in practice is not a realistic condition, or for a different but linear model and system mappings, we have $\tilde{\Psi} = \Psi$ (the arguments have been dropped) and then $\delta = 0$, and the algorithm converges superlinearly. Moreover, the problem does not become ill-conditioned as easily as the relaxed algorithm for large values of ρ.

7.3.2 Passive Coordination Algorithm

From the similarities in the structures of the passive coordination algorithm and the MSTP algorithm, it can be seen that these two algorithms will have similar convergence properties. In fact all the algorithms which are classified as ISOPE will have similar convergence conditions. This, as was mentioned earlier, is because of the way in which these algorithms and their associated control structures are derived. Starting from the OCP, depending on how we use the system feedback information (Bryds and Roberts(1986a)), we obtain a whole set of model-based problems. All these problems are equivalent to the OCP, and employ a modifier, λ, which attempts to quantify the differences in the model based-performance index and the system based performance index. In fact in chapter 9, we provide a generalised set of convergence conditions for these problems.

The following theorem then gives the conditions under which the algorithm, which employs a passive coordination technique, would converge. This is similar to the convergence theorem given for the MSTP algorithm.

Theorem 7.3.2.1

Assume that

1. All constraint sets are convex and compact and that all the mappings involved are continuous and continuously Frechet differentiable.
2. The functional $q'_C(\ ,HK_*(V),\alpha)$ is uniformly monotone with some constant $a(\alpha) > 0$ for every $\alpha \in \mathcal{A}$ and also that $q'_*(\ .\ )$ is also uniformly monotone with some constant $a^* > 0$.
   where $\mathcal{A} = \bigcup_{V\in\Omega} \alpha(V)$
3. There exist such numbers $\bar{A}$ and $\underline{a}$ such that
   $$\bar{A} > A(\alpha) \qquad a(\alpha) \geq \underline{a} > 0 \tag{7.12}$$
   where $A(\alpha)$ is the Lipschitz constant of $q(\ ,HK_*(V),\alpha)$
4. The model is selected such that
   $$\bar{A} \leq 2a^* - \frac{1}{2}A^* \tag{7.13}$$
   $$\frac{3}{4}A^* < \underline{a} \leq A^* \tag{7.14}$$
   where A^* is the Lipschitz constant of q^*
5. The class of systems considered is restricted in a manner
   $$a^* \geq \frac{5}{8}A^* \tag{7.15}$$

Then

I. There exists a solution $\bar{V}$ of the OCP

II. There exists such scalars ε_1 , ε_2 , such that $0< \varepsilon_1 <0.5$, $0.5< \varepsilon_2 \leq 1$, such that for every $\varepsilon \in (\varepsilon_1,\varepsilon_2)$ the algorithm generates a sequence $\{V^k\}$

which converges to $\bar{V}$.

Proof

The proof of this theorem is similar to that of theorem 4.2.1 and is hence omitted.

The conditions, as has been already mentioned, for convergence of the algorithm are similar to those given for the MSTP. We have already stated earlier that this is a feature of the algorithms which come under the class of ISOPE techniques. In fact, these conditions are more strict than required. One can develop conditions, which are similar those developed for the MSTP by **Brdys and Roberts (1987)**. The reason for the development of these conditions is that this facilitates the generalisation of the ISOPE algorithms, as will be seen in chapter 9.

7.4 CONCLUDING REMARKS

In this chapter we have seen that the decentralised algorithms, described in chapter 6, produce optimal solutions for the OCP, and also that under some given conditions they will converge to that optimum solution.

The analysis given for the Newton-like algorithm is local assuming that the algorithm works in a neighbourhood of the optimal point $\hat{X} = (\hat{C}, \hat{U})$ such that at each iteration precisely those constraints are active at $\hat{X}(V^k, P^k)$ which are also active at $\hat{X}$. Such a neighbourhood always exists when the problem satisfies the strict complementarity conditions at the optimal point. Here it should be noted that at times when the model and system differ a lot or for highly non-linear systems and also when the constraint activities keep changing, the updating mechnism has to be modified in that we use scalar $\epsilon < 1$, which effectively reduces the step length. The updating scheme then becomes

$$\begin{bmatrix} \tilde{V}^{k+1} - V^k \\ P^{k+1} - P^k \end{bmatrix} = -\epsilon\, \tilde{\Psi}'(V^k,P^k)^{-1} \begin{bmatrix} V^k - \hat{C}(V^k,P^k) \\ \hat{U}(V^k,P^k) - HF(\hat{X}(V^k,P^k),\alpha) \end{bmatrix}$$

Generally the activity of the constraints will keep changing, when they are composed of the control set-points and the interactions. One cannot guarantee that these constraints will not be violated, as the system mappings are not known precisely, and it is not possible to predict the value of the interactions $U_i = HK_i(V)$ for a given value of V. Thus, it is not possible to predict the changes in activity of these constraints so that some a priori action may be taken.

On the other hand, the analysis for the passive coordination technique is more general, and no strict conditions are posed on it. The constraint activity is of no prime importance here. And it can handle the violations of the conditions better. However, for the constraints which are dependent on the interaction inputs and outputs, techniques have been developed which ensure that these are not violated (Lin et al (1987a, 1987b, 1987c)).

In the next chapter we present a simulation study of the algorithms described.

CHAPTER 8 SIMULATION RESULTS FOR THE DECENTRALISED ALGORITHMS

8.1 INTRODUCTION

The objective of any simulation exercise is to verify the results of the analysis of an algorithm. The simulation can be carried out using an abstract example, where we could incorporate those properties we wish to verify in the example or else change the characteristics of the example whereby we could gain further insight into the algorithm. Often it is found that in most cases the conditions for convergence and optimality of an algorithm, obtained through an analysis of the algorithm, are stricter than is necessary. This particular aspect has been dealt with already in chapter 4 where a set of convergence conditions were developed for the MSTP algorithm which were stricter than those put forward by **Bryds and Roberts (1987)**. This was done mainly to get an insight into the significance of the modifier, and to establish some framework for defining the model-reality differences. However, sometimes due to the nature of the algorithm, it may not be possible to develop weak convergence conditions and the conditions obtained will be very strict than what is necessary, **(Brdys et al(1986c))**. Thus, any simulation exercise has the following functions :

a) To verify the theoretical results obtained

b) To investigate the algorithm further so that those properties of the algorithm which could not be assessed theoretically could be brought out and investigated further.

In fact there is one more point to be kept in mind. That is, a proposed algorithm or technique has to have practical viability. To test this particular aspect we have to mimic a realistic situation, whereby there would be disturbances in the form of noise, present in the system. All the algorithms must also perform satisfactorily in such a situation. However, this particular aspect of a simulation study is beyond the scope of the present thesis.

Here in this chapter, the algorithms described in chapter 6 are tested using abstract examples. All the algorithms were found to converge to the optimum within the desired accuracy with, of course, varying rates of convergence.

8.2 ALGORITHMS WITH DYNAMIC COORDINATION

Both, the relaxed as well as the Newton-like, algorithms converged to the desired optimum. However, the rates at which they converged differed.

A sensitivity analysis is carried out for the algorithms with dynamic coordination. The algorithms were tested for sensitivity to the non-linearities in the system and also to the penalty factor. The results of such an analysis helps in verifying some of the conclusions reached in chapter 6 and 7. It was found that that an increase in the non-linearities in the system reduced the performance of the relaxed algorithms more than that of the Newton-like algorithm. This was even though the Newton-like updating scheme became more and more approximate, as was mentioned in the previous chapter. Again, it was found that with an increase in the value of the penalty factor, the performance of the relaxed algorithm tended to improve but deteriorated after some limit, where as with the Newton-like algorithm it made no appreciable difference. On the whole the Newton-like algorithm seemed to have better convergence properties than the relaxed algorithm.

8.2.1 Example

The example used for the simulation study consists of 3 interconnected sub-systems and has a total of 7 control set-points and 4 interconnection inputs and outputs. The configuration of this example is shown in figure 8.1.

For purposes of simulation, the real system is represented by the following equations :

Sub-system I:

$$Y_{11*} = 1.3C_{11} - C_{12} + 2.0U_{11*} + \beta_1 C_{11} U_{11}$$

The performance index is

$$Q_1 = \left(U_{11*} - 1 \right)^4 + 5\left(C_{11} + C_{11} - 2 \right)^2$$

The constraint set $\mathcal{CU}_1 = \left\{ 1 - (C_{11} + C_{12}) \geq 0 \; ; \; 0 \leq U_{11*} \leq 0.5 \right\}$

Sub-system II

$$Y_{21*} = C_{21} - C_{22} + \beta_2 C_{22}^{\,2} + 1.2U_{21*} - 3.0U_{22*}$$

$$Y_{22*} = 2C_{22} - 1.25C_{23} + \beta_3 C_{23} C_{22} + U_{22*} - U_{21} + 0.1$$

The performance index is

$$Q_2 = 2\left(C_{21} - 2 \right)^2 + C_{22}^{\,2} + 3C_{23}^{\,2} + 4U_{21*}^{\,2} + U_{22*}^{\,2}$$

$$\mathcal{CU}_2 = \left\{ 4 - (4C_{21} + C_{22} + 2C_{23}) \geq 0 \wedge 4.5 - (4C_{11} + 2U_{21*} + C_{23}) \geq 0 \right\}$$

Sub-system III

$$Y_{31*} = 0.8C_{31} + 2.5C_{32} - 4.2U_{31*}$$

The performance index is

$$Q_3 = \left(C_{31} + 1 \right)^2 + 2.5{C_{32}}^2 + \left(U_{31*} - 1 \right)^2$$

$$cu_3 = \left\{ C_{31} + U_{31*} + 1.0 \geq 0 \wedge 0 \leq C_{32} \leq 1 \right\}$$

The interconnections are given by

$$U_* = \begin{bmatrix} U_{11*} \\ U_{21*} \\ U_{22*} \\ U_{31*} \end{bmatrix} = H \begin{bmatrix} Y_{11*} \\ Y_{21*} \\ Y_{22*} \\ Y_{31*} \end{bmatrix} = HY_*; \text{ where } H = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

and $C = \{ C_{1i}, C_{2j}, C_{3k} \}$, i = 1,2 j = 1,3 k = 1,2 are the control set-points and U_*, Y_* are the interconnection inputs and outputs.

In the above set of expressions representing the system the parameters β_i, i=1,2,3, are the non-linearity parameters, the nominal values of which are { 0.1, 0.1, 0.1}. By varying these non-linearity parameters, we are effectively changing the mapping of the system. Then, by keeping the model mapping unaltered, we can judge the effect of the model-reality differences on the two algorithms. By this we achieve the objective of comparing the rates of deterioration in the performance of the algorithms as the model-reality differences increase. This can be effectively done by changing the non-linearity parameters.

It has also been mentioned earlier that, models do not represent the reality due to the simplifing assumptions one makes during the modelling process. Thus, we may find that the model developed, is a linear representation of the system, where as in actual fact the system is non-linear. This particular aspect can be seen from the example selected. The model used to represent the system, consists of the linear parts of the system mappings, and does not include the non-linearties present. Then by varying these non-linearities, which in reality one cannot predict accurately, we can see what effect this model simplification has on the performance of the algorithms.

Then the model used to represent this system is as follows

Sub-system I

$$Y_{11} = C_{11} - C_{12} + U_{11} + \alpha_1$$

Sub-system II

$$Y_{21} = C_{21} - C_{22} + U_{21} - 3U_{22} + \alpha_2$$

$$Y_{22} = 2C_{22} - C_{23} - U_{21} + U_{22} + \alpha_3$$

Sub–system III

$$Y_{31} = C_{31} + 2.5C_{32} - 4U_{31} + \alpha_4$$

The simulations for testing the sensitivity of the algorithms, were all started from the same set of initial conditions, which is the optimum obtained when using a fixed model, with the parameters $\{\alpha_i,\ i=1,2,3,4\}$ set to zero, which is

$$C^0 = \{\ 0.530,\ 0.847,\ 0.99,\ -0.147,\ 0.0031,\ -0.505,\ 0.343\ \}$$

$$U^0 = \{\ 0.17288,\ 0.02864,\ 0.331,\ 0.0050\ \}$$

$$P^0 = \{\ -0.5980,\ -1.4307,\ 0.6852,\ -0.44631\ \}$$

8.2.2 Results and Discussion

The starting point for solving any optimisation problem should be as near as possible to the optimum. However, for solving the system optimisation problem the real optimum is not known as we are dealing with an unknown system mapping. Then it is reasonable that the starting point is the model based optimum, as this would be more realistic. Hence, the simulations were started from the model based optimum, which has been enumerated above. Also in the simulation study, the system mapping was varied, by changing the non-linearities in the system.

Table 8.1 shows the comparative efficiency of the two algorithms in achieving the optimum. One can immediately see that the relaxed algorithm is more inefficient than the Newton-like scheme. This can be seen by the number of iterations that are required to reach the optimum, 33 for the relaxed algorithm as compared to 4 for the Newton-like algorithm. This was with nominal values of 2 for the penalty factor, and β = { 0.1, 0.1, 0.1 } for the nominal non-linearities.

It has been stated that for linear system and model mappings the Newton-like algorithm should, provided the activity of the constraints does not change, converge superlinearly. This can be confirmed from table 8.2, where results of the simulations, with linear system and model mappings and setting the penalty factor ρ to zero, using different starting points are shown. Here three starting points were selected:

a) The starting point A is close to the optimum and there are no changes in the activity of the constraints, as can be seen in table 8.2

b) The starting point B is the model based optimum, already mentioned above

c) The starting point C is far from the optimum, and as can be seen from table 8.2 there are changes in the activity of the constraints.

Using these three starting points, one can see the effect of the changes of the activity of the constraints on the algorithms. It can be seen that if the starting point is A, there no changes in the activity of the constraints and hence the algorithm converges as prescribed. However, using the model based optimum it can be seen that there are changes in the activity of the constraints, and this is reflected in the increased number of iterations. On the other hand, using the starting point C it can be seen that there are a number of changes in the activity of the constraints and it takes 4 iterations to reach the optimum. Notice that even when using a starting point which is far from the optimum, the algorithm converges to near optimum, very quickly.

Tables 8.3 and 8.4, show the effect of the value of the penalty factor ρ, on the performance of the algorithm. It has been stated that within reasonable limits, whatever be the value of ρ, the formulation of the problem for the relaxed algorithm gets ill-conditioned faster, where as the Newton-like algorithm does not. This can be seen from the results tabulated in tables 8.3 and 8.4. Table 8.3 tabulates the results obtained from simulating the example problem, using the nominal value for the non-linearities, that is β = { 0.1, 0.1, 0.1}, and for a range of values of the penalty factor ρ. Table 8.4 shows similar results, for the Newton-like algorithm, but where the system and model mappings are linear, that is the parameters β = {0.0, 0.0, 0.0}. These results are shown graphically in figure 8.2. Here it can be seen that, for the example considered, the performance of the relaxed algorithm gets better as the value of the penalty factor increases but beyond a value of $\rho = 2$ this performance deteriorates. On the other hand the performance of the Newton-like algorithm shows no appreciable sign of deterioration, and is fairly constant.

Table 8.5 shows the effect of the non-linearities in the system on the algorithm. It must be remembered here that the Newton-like updating scheme, becomes more imprecise as the non-linearities increase. This is because of the approximation made by ignoring the higher derivative terms in the control updating scheme. Thus by varying the non-linearity parameters, one can study their effect on the performance of the algorithms. Then using the nominal value of $\rho = 2$, and the model based optimum as the starting point for all the simulations, we get the results tabulated in table 8.5 and graphically represented in figure 8.3. It can be seen that as the non-linearities in the system are increased the performance of the algorithms deteriorates. But the deterioration is much quicker for the relaxed algorithm than for the Newton-like algorithm.

8.3 ALGORITHMS WITH PASSIVE COORDINATION

Some of the advantages of this algorithm have already been discussed in detail, in the earlier chapters. It has been proved that this algorithm converges and the point of convergence is the optimum, under some conditions. Here, in this section this algorithm is tested using two example systems.

8.3.1 Examples

To test the algorithm, two example systems were used. The configuration of these systems is the same as that used for simulating the dynamic coordination algorithms. That is, both the examples had three interconnected sub-systems, a total of seven set-points and four interconnection inputs and outputs.

8.3.1.1 Example A

The first example selected has, as was mentioned, three sub-systems and the same configuration as the example used for testing the dynamic coordination algorithm. Then, as before, the system mappings are represented by the following equations :

Sub-system I:

$$Y_{11*} = 1.5C_{11} - C_{12} + 0.02U_{11*}$$

The performance index is

$$Q_1 = 0.1 \left(Y_{11*} - 1 \right)^2 + 5C_{11}^{\,2} + 5C_{12}^{\,2}$$

The constraint set $\mathcal{CU}_1 = \left\{ 1 - (C_{11}^{\,2} + C_{12}^{\,2} + 0.1U_{11*}) \geq 0 \; ; \; U_{11*} \leq 2.0 \right\}$

Sub-system II

$$Y_{21*} = C_{21} - C_{22} + 0.1_2C_{22}^{\,2} - 0.5U_{21*} + 0.05\,0U_{22*}$$

$$Y_{22*} = 2C_{22} - 1.3C_{23} + 0.25C_{23}C_{22} + 0.09U_{22*} - U_{21} + 0.1$$

The performance index is

$$Q_2 = 2\left(C_{21} - 2 \right)^2 + C_{22}^{\,2} + 3C_{23}^{\,2} + 0.1\left(Y_{21*} - 2.0\right)^2$$

$$\mathcal{CU}_2 = \left\{ 1.2 - (0.5C_{21} + C_{22} + 2C_{23} + 0.01(U_{21} + U_{22*})) \geq 0 \; \wedge \right.$$
$$\left. -10.0 \leq C_{2i} \leq 10.0, \; i=1,2,3 \right\}$$

Sub-system III

$$Y_{31*} = 1.3C_{31} + 2.5C_{32} - 0.8U_{31*}$$

The performance index is

$$Q_3 = \left(C_{31} + 1 \right)^2 + 2.5C_{32}{}^2 + 0.01\left(Y_{31*} - 3.0 \right)^2$$

$$\mathcal{CU}_3 = \left\{ C_{31} + C_{32} + 0.1U_{31*} + 1.0 \geq 0 \wedge 0 \leq C_{32} \leq 1 \right\}$$

The interconnections are given by

$$U_* = \begin{bmatrix} U_{11*} \\ U_{21*} \\ U_{22*} \\ U_{31*} \end{bmatrix} = H \begin{bmatrix} Y_{11*} \\ Y_{21*} \\ Y_{22*} \\ Y_{31*} \end{bmatrix} = HY_*; \text{ where } H = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

and $C = \{ C_{1i}, C_{2j}, C_{3k} \}$, $i = 1,2$ $j = 1,3$ $k = 1,2$ are the control set-points and U_*, Y_* are the interconnection inputs and outputs.

The model used to represent this system is as follows

Sub–system I

$$Y_{11} = C_{11} - C_{12} + U_{11} + \alpha_1$$

Sub–system II

$$Y_{21} = C_{21} - C_{22} - U_{21} + U_{22} + \alpha_2$$

$$Y_{22} = 2C_{22} - C_{23} - 1.5U_{21} + 0.5U_{22} + \alpha_3$$

Sub–system III

$$Y_{31} = C_{31} + 2.5C_{32} - 1.0U_{31} + \alpha_4$$

The starting point for this example was $C^0 = \left\{ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 \right\}$

8.3.1.2 Example B

The second example is similar to the previous example. It again consists of three sub-systems, and has seven control set-points. The system mappings are, as has been done for the other examples, represented by the following :

Sub–system I:

$$Y_{11*} = 1.5C_{11} - C_{12} + 0.5U_{11*}$$

The performance index is

$$Q_1 = 0.1 \left(Y_{11*} - 1 \right)^2 + 5C_{11}{}^2 + 5C_{12}{}^2$$

The constraint set $\mathcal{CU}_1 = \left\{ 1 - (C_{11}{}^2 + C_{12}{}^2) \geq 0 \; ; \; U_{11*} \leq 2.0 \right\}$

Sub–system II

$$Y_{21*} = C_{21} - C_{22} + 0.1_2C_{22}^{\,2} - 1.5U_{21*} + 0.5.0U_{22*}$$

$$Y_{22*} = 2C_{22} - 1.3C_{23} + 0.25C_{23}C_{22} + 0.5U_{22*} - U_{21} + 0.1$$

The performance index is

$$Q_2 = 2\left(C_{21} - 2 \right)^2 + C_{22}^{\,2} + 3C_{23}^{\,2} + 0.1\left(Y_{21*} - 2.0\right)^2$$

$$\mathcal{CU}_2 = \left\{ 1.2 - (0.5C_{21} + C_{22} + 2C_{23}) \geq 0 \wedge \; -10.0 \leq C_{2i} \leq 10.0,\ i=1,2,3 \right\}$$

Sub–system III

$$Y_{31*} = 1.3C_{31} + 2.5C_{32} - 0.1U_{31*}$$

The performance index is

$$Q_3 = \left(C_{31} + 1 \right)^2 + 2.5C_{32}^{\,2} + 0.01\left(Y_{31*} - 3.0 \right)^2$$

$$\mathcal{CU}_3 = \left\{ C_{31} + C_{32} + 1.0 \geq 0 \wedge 0 \leq C_{32} \leq 1 \right\}$$

The interconnections are given by

$$U_* = \begin{bmatrix} U_{11*} \\ U_{21*} \\ U_{22*} \\ U_{31*} \end{bmatrix} = H \begin{bmatrix} Y_{11*} \\ Y_{21*} \\ Y_{22*} \\ Y_{31*} \end{bmatrix} = HY_*;\ \text{where } H = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

and $C = \{ C_{1i}, C_{2j}, C_{3k} \}$, $i = 1,2$ $j = 1,3$ $k = 1,2$ are the control set-points and U_*, Y_* are the interconnection inputs and outputs.

The model used to represent this system, is the same as that used for example B, and is as follows

Sub–system I

$$Y_{11} = C_{11} - C_{12} + U_{11} + \alpha_1$$

Sub–system II

$$Y_{21} = C_{21} - C_{22} - U_{21} + U_{22} + \alpha_2$$

$$Y_{22} = 2C_{22} - C_{23} - 1.5U_{21} + 0.5U_{22} + \alpha_3$$

Sub–system III

$$Y_{31} = C_{31} + 2.5C_{32} - 1.0U_{31} + \alpha_4$$

The starting point for this example was $C^0 = \left\{ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 \right\}$

8.3.2 Results and Discussion

The results of the simulation for these two examples are shown in table 8.6. As has been mentioned before the iterative loop gains are arrived at after some trials. It can be seen that the examples converge to the optimum, rather efficiently. This can also be seen in the figures 8.4 and 8.6. These figures show the variation of the system based performance index with each iteration. In chapter 4, it was shown that the function T(V) converges to zero as the algorithm approaches the optimum. This was a part of the proof for the convergence of the algorithm. For the passive coordination algorithm also it is seen that such a function will also converge to zero as the algorithm approaches the optimum. Generally, the function T(V) is not known exactly, as it is a function of the monotone constant, a_k, of the system based performance index. Hence, here we use the following function, $T_1(V)$, which is:

$$T_1(V) = \sqrt{\frac{\sum_{i=1}^{n}(V^k_i - \bar{V}_i)^2}{N}}$$

and the figures 8.6 and 8.7 show the variation of this function with each iteration. It can be seen that this function converges to zero, as the algorithm approaches the optimum.

8.4 CONCLUDING REMARKS

It has been shown that the algorithms which solve the decentralised control problem produce and converge to the optimum of the OCP. Of course the rate at which they do so vary, and this is because of the modification made to the problem, which gives different properties to the model based problem. This would then, make the behaviour of the algorithms differ. However, in general it can be said that they all converge to the optimal solution. At this point, it should, however, be pointed out, that, with the example, with linear system and model mappings, used for the dynamic coordination algorithm, it was found difficult to obtain the convergence to the optimum. This could mean that the algorithm, using a passive coordinator has some special properties which have not been investigated. An algorithm, which has a similar problem formulation to the passive coordination algorithm, had been proposed by Brdys and Ulanicki (1978). Here the problem was solved in a completely decentralised manner, and as a result they had some problems regarding the continuity of some of the mappings involved. It could be possible that these conditions may also be valid here and are subject to

investigation at the time of writ ing this thesis.

In fact, as been often repeated in this thesis, all the ISOPE algorithms, when they converge, will converge to the optimum. This is because of the manner in which they are derived. They are all obtained after analysing the first order necessary conditions for optimality of the integrated problem of parameter estimation and system optimisation. It is this fact, which helps us to generalise these algrithms.

Then in the next chapter, we shall generalise these algorithms. In the process of generalising these algorithm, we see that we get a generalised set of variables, and also a generalised integrated problem. A refined set of generalised optimality conditions are not provided as one can assume that all the ISOPE algorithms are optimal. The algorithms considered, include those algorithms where the performance index is augmented. The unaugmented algorithms are considered as a special case of the augmented algorithms when the penalty factor ρ is set to zero. In fact, this can be seen is chapter 4 where the convergence for the MSTP algorithm and its associated augmented algorithm are the same. The only difference occurs in that we have to take into account the augmentations. If ρ is set to zero the conditions are then the same as those of the MSTP.

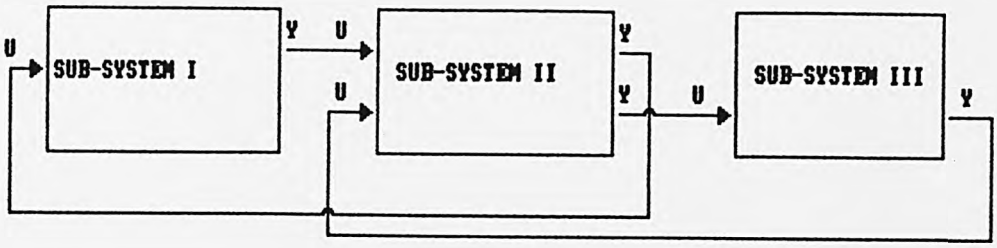


Figure 8.1 Configuration of Example

ALGORITHM TYPE	ITERATIVE LOOP GAINS		ITERATIONS	No of SET-POINT CHANGES	PENALTY FACTOR ρ $\rho = \rho$	NON-LINEARITY PARAMETERS $\beta = \langle \beta\ \beta\ \beta \rangle$
RELAXED SCHEME	0.4	0.5	33	231	2.0	0.1, 0.1, 0.1
NEWTON-LIKE SCHEME	—	—	4	28	2.0	0.1, 0.1, 0.1

Table 8.1 A Comparitive Table of the Efficiency of the Two Algorithms which use Dynamic Coordination

STARTING POINT	ITERATION No	PERFORMANCE INDEX	CONSTRAINT SET ACTIVITY			
			CU	CU		CU
A	1	4.527806	0.0	0.6522	0.0	0.2318
	2	4.161889	0.0	0.6521	0.0	0.0226
B	1	6.267736	0.0	0.6845	-0.001	0.1632
	2	4.14660	0.0	0.6522	0.0	0.2265
	3	4.162402	0.0	0.6521	0.0	0.022
C	1	30.842911	-0.004	0.6076	0.0	0.9993
	2	4.010699	-0.001	0.6536	0.0003	0.2275
	3	4.151252	-0.001	0.6522	-.0001	0.2264
	4	4.16240	.00001	0.6522	0.0	0.2264

Table 8.2 Effect of Change in Activity of Constraints on the Newton-like ALgorithm

Parameters used forsimulation in obtaining the above results.

Penalty factor = 0

Non-linearity parameters = {β β β}
= {0,0,0}

Starting Points

A
C = {0.14,1.3,1.2,-0.27, 0.007,-0.7,0.3}
U = {0.5,-0.18,0.24,-0.02}
P = {-0.516,0.023,0.615, -0.419}

B
C = {0.531,0.848,0.991,-0.148 0.0031,-0.5051,0.3426}
U = {0.1729,0.0287,0.331,0.005}
P = {-0.5981,-1.4307,0.6852, -0.44631}

C
C = {0,0,0,0,0,0,0}
U = {0,0,0,0}
P = {0,0,0,0}

	ITERATIONS		GAINS		SET-POINT CHANGES	
AUGMENTATION	RELAXED ALG	NEWTON-LIKE ALG			RELAXED ALG	NEWTON-LIKE ALG
0.0	NO-CONVERGENCE	3	-	-	---	21
0.5	67	3	0.1	0.2	469	21
0.75	43	3	0.2	0.2	291	21
1.0	35	4	0.3	0.3	245	28
1.5	34	4	0.35	0.4	238	28
2.0	33	4	0.4	0.5	231	28
3.0	44	4	0.4	0.6	308	28
4.0	61	5	0.4	0.5	427	35
5.0	----	5	---	---	---	35

Table 8.3 Sensitivity of the Two Dynamic Coordination Algorithms to the Penalty Factor with Non-linearities Kept Constant

AUGMENTATION	0.0	0.5	0.75	1.0	1.5	2.0	4.0	6.0
ITERATIONS	3	3	3	3	3	3	3	3

Table 8.4 The Performance of the Newton-like Algorithm with Linear model and System Mappings

	ITERATIONS		GAINS		SET-POINT CHANGES	
NON-LINEARITIES	RELAXED ALG	NEWTON-LIKE ALG			RELAXED ALG	NEWTON-LIKE ALG
0.1, 0.1, 0.25	36	4	0.35	0.5	252	28
0.25,0.1, 0.25	43	4	0.3	0.4	301	28
0.5, 0.1, 0.25	--	4	----	---	---	28
1.0, 0.1, 0.25	75	4	0.15	0.3	525	28
1.5, 0.1, 0.25	--	5	----	---	---	35
2.0, 0.1, 0.25	77	5	0.15	0.2	539	35
3.0, 0.1, 0.25	NO CONVERGENCE	5	----	---	---	35
4.0, 0.1, 0.25	-DO-	5	----	---	---	35
5.0, 0.1, 0.25	-DO-	6	----	---	---	42
6.0, 0.1, 0.25	-DO-	6	----	---	---	42
10.0,0.1, 0.25	-DO-	6	----	---	---	42

Table 8.5 Sensitivity of the two Dynamic Coordination Algorithms to Variations in the Non-Linearities of the System

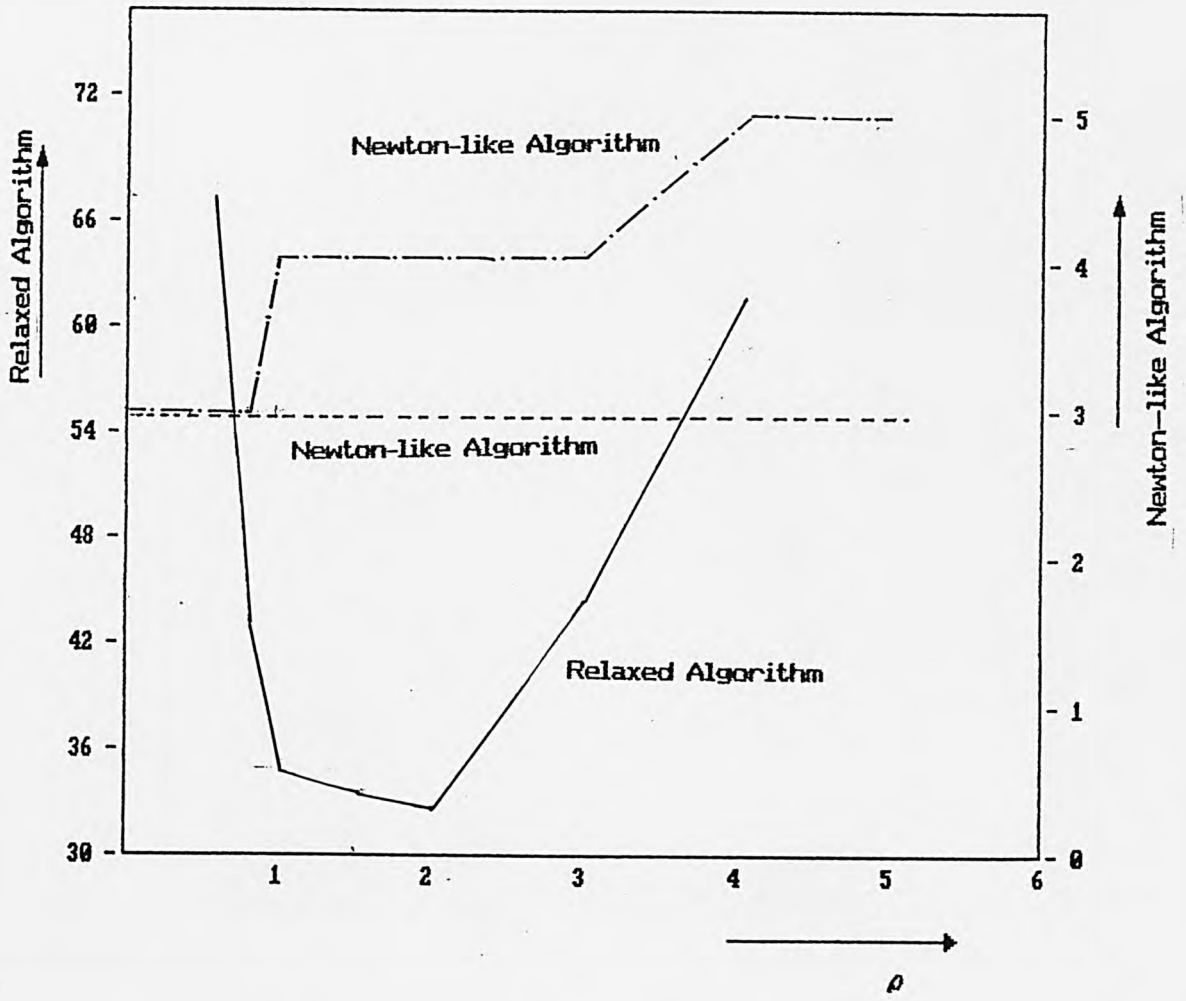


Figure 8.2 Graphical Representation of the Sensitivity Analysis of Tables 8.2 and 8.3

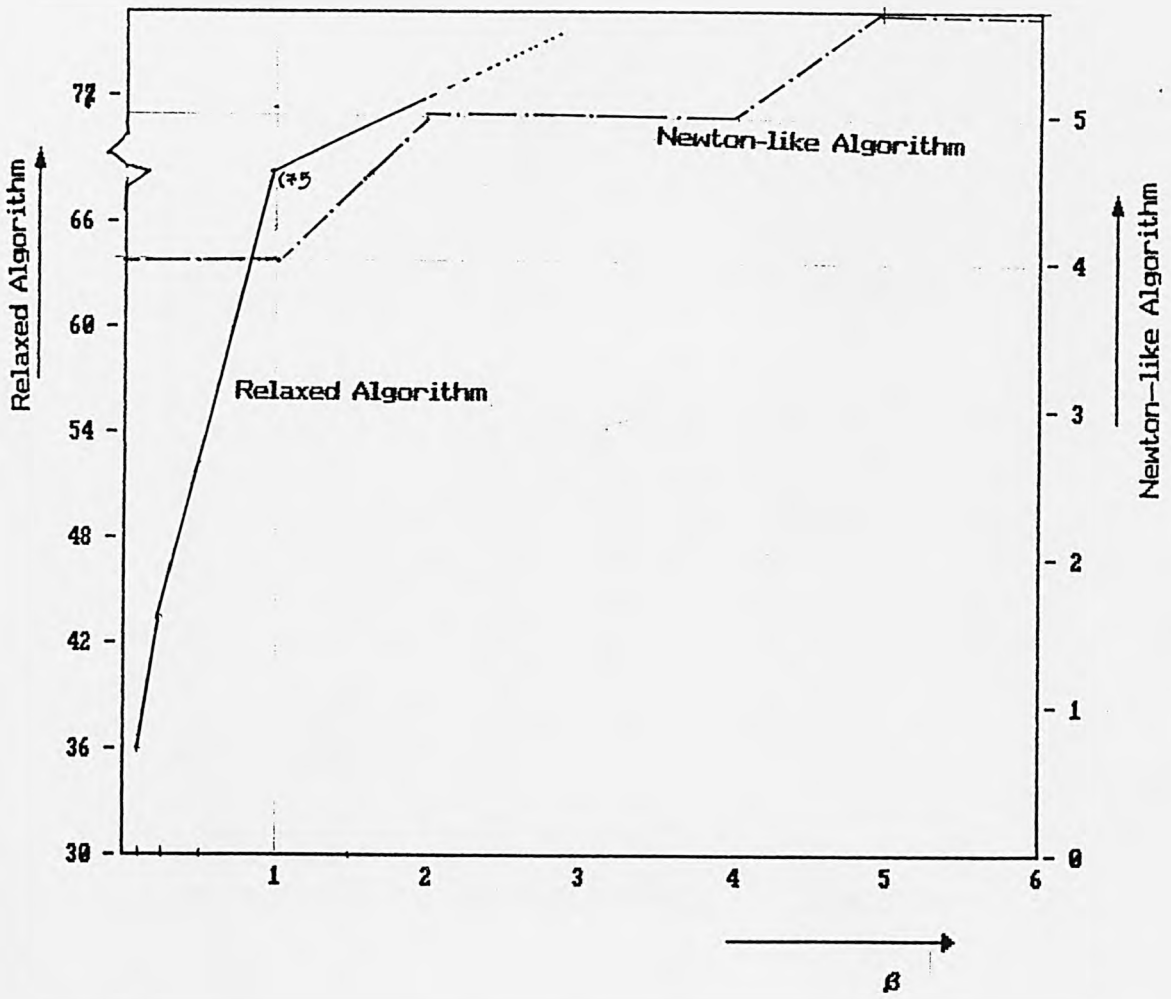


Figure 8.3 Graphical Representation of the Sensitivity Analysis of Table 8.5

EXAMPLE	ITERATIONS	REAL OPTIMUM	OPTIMUM OBTAINED	GAINS
A	4	0.18834	0.1884	1.0
B	26	0.34312	0.34309	0.9

Table 8.6 Simulation Results Using Passive Coordination Technique when Convergence is Attained

Figure 8.4 Performance Index Vs Iterations Using Passive Coordination (Example A)

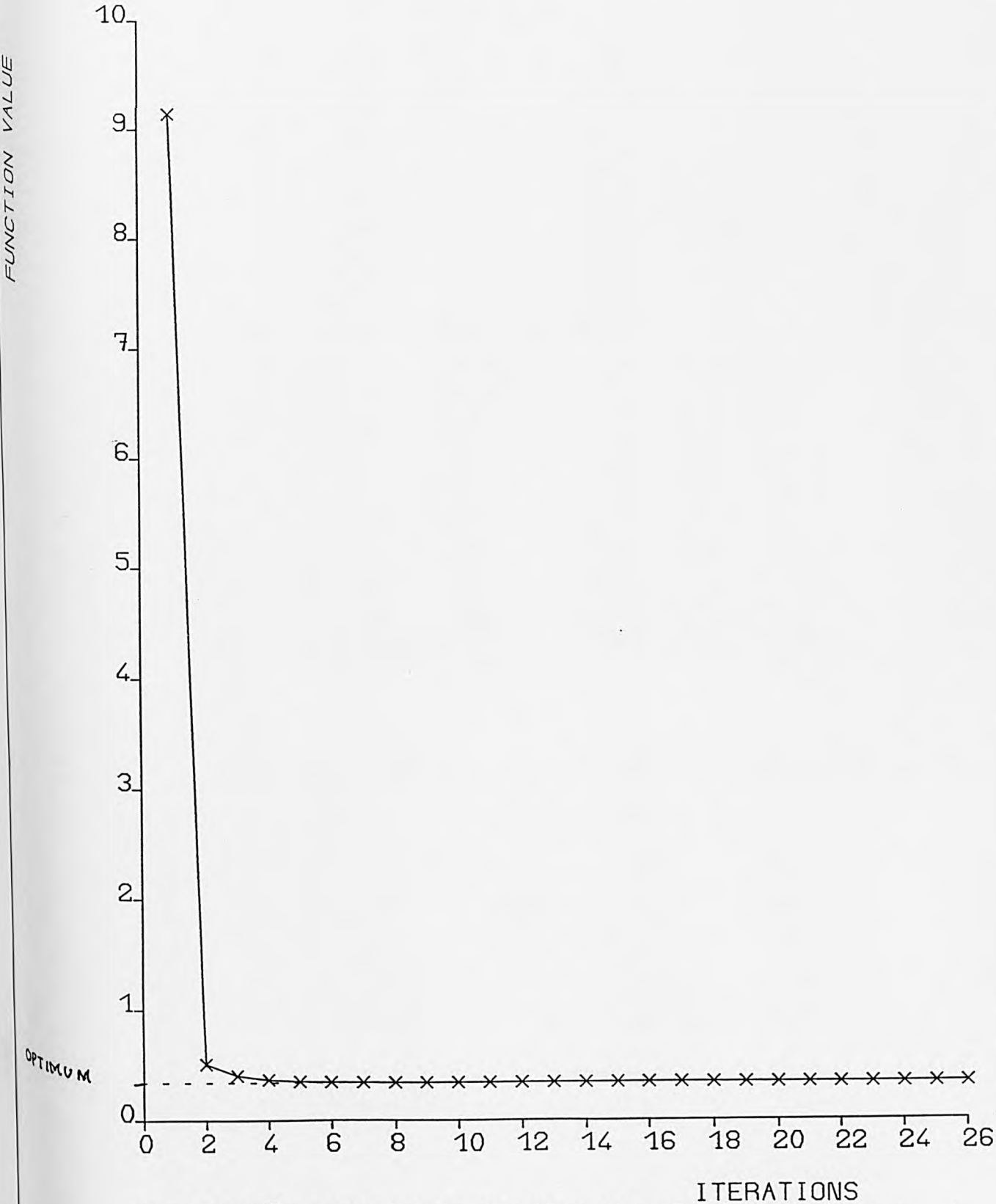
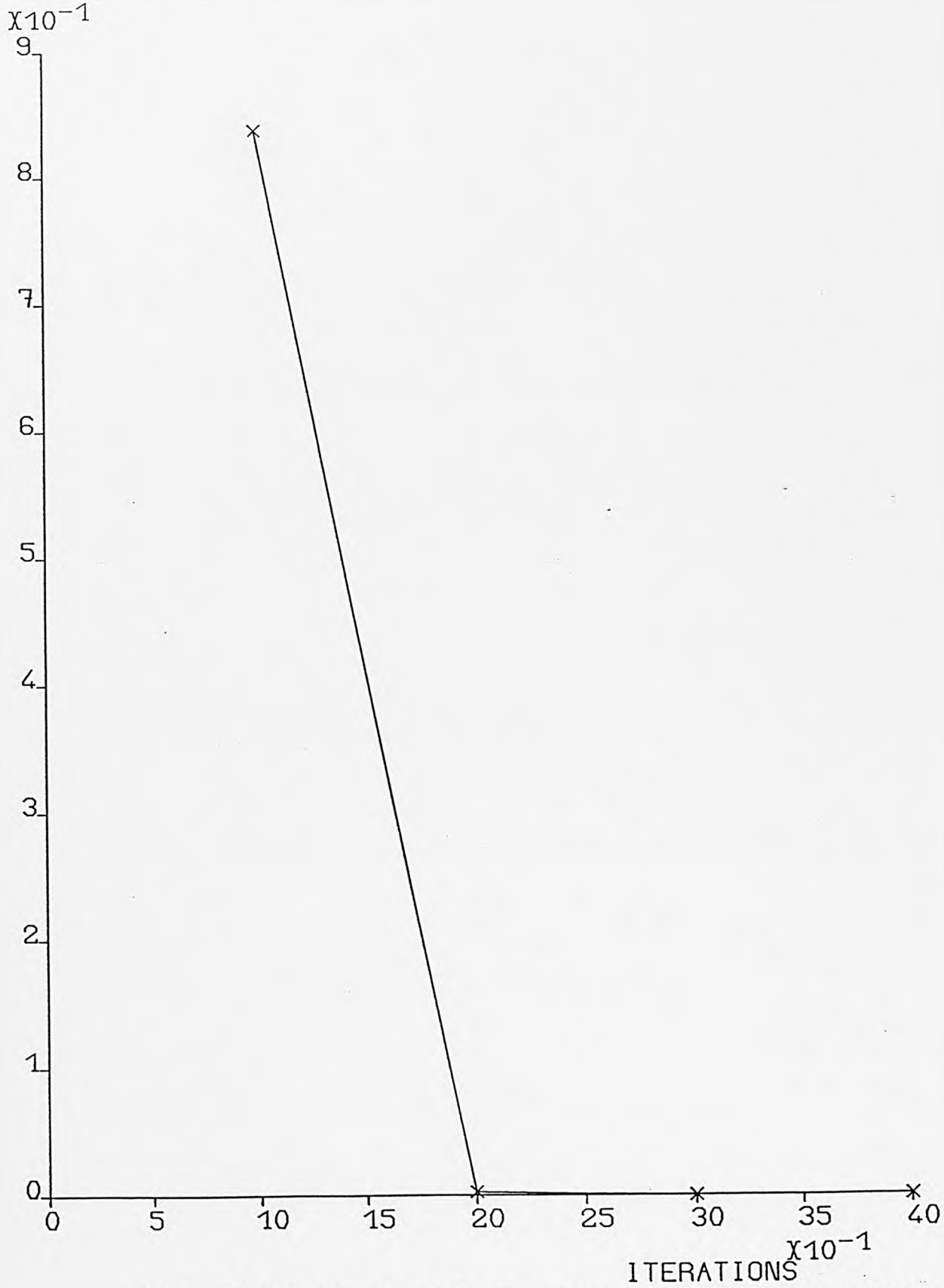
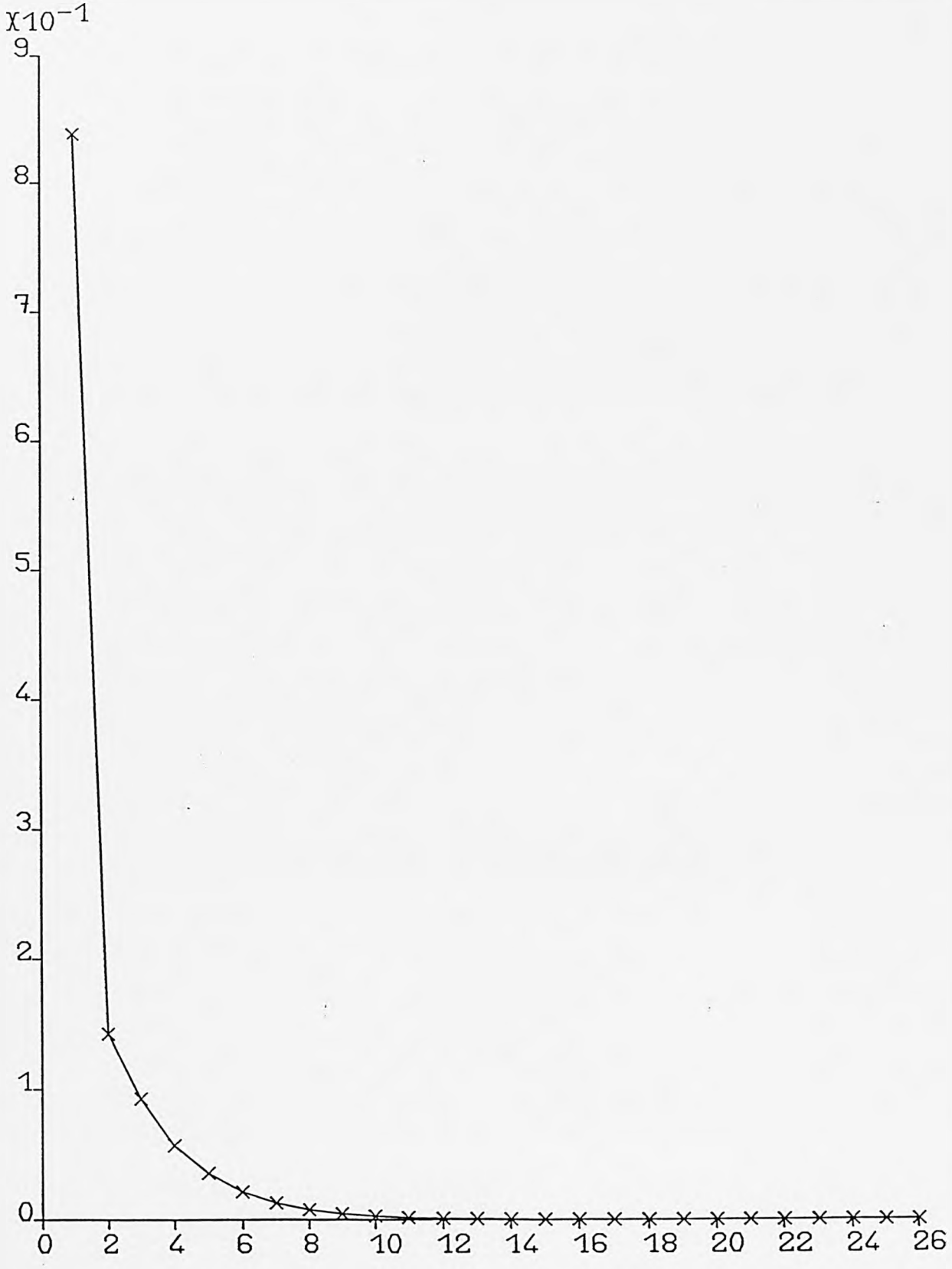


Figure 8.5 Performance Index Vs Iterations Using Passive Coordination

(Example B)



8.6 Function $T_1(V)$ Vs Iterations Using Passive Coordination (Example A)



8.7 Function $T_1(V)$ Vs Iterations Using Passive Coordination (Example B)

CHAPTER 9 GENERALISATION OF INTEGRATED SYSTEM OPTIMISATION AND PARAMETER ESTIMATION TECHNIQUES

9.1 INTRODUCTION

Consider a group of system optimisation algorithms, which have the following features :

a) Each algorithm solves the same problem.

b) The problem, arrived at after some form of an analysis is equivalent, in each case, to the original problem.

c) The analysis through which each algorithm is arrived at is the same in all cases.

d) Because of (b) and (c) the algorithmic and control structure are similar in all cases.

If such a group exists, then, it can be seen that all the algorithms will have similar properties, and hence the optimality and convergence conditions should be the same for each algorithm.

Then consider the family of ISOPE algorithms. It can be seen from the algorithms described in this thesis, and those given by **Brdys and Roberts (1986a), Ellis et al (1986),** which are classified as ISOPE, possess all the above features. Then the logical conclusion one can draw from here is that these algorithms can be generalised.

Then, in this chapter we generalise these algorithms. We have already presented the properties of the centralised algorithms, and the passive coordination algorithm in chapters 4 and 7; and hence a detailed description of these will be omitted here. However, we shall select one more algorithm which has not been described thus far. Then we present the generalised algorithm and finally give the generalised properties for these algorithms. It should be noted here that the generalisation is made for all the algorithms. However, it has been found to be difficult to prove that the conditions are enough for the relaxed dynamic algorithm described in chapter 6. However, because this algorithm has the features mentioned above it is assumed that these are also valid for this algorithm.

It should be noted that the generalisation is made for the relaxed algorithms where the performance index is augmented. The algorithms, where the performance index is not augmented, will be special cases of the augmented

algorithms, with the penaly factor ρ set to zero. Then the algorithms which are selected are as follows :

a) ISOPE1 : The MSTP algorithm.

b) ISOPE2 : The algorithm proposed by **Brdys and Roberts (1986a)**, which was later used in an augmented manner by **Abdullah (1988)**.

9.2. GENERALISED ALGORITHMS

The key feature of the ISOPE algorithms is to replace the problem given by (5) , after an analysis of the first order optimality conditions **(Roberts (1979), Brdys and Roberts (1986a))**, by an equivalent problem which is ultimately decomposed into :

a) A parameter estimation problem

b) A modified model-based optimisation problem

c) A coordination problem

In order that a generalised form of the ISOPE algorithm can be developed, let us define the following generalised variables :

For ISOPE1

$$\left.\begin{array}{l} X = [C];\ X \in \Re^n \\ Z = [V];\ Z \in \Re^n \end{array}\right\} \qquad (9.1)$$
$$\mathfrak{G} = \{\ (X) : G(C,Y) \leq 0\ \}$$

and, for ISOPE2:

$$\left.\begin{array}{l} X = [C,\ U];\ X \in \Re^{n+m} \\ Z = [V,\ W];\ Z \in \Re^{n+m} \end{array}\right\} \qquad (9.2)$$
$$\mathfrak{G} = \{\ (X) : G(X) \leq 0\ \}$$

Then using the generalised variables we can now, formulate the generalised system optimisation problem as:

$$\min_{X \in \mathfrak{G}} \left\{ q(X,\alpha) + \rho\phi(X,Z) \right\}$$
$$\text{s.t } \{\ X = Z;\ F(Z,\alpha) = K_{\#}(V)\ ;\ e(X,\alpha) = 0\ \} \qquad (9.3)$$

where $q(X,\alpha) \triangleq Q(X,Y)$

$e(X,\alpha) \triangleq U - HF(X,\alpha)$ (for ISOPE2 only)

$\phi(X,Z) = \|V - C\|^2$ (for ISOPE1)

$\phi(X,Z) = \|V - C\|^2 + \|W - U\|^2$ (for ISOPE2)

In addition, if the non-augmented forms of the algorithm are to be used, then $\rho = 0$. After analysis of the first order conditions of optimality for (9.3), the equivalent decomposed problem is:

The Generalised Parameter Estimation Problem

$$F(Z,\alpha) = K_{\sharp}(V) \text{ ----} > \alpha \tag{9.4}$$

The Generalised Modified Model Based Optimisation Problem

$$\min_{X \in \mathfrak{G}} \left\{ q(X,\alpha) + \lambda^T X + P^T e(X,\alpha) + \rho\phi(X,Z) \right\} \tag{9.5}$$

$P \in \Re^m$ is known as the price vector, and takes account of the possible imbalances between the various sub-systems in ISOPE2.

The Generalised Coordination Problem

$$X = Z \tag{9.6}$$

$$U = HF(X,\alpha) \quad \text{(for ISOPE2 only)} \tag{9.7}$$

$$\lambda = \left[F'_{z}(Z,\alpha) - K'_{\sharp}(V)\right]^T \left[Q'_{Y}(X,F(X,\alpha)) - H^T P\right] \tag{9.8}$$

It should, however, be noted that there are important differences on how these separate problems are solved depending on whether ISOPE1 or ISOPE2 is being applied.

For the decentralised problem, ISOPE2, the parameter estimation and the modified optimisation problems are separated into N local problems. in manner similar to that in chapter 6. In the ISOPE2, the modifier, λ acts as a genuine coordination variable and can be written in the partitioned form

$$\lambda^T X = \sum_{i=1}^{N} \lambda_i X_i \tag{9.9}$$

where the local modifier, λ_i effectively contains coordinating information about distant sub-systems.

In ISOPE1, the centralised case, the coordination problem does not exist is the strict sense and the function of λ is to simply modify the performance objective and ensure the satisfaction of the optimality conditions for the overall problem.

9.2.1 The Algorithmic Structure

The algorithmic structure is then:

STEP I : Set k=0

Select suitable values for X^k and if ISOPE2, for P^k

Select scalars ε_z , ε_p such that $0 < \varepsilon_z, \varepsilon_p \leq 1$

Set $\rho = 0$ IF using the non-augmented problem

Else select a suitable value for the penalty factor ρ

STEP II: Through local problems set

$$Z^{k+1} = Z^k + \xi_Z(X^k - Z^k) \tag{9.10}$$

For ISOPE2 only set

$$P^{k+1} = P^k + \xi_P(U^k - HF(X^k,\alpha)) \tag{9.11}$$

Set k <--- k+1

STEP III: Apply Z^k to the model and V^k to the system and take measurements $K_*(V^k)$ and evaluate the parameters α

STEP IV: Apply additional perturbations about V^k to the system to get a finite approximation for $K'_*(V^k)$. Use the model, at Z^k, to evaluate the remaining terms in the expression for the modifier, λ. Evaluate λ and partition into local modifiers λ_i to be used in the local optimisation problem.

STEP V: Solve the local modified model-based optimisation problem and collect the results to obtain X^k.

STEP VI: Check for satisfacton of the coordination conditions.
If (yes) STOP, else go STEP II.

The vector z is updated in STEP II, using an under-relaxation scheme (Cohen (1978)) and represents the applied controls and the values of the interaction inputs used in the parameter estimation problem. The scalars ξ_Z are known as the iterative loop gains.

9.3. GENERALISED CONVERGENCE CONDITIONS

One of the remarkable features of the ISOPE algorithms is the manner in which they are derived. Even though they start from different problem formulations eventually they have the same structure. It is because of this that these algorithms can be generalised. The convergence conditions for the ISOPE algorithms are now stated.

The conditions derived here are for both the augmented as well as the unaugmented algorithms. If, in these conditions, the value of the penalty factor, ρ, is set to zero, the conditions for the algorithms where the performance index is not augmented are obtained. It is in this sense that these conditions are termed generalised convergence conditions. The conditions derived here use some of Cohen's (1980) techniques. Let Ω be the algorithmic solution set.

GENERALISED CONVERGENCE THEROEM (GCT)

Assume that

1. Every point of the set $\mathcal{S}$ satisfies the regulatory conditions.
2. The set $\mathcal{S}$ is compact and convex.
3. The functional $q'_x(\,.\,,\alpha)$ is uniformly monotone on $\mathcal{S}$ with some constant $a(\alpha) > 0$ for every $\alpha \in \mathcal{A}$ and also that $q'_*(\,.\,)$ is also uniformly monotone on $\mathcal{S}$ with some constant $a^* > 0$.

   where $\mathcal{A} = \bigcup_{z \in \Omega} \alpha(z)$

4. There exist such numbers $\bar{A}$ and $\underline{a}$ such that

$$\bar{A} - 2\rho \geq A(\alpha) \qquad a(\alpha) + 2\rho \geq \underline{a} > 0 \tag{9.12}$$

   where $A(\alpha)$ is the Lipschitz constant of $q(\ ,\alpha)$ on $\mathcal{S}$.

5. The model is selected such that

$$\bar{A} \leq 2a^* - \frac{1}{2}A^* \tag{9.13}$$

$$\frac{3}{4}A^* < \underline{a} \leq A^* \tag{9.14}$$

   where A^* is the Lipschitz constant of q_* on $\mathcal{S}$

6. The class of systems considered is restricted in a manner

$$a^* \geq \frac{5}{8}A^* \tag{9.15}$$

7. The mappings $F(x,\alpha)$ and $K_*(V)$ are linear for every $\alpha \in \mathcal{A}$

Then

I. There exists a solution $\bar{z}$ of the OCP and for algorithms of type ISOPE2 every point belonging to the solution set is of the form $(\bar{z},\bar{P})$.

II. There exists such scalars ξ_1 , ξ_2 , such that $0 < \xi_1 < 0.5$, $0.5 < \xi_2 \leq 1$, such that for every $\xi_z \in (\xi_1,\xi_2)$ the algorithm generates a sequence (z^k) which converges to $\bar{z}$ and provided (for algorithms of type ISOPE2) there exist additional scalars ξ_p such that

$$0 < \xi_p \leq \xi^k{}_p < \left[\frac{\underline{a}}{\gamma} \right]$$

where $\gamma = \max \{ 1, r^2\|H\|^2 \}$ and r is the Lipschitz constant of $K_*()$ on $\mathcal{S}$.

The proof for this theorem for ISOPE1 is given in chapter 4, and is omitted here. The proof for ISOPE2 is given by Brdys and Roberts (1985), and is hence omitted here.

Notice here the restriction placed on the selection of the model. The model should have properties such that it satisfies the conditions (9.13) and (9.14). The advantage of augmenting the problem with the quadratic penalty factor is clearly seen through the conditions posed on the selection of the model. If the penalty factor, ρ, is set to zero, then condition 4, Equ(9.12), becomes

$$\bar{A} \geq A(\alpha) \qquad a(\alpha) \geq \underline{a} > 0 \tag{9.16}$$

This is now more stronger than is necessary when augmenting the problem. With appropriate selection of the penalty factor the conditions given for the model selection will always be satisfied.

9.4. DISCUSSION AND CONCLUDING REMARKS

The algorithms discussed here are representative of the various ISOPE algorithms. The reason one can generalise these algorithms is the manner in which they are derived, and they all have a similar structure. However, there is an exception to this generalisation. The Approximate Linear Model approach, deviates from the algorithms discussed here, in that a model is not provided to the algorithm a priori. However, by changing the wording of the model selection conditions from 'The model is selected such that' to 'The algorithm selects a model such that' it is seen that the rest of the conditions remain the same.

The important condition to note here is condition 7. This places a severe restriction on both the model and the system, in that they should be linear. However this condition is valid only for those algorithms which resemble ISOPE2 only and not necessary for all the other algorithms. However, this is only a mathematical limitation, as simulation results have shown, **(Abdullah (1988), Tatjewski et al (1987c))**, that algorithms of type ISOPE1 converge even if the relevent mappings are not linear.

We have already mentioned that, it has proved difficult to prove the above theorem for the dynamic relaxed algorithm described in chapter 6. However following the reasoning given in section 9.1, we can assume that the above theorem is valid for that algorithm also.

It is also important to note here that although generalised results for the algorithms can be derived, one cannot say that the rate of convergence will be the same for all the algorithms. This is because of the modifications made to the OCP to get the required form of the problem for the algorithm. The numbers $\underline{a}$, which bear a relationship to convergence rate will necessarily be subject to variation.

CHAPTER 10 CONCLUSIONS AND SCOPE FOR FUTHER WORK

10.1 INTRODUCTION

The notion that the design of an optimising control system for any system can be divided into two parts, one part seeing to it that the system performs at some desired level (regulatory control) using, perhaps, a standard three term controller, and another part which deals with optimising the performance of the system (Supervisory control), is well established (**Lefkowitz (1977), Mesarovic et al (1970), Findeisen et al (1980)**). Then, if the reference inputs to the regulatory controllers are the solutions of the system optimisation problem, the system would then be performing at its optimum. This thesis was concerned with the problem of solving the optimising control problem. That is the supervisory problem.

At the outset, we had stated the need for developing techniques which use an adaptive model to solve the optimising control problem. It was also stated that, a technique, MSTP, was proposed by **Roberts (1979)**, which uses an adaptive model, which provided the optimal solutions for the system optimisation problem. This method was found to be rather efficient in solving the problem on hand, even when the model had a structure which did not correspond to the real system it was supposed to model (**Ellis and Roberts (1982)**. In fact this was also brought out by **Brdys and Roberts (1987)**, where the convergence was proved under very weak conditions. These conditions required that the model based problem be convex, and no other extraordinary conditions were posed. Here no restriction, whatsoever, was put on the structure of the model. Then, under such conditions, it is not very surprising that this algorithm is very powerful indeed. This algorithm, MSTP, has been extended to improve its performance and also to solve the decentralised problem (**Michalska et al (1986), Brdys and Roberts (1986a), 1986c, 1986d, Chen (1986), Abdullah (1988), Ellis et al (1988), Kambhampati and Ellis (1988), Tatjewski and Roberts (1987a, 1987b)**).

The objectives of this research were stated in chapter 1, and these are briefly:

a) To understand what are the model-reality differences.

b) To determine what is the significance of the modifier in ISOPE algorithms, and how far are the model-reality differences reflected in it.

c) To determine the effect of the augmentation of the performance index have on the properties of the algorithm.

d) To determine how to reduce the need for derivatives, of the outputs of the real system.

e) To examine the possibilities of generalising the ISOPE algorithms.

Of course, in answering the above, the objective of studying and analysing the properties of the algorithms is also undertaken.

10.2 CONCLUSIONS

The analysis presented in chapter 4, enables us to complete some of the above objectives. The analysis, of chapter 4, was made for the centralised algorithms where, although the system could consist of several sub-systems, we were concerned with the global system taken as a whole. This analysis is, then, clearly valid for the case where the global problem is decomposed into several sub-problems corresponding to sub-systems, which make up the total system.

Through the analysis, which was presented in chapter 4, we can come to the following conclusions :

a) That, apart from the model parameters which are estimated, the difference in derivatives of the model outputs with respect to the set-points and corresponding derivatives of the system constitute the model-reality differences. In fact, from the simulation results shown in chapter 4, it was seen that these derivative differences play a greater role in describing the model-reality differences, than do the parameters.

b) The second objective, regarding the significance of the modifier is again quite apparent from the analysis shown in chapter 4. It was shown that the modifier could have two interpretations, viz:

(i) As parameter which quantifies the violations of the sufficiency conditions by the model

or

(ii) As compensator, which compensates for the differences in the properties of the model based performance index and the system based performance index.

c) The third objective follows from that of the second objective. If we

interpret the modifier, λ, as a compensating parameter, the effect of the augmentation is such that the penalty factor reduces the differences in the properties of the two indices. Thus, if the differences are large than the value of the penalty factor must also be correspondingly large.

The important conclusion here, is that the derivative differences play a greater role in describing the model-reality differences. For whatever the interpretation we give to the modifier, the derivative differences are ultimately reflected there. The stability of the iterative process depends on these differences. A very large difference, would cause the iterative process to become unstable, as seen in chapter 5, and would result in our selecting very low gains. This in turn results in a very slow convergence to the optimum. Again, if these derivative differences are large, it can be seen that the model will not satisfy the conditions given in theorem 4.3.1.1. This conclusion is verified by the simulation results presented in chapter 5. This is more apparent with the results obtained by simulating the fuel–gas mixing system. The linear model used here is not structurally correct, but its derivatives were closer to the reality and hence it gave better results than the second model which had similar structure to the reality but some of the physical constants were varied by 5%.

It has been stated that the drawback with the ISOPE algorithms is that derivative information from the system is needed, so that the optimisation problem can be solved. This need comes about because of the integration of the two problems of system optimisation and parameter estimation. This results in one of the constraints becoming based on the system mapping, which is $F(C,\alpha) = K_{\sharp}(C)$. As long as this constraint is present in the problem, and there are no other effective method of solving the integrated problem, it is felt that there is no alternative other than to use derivative information. Of course some techniques have been proposed (Chen (1986)), but these are not more efficient than the ISOPE techniques.

If one cannot eliminate this need for derivative information, then the only other alternative is to develop algorithms which reduce this requirement. That is algorithms which converge faster, and as a result fewer number of iterations are needed for attaining the optimum. It is towards this end that the augmented and the Newton-like algorithms were developed. These techniques have been used in both the centralised as well as a decentralised situations.

One simple method, which has been proposed in this research is the ALM–ISOPE approach. This technique can be used only in the centralised case as

the model developed by the algorithm is not decomposable. This approach, is a logical extension of the conclusion that the derivative differences describe the model–reality differences. If this is the case then, an iterative process which develops a model using the system derivatives would function relatively more efficiently than the MSTP. This conclusion is easily verified, by the simulation results presented in chapter 5. This particular approach can be viewed as a system linearisation approach, where we linearise the system at a given point. This results in the model so developed being a better description of the reality, in the region around the point at which the linearisation has been performed. This method however, cannot be extended for the decentralised cases, for which the reason has already been mentioned.

Other techniques which improve the performance of the ISOPE algorithms, both centralised as well as decentralised, are the augmented algorithms. In both these techniques, the performance index is augmented by a quadratic penalty function. The effect of this augmentation is that the resulting convergence conditions are weakened, as can be seen in chapters 4 and 9. For the sake of clarity we give the conditions posed, in general for all ISOPE algorithms where the performance index is augmented, on the model again (see chapter 9)

1. There exist such numbers $\bar{A}$ and $\underline{a}$ such that

$$\bar{A} - 2\rho \geq A(\alpha) \qquad a(\alpha) + 2\rho \geq \underline{a} > 0 \tag{10.1}$$

2. The model is selected such that

$$\bar{A} \leq 2a^* - \frac{1}{2}A^* \tag{10.2}$$

$$\frac{3}{4}A^* < \underline{a} \leq A^* \tag{10.3}$$

Here if the penalty factor, ρ, is set to zero we see that the resulting conditions are stricter. However, the problem, with such an augmentation is that if a proper value of the penalty factor is not chosen it could be possible that there is no effect on the performance of the algorithm, or else the problem could become ill–conditioned.

With ISOPE algorithms, we need to select some parameters known as the iterative loop gains, which are used in the under–relaxation scheme, for updating the control set–points. Generally, these gains are obtained through a trial and error procedure. This, when implementing the algorithms in an on-line situation would pose problems as there is no way of knowing what the value of these gains should be. Then, when using an augmented approach, it is seen that the number of parameters which have to be selected through a trial and error procedure increases. A not so proper selection of these parameters could have far reaching results on the performance of the algorithms. The reasons for selecting

appropriate gains has been discussed in chapter 4. A very high value for the gains could lead the algorithm to become unstable, and similarly a low value could make the iterative process very slow. Similar is the case with the penalty factor. A low penalty factor, would not result in any appreciable improvement in the performance of the algorithm, and a high penalty factor could make the problem ill-conditioned.

To overcome these difficulties the Newton-like algorithms have been proposed. It has been seen, whether in the centralised case **(Tatjewski and Roberts (1987))**, or in the decentralised case (chapter 8), the Newton-like algorithm overcomes these problems rather well. In this method the updating scheme, for the controls, is more precise as opposed to the approximate method of the relaxation scheme. This results in the algorithm converging to the optimum very quickly. In fact for the case where the system and model mappings are linear it is seen that we can have superlinear convergence. It also is seen that within reasonable limits of the penalty factor value, the performance of the algorithm remains fairly constant. This is in marked contrast to the algorithms which use the relaxation scheme, where, within reasonable limits, it is seen that as the value of the penalty factor is increased the performance of the algorithm is improved but after some value this performance deteriorates, (see chapter 8, **(Roberts et al (1986))**. Also the Newton-like algorithm can handle large model-reality differences better. It is seen that, as these model-reality differences are increased, (by using the non-linearity parameters (see chapter 8)), the deterioration in the performance of the relaxed algorithm is more than it is with the Newton-like algorithm.

We have often repeated that the algorithms which are classified under ISOPE, that is, those which are described in this thesis and those described elsewhere **(Michalska et al (1985), Bryds and Roberts(1986a), Chen (1986), Abdullah (1988), Ellis et al (1986), Kambhampati and Ellis(1988))**, have some common features. It is these features which help in the generalisation of the ISOPE algorithms. This generalisation was presented in chapter 9. However, as mentioned it was not possible to include one algorithm, namely the algorithm using dynamic coordination. However, because of the features mentioned in chapter 9, we can conclude that it should be possible to fit this algorithm into the general scheme.

10.3 FURTHER WORK

It has been often repeated that the disadvantage with the ISOPE algorithms is that derivatives of the system are needed. Efforts have been made to either totally eliminate this necessity or to make the algorithm converge faster. The algorithms, which have been developed to make the convergence faster have proved

successful and some of them have been discussed in this thesis. On the other hand, the attempts to eliminate the need for the derivatives have been sucessful to a limited extent. The algorithm developed by **Liu et al (1988)** has been successful. However, it has been found that the efficiency of the algorithm reduces as the number of variables, that is the set-points, increases. These algorithms, however, are suitable for the centralised cases and attempts have to be made to extend this for the decentralised cases.

There is one definite advantage, with the centralised structures. This is that whatever the form the algorithm takes, whether we use a linearisation technique as was done with the ALM-ISOPE or with the point-linearized inout-output mapping technique of **Hu et al (1988)**, one does not have to ensure that the model is separable into different sub-models. This is the case with the decentralised algorithms. It is for this reason that efforts to extend these algorithms to the decentralised cases have proved to be difficult. However, these techniques when incorporated with the existing algorithms, it is possible to develop some efficient algorithms (**Lin et al (1987a, 1987b, 1987 c, 1987d)**). These use a two model technique, where we not only use the model provided, but also develop an approximate linear model along the lines of the ALM-ISOPE technique. Here, this linear model is used to update the price vetors. Thus, instead of using the following

$$P^{k+1} = P^k + \epsilon\,(U^{k+1} - HF(C^{k+1}, U^{k+1}, \alpha^k)$$

we use

$$P^{k+1} = P^k + \epsilon\,(U^{k+1} - H(WC^{k+1} + \alpha^k)$$

where $\epsilon \in \Re^m$ and $W \in \Re^{mXn}$ are as before.

The advantage here is that this linearised model is a better representation of the system near the point considered, and hence would be a better measure of the interactions taking place.

From the survey presented here in this chapter and else where in the thesis, it can be seen that, here at the Control Engineering Centre, City University, a lot of effort has been and is being put into developing algorithms for solving the system optimisation problem. Most of the work done so far has been mostly theoretical, and the simulation have used abstract examples. For any algorithm to be practical, it has to be tested on some existing system, like a pilot scale plant, of which they are quite a few in the laboratory here in the City University. Other than the MSTP and AISOPE algorithms, which have been sucessfully implemented on pilot scale plants **(Brdys et al (1986b), Bakalis (1986))**

in the laboratory, none of the other algorithms have been tested on real situations. This task is, however, rather enormous, as there are numerous algorithm which can be used. However, it is expected that the ALM-ISOPE and the Newton-like algorithms would be the most efficient, as simulation results have shown them to be so.

Then the major project which has to be undertaken is to test these ISOPE algorithms, either all of them or those which have been seen, through simulation to be the most efficient.

REFERENCES

Abdullah. N (1988) " Augmented Integrated System Optimisation And Parameter Estimation for On-Line Hierarchical Control of Large Scale Industrial Processes — PhD. Thesis, City University, London, U. K

Brdys. M, and Ulanicki. B (1978) " On the completely Decentralised Control with Local Feedback in Large Scale Systems" — Arch. Autom. Telemach, vol 23, pp 21-36

Brdys. M, and Roberts. P. D (1985) "Hierarchical Single and Double Iterative Techniques for Optimising Control of Large Scale Industrial Processes : Derivation, Applicability and Convergence" — IEE Int. Conference CONTROL 85, University of Cambridge

Brdys. M, and Roberts. P. D (1986a) " Optimal Structures for Steady State Adaptive Optimising Control of Large Scale Industrial Processes " — Int. J. Sys. Sc. — Vol 17, No 10, pp 1449 — 1479

Brdys. M, Ellis. J. E, Roberts. P. D (1986b) " Augmented Integrated System Optimisation and Parameter Estimation Technique — Derivation, Optimality and Convergence " — IEE Proc. Part D. pp 134 — 201

Brdys. M, Roberts. P. D., Badi. M. M, Kokkinos. I. C (1986c) " Model Based Double Iterative Strategy for Integrated System Optimisation ans Parameter Estimation of Large Scale Industrial Processes " — IFAC/IFORS Symposium on Large Scale Systems Theory and Applications, 4^{th}, Zurich

Brdys. M, Chen. S, Roberts. P. D, (1986d) "An Extension to the Modified Two-Step Algorithm for Steady-State Optimisation and Parameter Estimation " — Int. J. Sys. Sci — Vol 17, No 8, pp 1229 —1243

Brdys. M, and Roberts. P. D, (1987) "Convergence and Optimality of the Modified Two-step Algorithm for Integrated System optimisation and Parameter Estimation" — Int. J. Sys. Sci.

Chen. S (1986) " Integrated System Optimisation and Parameter Estimation for On-line Control of Industrial Processes " — PhD Thesis, City University, London, U. K

Cohen. G, (1978) "Optimisation by Decomposition and Coordination : A Unified Approach " — IEEE Trans on Auto. Control — AC—23, 2, pp 222 — 232

Cohen. G, (1980) "Auxiliary Problem Principle and Decomposition of Optimisation Principles " — JOTA — Vol 23, No 3, pp 277 — 232

Durbeck. R. C, (1965) " Principle for Simplification of Optimising Control Models " — PhD. Thesis, Case Western Reserve University, Cleveland, Ohio, U. S. A

Ellis. J. E, and Roberts. P. D, (1982) " Measurement and Modelling Trade—offs for Integrated System Optimisation and Parameter Estimation " — Large Scale Systems, — Vol 3, pp 191 — 204

Ellis. J. E, and Roberts. P. D, (1985) " On the practical Viability of Integrated System Optimisation and Parameter Estimation" — IEE Int. Conference "CONTROL 85", University of Cambridge, Pub No 252, pp 281 — 285

Ellis. J. E, Roberts. P. D, and Michalska. H (1986) "Adaptive Models in Decentralised Optimising Control" — 4th IFAC/IFORS Symposium on Large Scale Systems Theory and Applications, Zurich

Ellis. J. E, Sheng. G. H, Kambhampati. C, and Roberts. P. D (1988) " Approaches to the Optimising Control Problem " — Int. J. Sys. Sci (To Appear)

Foord. A. G, (1974) " On-line Optimisation of a Petrochemical Complex" — PhD Thesis, University of Cambridge, U. K

Findeisen. W, Bailey. F N, Brdys. M, Malinowski. K, Tatjewski, P; Wozniak, A (1980) CONTROL AND COORDINATION IN HIERARCHICAL SYSTEMS —John Wiley and Sons : Chichester ,New York

Fletcher. R. (1970) " Function Minimisation Without Evaluating Derivatives — A

Review" — Comp. J. , 5, pp 405 423

Fletcher. R, (1981) " Practical Methods of Optimisation : Vol I and II" — J. Wil ey & Sons, Chichester.

Gill. P. E, Murray. W, and Wright. M. H (1981) " Practical Optimisation" — Academic Press

Hogan. W. W, (1973) " Point — to — Set Maps in Mathematical Programing" — SIAM REVIEW , Vol 15, No 3, pp 591 — 603

Hu. G. H, Lui. Z. Y, Roberts. P. D, Ellis. J. E (1988) "An Algorithmwith the Approximate Point Linearised Inout/Output Mapping for Integrated System Optimisation and Parameter Estimation" — City University Research Memorandum CEC/ GHH — ZYL — JEE — PDR/ 75 (CITY UNIVERSITY, London, U. K)

Kambhampati. C, Ellis. J. E (1987) " Approaches to the Centralised On-Line Optimisation Problem " — City University Research Memorandum (CITY UNIVERSITY, London, U. K) CEC/CK — JEE/ 55

Kambhampati. C, Ellis. J. E (1988) " A Passive Coordination Technique for Solving the On-Line Decentralised Optimising Control Problem " — IEE Int. Conference CONTROL 88, University of Oxford, Pub No 285 pp 532 — 535

Kantorovich. I. V, and Akilov. G, (1964) "Functional Analysis" — Pergamon Press

Lefkowitz. I, (1977) " Integrated COntrol of Industrial Systems" — Phil Trans. R. Soc. London, A 287, pp 443 — 465

Lasdon. L. S (1970) " Optimisation Theory for Large Scale Systems" — Macmillan, London

Luenberger D. G (1973) " Linear and Non-Linear Programming" — Addison—Wesley Publishing Company, Reading, Mass, U. S. A

Lin J, Hendawy. Z. M, Roberts. P. D (1987a) " An Extension of the Integrated System Optimisation and Parameter Estimation to Hierarchical Control of Steady State Systems with Output Dependent Constraints" — City University Research Memorandum (CITY UNIVERSITY, London, U. K) CEC/JL — ZMH — PDR/61

Lin J, Roberts. P. D, Kambhampati. C, (1987b) " A Two Model Approach in a Double Iterative Structure for Integrated System Optimisation and Parameter Estimation of Large Scale Industrial Processes" — City University Research Memorandum (CITY UNIVERSITY, London, U. K) CEC/JL — PDR — CK/70 (To be presented at the National Conference on Automatic Control, China, MAY 1988)

Lin J, Kambhampati. C, Roberts. P. D (1987c) " A New Two Model Approach in a Double Iterative Structure for Integrated System Optimisation and Parameter Estimation of Large Scale Industrial Processes" — City University Research Memorandum (CITY UNIVERSITY, London, U. K) CEC/JL — PDR — CK/71

Liu. Z. Y, Roberts. P. D (1988) " A Sequential Model Approximation Approach for Optimising Control of Steady State Systems" — City University Research Memorandum CEC/ ZYL—PDR/86, (CITY UNIVERSITY, London, U. K)

Mesarovic. M. D, Macko. D, and Takahara. Y (1970) "Theory of Heirarchical Multi—Level Systems" — Academic Press (New York)

Maxwell. M, (1983) "Distributed and Centralised Control for Profit" — Distributed Computer Control Systems 5th IFAC Workshop — Sabi—Sabi, South Africa, pp 39 — 45

Michalska. H, Ellis. J. E, and Roberts. P. D (1985) " Joint Coordination Method for Steady State Control of Large Scale Systems" — Int. J. Sys. Sci, Vol 16, No 5, pp 605 — 618

Ortega and Rheinboldt (1970) " Iterative Solution of Non-linear Equations in Several Variables — Academic Press

Bakalis. P. S, (1986) "On—Line Hierarchical Control" — PhD. Thesis, City University,

London, U. K

Powell. M. J. D (1964) " An efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives" — Comp. J — Vol 7, pp 155 — 162

Roberts. P. D (1979) " An Algorithm for Steady—State Optimisation and Parameter Estimation" — Int. J. Sys. Sci — Vol 10, pp 719 — 724

Roberts. P. D, Williams. T. W. C (1981) " On an Algorithm for Combined System Optimisation and Parameter Estimation" — Automatica — Vol 17, pp 199 — 209

Roberts. P. D, and Lalui. A (1982) " Integrated System Optimisation and Parameter Estimation of a Chemical Plant" — IEEE Conference on Application of Adaptive and Multivariable Control , Hull, U. K

Singh. M. G and Titli. A (1978) "Systems : Decomposition, Optimisation and Control" — Pergamon Press

Singh. M. G, Titli. A, and Malinowski. K, (1983) "Decentralised Control Design — An Overview " — 3rd IFAC/IFORS Symposium — Large Scale Systems — Theory and Applications, Warsaw, Poland

Sheng. G. H, Ellis. J. E (1985) " An Assessment of Adaptive Model Strategies for Optimum Operating Conditions" — City University Research Memorandum CEC/GS — JEE/21 (CITY UNIVERSITY, London, U. K)

Tatjewski. P, Roberts. P. D (1987a) "Newton-like Algorithm for Integrated System Optimisation and Parameter Estimation Technique" —" Int. J. Control, Vol 46, No 4, pp 1155 — 1170

Tatjewski. P, and Roberts. P. D (1987b) " Newton-like Algorithms for Hierarchical Steady State Control of Large Scale Processes — Part I : The Single Loop Algorithm, Part II : The Double Loop Algorithm — and Simulation Results — City University Research Memorandum CEC/PT — PDR/59 (CITY UNIVESITY, London, U. K)

Roberts, P.D; Abdullah, N; Tatjewski, P (1988) "Comparision of Some Algorithms for Hierarchical Steady-State Optimising Control of Interconnected Industrial Processess — IEE Control 88 (IEE) Conference - Oxford, No 285,pp 532-535

Zangwill. W. I, (1967) " Minimising a Function without Calculating Derivatives" — Comp. J Vol 10, pp 293 — 296

Zangwill. W. I. (1969) " Non—Linear Programming : A Unified Approach" — Prentice—Hall Inc, Englewood Cliffs, NJ, U. S. A

```
C ***************************************************************
C *                                                             *
C * PROGRAM FOR MSTP AND AISOPE ALGORITHMS. FOR MSTP SET R1=0.0 *
C * AND FOR AISOPE SET R1 TO ANYTHING ELSE (R1 IS THE PENALTY FACTOR*
C * SEE FLOW CHART TO SEE HOW THE PROGRAM WORKS                 *
C *                                                             *
C ***************************************************************
C
      PROGRAM MAISOPE
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION W(100),X(5),XVP(5),XIP(5),XUOP(5),EINC(5),
     &          YMDL(3),YREAL(3),YRLC(3),DEDV(5,3),COOR(5),YMODEL(3),
     &          DYDA(3,3),DFDA(3,1),UNIMAT(3,3),DADY(3,3),PE(3),
     &          DFDY(3,1),DFDV(5,1),DMDY(5,3),DRDY(5,3),
     &          BB(8,8),AA(8,8),WKSPCE(10),R1,DSQRTC,DSQRT
      INTEGER IW(7),IOPT
      COMMON/USED1/COOR,R1,XVP
      COMMON/USED3/TOC
      COMMON/USED2/PE
C
C     INITIALISATION OF PARAMETERS
C
C*****************************************************************
***
C              PART I
      WRITE(*,'(24(/))')
      PRINT*,'INTEGRATED SYSTEM OPTIMISATION AND PARAMETER'
      PRINT*,'ESTIMATION'
      WRITE(*,'(12(/))')
      PRINT*,'ENTER CHOISE FOR PROGRAM RUN:'
      PRINT*,'                          :<0> FOR STD ISOPE'
      PRINT*,'                          :<1> FOR    AISOPE'
      READ*,IOPT
      PRINT*,'ENTER NUMBER OF SET POINTS'
      READ*,NC
      PRINT*,'ENTER INITIAL VALUES OF SET POINTS'
      READ*,(XIP(I),I=1,NC)
      PRINT*,'ENTER GAIN'
      READ*,GK
      PRINT*,'ENTER CONVERGENCE CRITEREA'
      READ*,S
      PRINT*,'ENTER R1 (PENALTY FACTOR)'
      PRINT*,'ENTER <0> (ZERO) FOR MSTP, AND ANYTHING ELSE FOR AISOPE'
      READ*,R1
C
   5  NP=3
      LIW=10
      LW=100
      IA=7
      IAA=7
      PFI=0.0
C
C
```

```
C*************************************************************************
*
C
C                    PART II
C
      OPEN(40,FILE='GRPH.DAT',STATUS='NEW')
      OPEN(20,FILE='SETPT.DAT',STATUS='NEW')
      OPEN(21,FILE='PARA.DAT',STATUS='NEW')
      OPEN(41,FILE='REALFUNC.DAT',STATUS='NEW')
      OPEN(15,FILE='MODIF.DAT',STATUS='NEW')
      OPEN(14,FILE='DEDV.DAT',STATUS='NEW')
C
      DO 900 I=1,NC
      XVP(I)=XIP(I)
      XUOP(I)=XIP(I)
  900 CONTINUE
C
      ITERAT=1
C
C     PARAMETER ESTIMATION PROBLEM
C
  850 DO 800 I=1,NC
        XVP(I)=XVP(I)+GK*(XUOP(I)-XVP(I))
  800 CONTINUE
C
C     EVALUATE  PLANT OUTPUTS & EQUATE TO MODEL OUTPUTS
      CALL PLANT(NP,IA,IAA,XVP,YREAL)
      DO 750 I=1,NP
      YMDL(I)=YREAL(I)
  750 CONTINUE
C
      PE(1)=YREAL(1)-XVP(1)+XVP(2)-2.0*YREAL(2)
      PE(2)=YREAL(2)-XVP(3)+XVP(4)-YREAL(1)+YREAL(3)*3
      PE(3)=YREAL(3)-2.0*XVP(4)+XVP(5)+YREAL(1)-YREAL(2)
C
C
      CALL FUNCT(XVP,YREAL,F)
      WRITE(41,*)FA
C
C
      IF(IOPT.EQ.0) GO TO 400
C
C     OBTAIN REAL PROCESS MEASUREMENTS
C     DIFFERENCE BETWEEN MODEL AND PLANT OUTPUT DERIVATIVES
C
      CALL PLANT(NP,IA,IAA,XVP,YR)
      CALL MODEL(NP,IA,IAA,XVP,YMODEL,PE)
      DO 700 I=1,NC
      EINC(I)=0.01*XVP(I)
      IF(EINC(I).LE.1.0D-08) EINC(I)=0.001
      XVP(I)=XVP(I)+EINC(I)
      CALL MODEL(NP,IA,IAA,XVP,YMDL,PE)
      CALL PLANT(NP,IA,IAA,XVP,YREAL)
```

```
      DO 10 J=1,NP
      DMDY(I,J)=(YMDL(J)-YMODEL(J))/EINC(I)
      DRDY(I,J)=(YR(J)-YREAL(J))/EINC(I)
      DEDV(I,J)=(YMDL(J)-YREAL(J))/EINC(I)
   10 CONTINUE
      XVP(I)=XVP(I)-EINC(I)
  700 CONTINUE
C
      WRITE(14,*)((DEDV(I,J),J=1,3),I=1,5)
C
      CALL OPFCT(NP,IA,IAA,XIP,FA1,PE)
      CALL MODEL(NP,IA,IAA,XVP,YRLC,PE)
      DO 650 I=1,NP
      EINC(I)=0.01*PE(I)
      IF(EINC(I).LE.1.0D-08)EINC(I)=0.0001
      PE(I)=PE(I)+EINC(I)
      NNP = NP
      CALL MODEL(NNP,IA,IAA,XVP,YMDL,PE)
      DO 20 J=1,NP
      DYDA(I,J)=(YMDL(J)-YRLC(J))/EINC(I)
   20 CONTINUE
      NNP = NP
      CALL OPFCT(NNP,IA,IAA,XIP,FA2,PE)
      DFDA(I,1)=(FA2-FA1)/EINC(I)
      PE(I)=PE(I)-EINC(I)
  650 CONTINUE
C
C     COMPUTATION OF MODIFIERS (COOR) BY MATRIX OPERATIONS
C
      IFAIL=1
      CALL F01CBF(UNIMAT,3,3,IFAIL)
      IF(IFAIL.EQ.0)GO TO 600
      WRITE(1,30)IFAIL
   30 FORMAT('IDENTITY MATRIX NOT FOUND,IFAIL=',I3)
      STOP
  600 IFAIL=1
      CALL F04AEF(DYDA,3,UNIMAT,3,3,3,DADY,3,WKSPCE,AA,8,BB,8,IFAIL)
      IF(IFAIL.EQ.0) GO TO 550
      WRITE(1,40)IFAIL
   40 FORMAT('ERROR IN F04AEF,IFAIL=',I2)
       STOP
  550 N=1
      CALL F01CKF(DFDY,DADY,DFDA,NP,N,NP,Z,N,N,IFAIL)
      CALL F01CKF(DFDV,DEDV,DFDY,NC,N,NP,Z,N,N,IFAIL)
      DO 500 I=1,NC
      COOR(I)=DFDV(I,1)
  500 CONTINUE
C
      WRITE(15,*)(COOR(I),I=1,5)
C
  400   DO 450 I=1,NC
      X(I)=XIP(I)
  450 CONTINUE
C
```

```
C     SOLVE OPTIMISATION PROBLEM
C
      CALL OPTIM(X)
 350  IF(ITERAT.EQ.500) THEN
       GO TO 390
      ELSE
       CONTINUE
      ENDIF
C
      DSQRTC=0.0
      DO 300 I=1,NC
      DSQRTC=DSQRTC+(X(I)-XIP(I))**2
  300 CONTINUE
      DSQRTC=DSQRT(DSQRTC/5.0)
C
      IF(DSQRTC.GT.S)GO TO 250
C
 390  CALL RESULT(X,XVP,ITERAT,F,YREAL,YMDL,PE,COOR,CTR,DSQRTC)
C
C CLOSE FILES AND STOP
C
      CLOSE(40,STATUS='KEEP')
      CLOSE(20,STATUS='KEEP')
      CLOSE(41,STATUS='KEEP')
      CLOSE(22,STATUS='KEEP')
      CLOSE(15,STATUS='KEEP')
      CLOSE(14,STATUS='KEEP')

      STOP
C
C     ARGUMENT TRANSFER
C
 250  DO 200 I=1,NC
      XIP(I)=X(I)
      XUOP(I)=X(I)
  200 CONTINUE
C
      CALL RESULT(X,XVP,ITERAT,F,YREAL,YMDL,PE,COOR,CTR,DSQRTC)
C
      ITERAT=ITERAT+1
C
      GO TO 850
C
      END
C
C ***********************************************************************
C
C SUBROUTINE PLANT : SIMULATES STEADY STATE RESPONSE OF A SYSTEM
C
C                    INPUTS CONTROL SET-POINTS (XVP)
C                    OUTPUTS SYSTEM OUTPUTS (YREAL)
C         SUBROUTINE USES A NAG ROUTINE F04ATF FOR SOLVING THE SET
C                    OF SIMULTANEOUS EQUATIONS
C
```

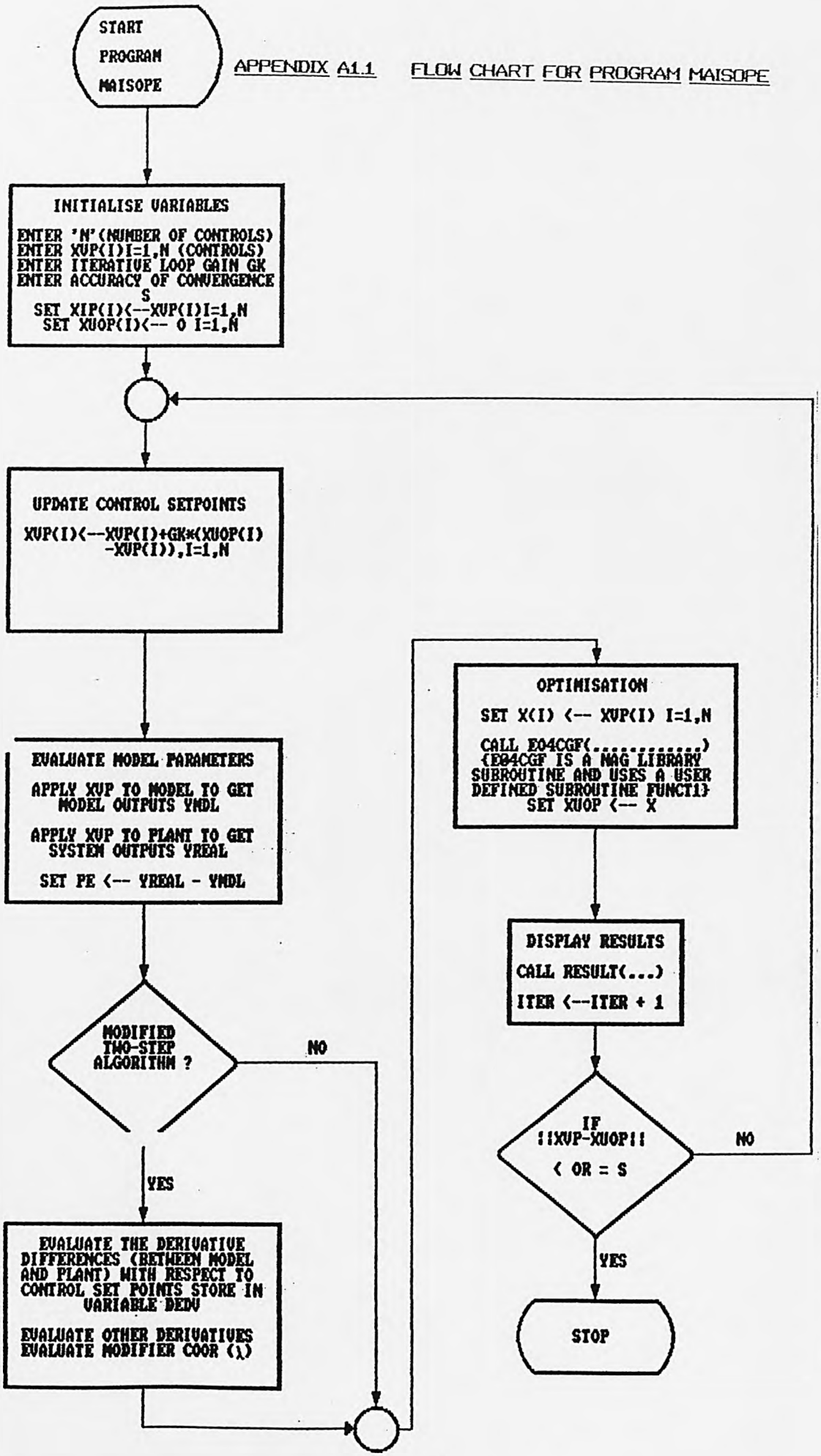
```
      SUBROUTINE PLANT(NP,IA,IAA,XVP,YREAL)
C     USING F04ATF TO SOLVE LINEAR EQUATIONS
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 A(7,7),B(7),CD(7),AA(7,7),WKSPC1(10),WKSPC2(10),XVP(5),
     &       YREAL(3)
C
C
      A(1,1)=1.0
      A(1,2)=-(2.0+0.15*XVP(1))
      A(1,3)=0.0
      A(2,1)=-1.2
      A(2,2)=1.0
      A(2,3)=3.0
      A(3,1)=1.0
      A(3,2)=-1.0
      A(3,3)=1.0
C
      B(1)=1.3*XVP(1)-1.0*XVP(2)
      B(2)=1.0*XVP(3)-1.0*XVP(4)+0.1*XVP(4)**2
      B(3)=2.0*XVP(4)-1.25*XVP(5)+0.25*XVP(4)*XVP(5)
C
      IFAIL=1
      CALL F04ATF(A,IA,B,NP,CD,AA,IAA,WKSPC1,WKSPC2,IFAIL)
      IF (IFAIL.EQ.0) GO TO 20
      WRITE(1,99)IFAIL
   99 FORMAT('ERROR IN F04ATF (PLANT),IFAIL=',I2)
C      STOP
   20 DO 30 I=1,NP
      YREAL(I)=CD(I)
   30 CONTINUE
C
      RETURN
      END
C
C ***********************************************************************
      SUBROUTINE PFORM(PFI,F)
      REAL*8 PFI,TOC,F
      COMMON /USED3/TOC
      PFI=F+TOC
      RETURN
      END
C
C ***********************************************************************
C
C SUBROUTINE MODEL : SIMULATES STEADY STATE RESPONSE OF A SYSTEM
C
C                    INPUTS CONTROL SET-POINTS (XVP)
C                    OUTPUTS MODEL  OUTPUTS (YREAL)
C         SUBROUTINE USES A NAG ROUTINE F04ATF FOR SOLVING THE SET
C                    OF SIMULTANEOUS EQUATIONS
C
      SUBROUTINE MODEL(NP,IA,IAA,XVP,YMDL,PE)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 XVP(5),YREAL(3),YMDL(3),PE(3)
```

```
      REAL*8 AM(7,7), BM(7),CM(7),AAM(7,7),WKSPC3(10),WKSPC4(10)
C
      AM(1,1)=1.0
      AM(1,2)=-2.0
      AM(1,3)=0.0
      AM(2,1)=-1.0
      AM(2,2)=1.0
      AM(2,3)=3.0
      AM(3,1)=1.0
      AM(3,2)=-1.0
      AM(3,3)=1.0
C
      BM(1)=XVP(1)-XVP(2)+PE(1)
      BM(2)=XVP(3)-XVP(4)+PE(2)
      BM(3)=2.0*XVP(4)-XVP(5)+PE(3)
C
C
      IFAIL=1
      CALL F04ATF(AM,IA,BM,NP,CM,AAM,IAA,WKSPC3,WKSPC4,IFAIL)
      IF(IFAIL.EQ.0) GO TO 20
      WRITE(1,99)IFAIL
   99 FORMAT('ERROR IN F04ATF(MODEL),IFAIL=',I2)
C      STOP
   20 DO 30 I=1,NP
      YMDL(I)=CM(I)
   30 CONTINUE
      RETURN
      END
C
C ***********************************************************************
C
C SUBROUTINE OPFCT  EVALUATES THE VALUE OF THE INDEX PERFROMANCE
C                   USED FOR EVALUATING THE DERIVATIVES OF THE INDEX
C                   WITH RESPECT TO THE PARAMETERS
C
      SUBROUTINE OPFCT(NP,IA,IAA,XIP,FA,PE)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 XIP(5),PE(3),YREAL(3),YMDL(3),FA
      CALL MODEL(NP,IA,IAA,XIP,YMDL,PE)
C     EXPRESSION OF PERFORMANCE INDEX
      FA=(YMDL(1)-1.0)**2+XIP(1)**2+XIP(2)**2+2.0*(YMDL(2)-2.0)**2+
     &(YMDL(3)-3.0)**2+XIP(3)**2+XIP(4)**2+XIP(5)**2
C
      RETURN
      END
C
C***********************************************************************
C
C SUBROUTINE RESULT DISPLAYS ALL THE RELEVANT DATA AT THE END OF
C                   EACH ITERATION.
C                   THE VALUE OF THE PERFORMANCE INDEX
C                   THE CURRENT SET-POINT VALUES
C                   THE CURRENT MODEL AND SYSTEM OUTPUTS
C                   THE CURRENT ITERATION NUMBER
```

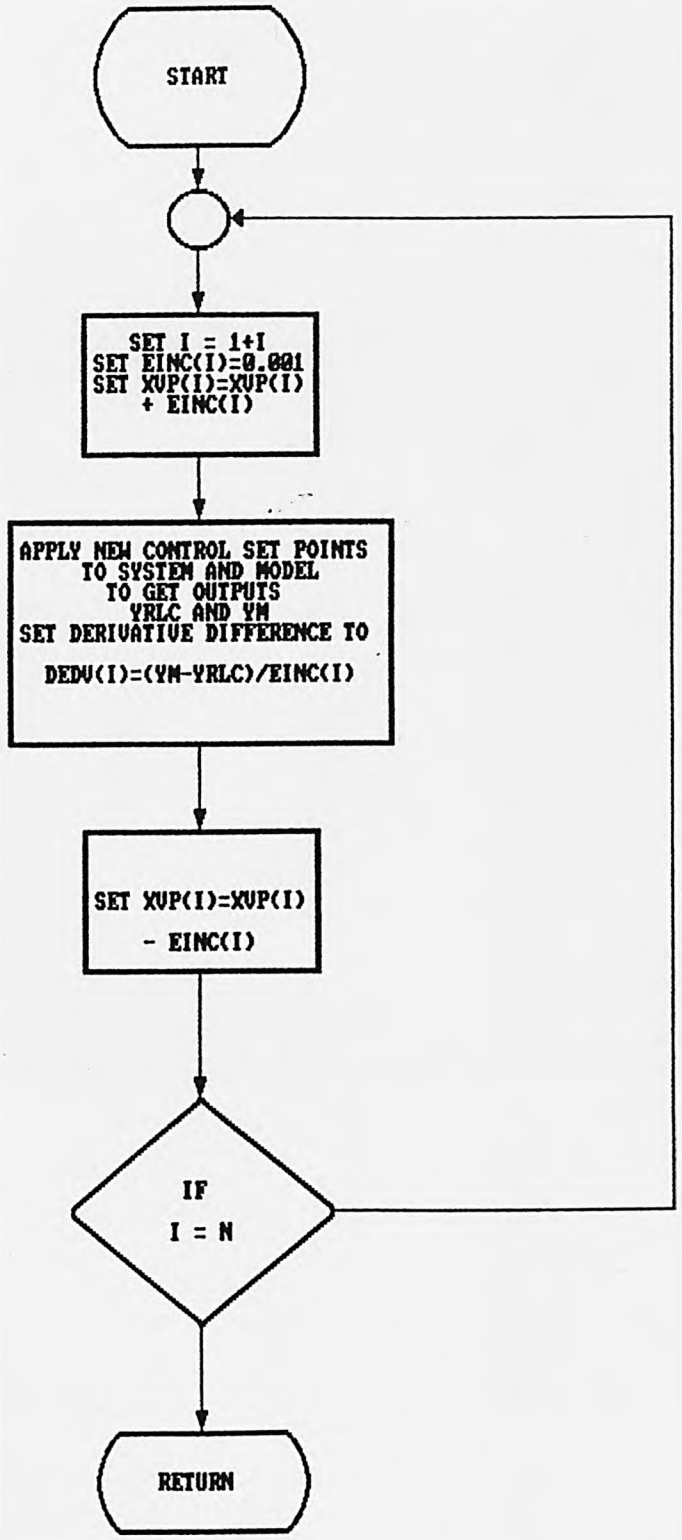
```
C                   THE VALUE OF THE MODIFIER
C
      SUBROUTINE RESULT(X,XVP,ITERAT,F,YREAL,YMDL,PE,COOR,DSQRTC)
C
      DOUBLE PRECISION X(5),YREAL(3),PE(3),YMDL(3),COOR(5),XVP(5),
     *                 F,DSQRTC
C
      IF (ITERAT.EQ.1) THEN
       PRINT*,'INTEGRATED SYSTEM OPTIMISATION AND PARAMETER ESTIMATION'
      ELSE
       CONTINUE
      ENDIF
      WRITE(*,'(2(/))')
      WRITE(*,11)ITERAT
      WRITE(*,1)F
      WRITE(*,2)(X(I),I=1,5)
      WRITE(*,3)(XVP(I),I=1,5)
      WRITE(*,4)(YREAL(I),I=1,3)
      WRITE(*,5)(YMDL(I),I=1,3)
      WRITE(*,6)(PE(I),I=1,3)
      WRITE(*,7)(COOR(I),I=1,5)
      WRITE(*,9)DSQRTC
      WRITE(*,10)
C
  1   FORMAT('FUNCTION VALUE=    ',F10.5)
  2   FORMAT('X      = ',5(F10.5,2X))
  3   FORMAT('XVP    = ',5(F10.5,2X))
  4   FORMAT('YREAL  = ',3(F10.5,2X))
  5   FORMAT('YMDL   = ',3(F10.5,2X))
  6   FORMAT('PE     = ',3(F10.5,2X))
  7   FORMAT('COOR   = ',5(F10.5,2X))
  9   FORMAT('DSQRTC= ',F10.5)
  10  FORMAT('*******************************************************')
  11  FORMAT('ITERATION   ',I10)
C
      RETURN
      END
C*********************************************************************
*
C
C SUBROUTINE OPTIM SOLVES THE OPTMISATION PROBLEM
C                 INPUTS THE STARTING POINT FOR OPTIMISATION
C                 THROUGH THE VARIABLE X AND OUTPUTS THE OPTIMUM
C                 VALUE THROUGH THE SAME VARIABLE X
C
C OPTIM USES THE NAG ROUTINE E04CGF FOR SOLVING THE OPTIMISATION
C PROBLEM.
C
      SUBROUTINE OPTIM(CS)
C
      REAL*8 CS(5),F, W(200)
      INTEGER IW(20),IFAIL
C
      LIW=10
```

```
      LW=100
      IFAIL=0
C
      CALL E04CGF(5,CS,F,IW,LIW,W,LW,IFAIL)
      IF(IFAIL.NE.0)THEN
        PRINT*,'IFAIL IN E04CGF=',IFAIL
        STOP
      END
C
      RETURN
      END
C
C **********************************************************************
***
C
C SUBROUTINE FUNCT1 IS A USER DEFINED SUBROUTINE.
C                   IT CONTAINS THE FUNCTION TO MINIMISED
C                   AND IS CALLED BY THE SUBROUTINE E04CGF
C
      SUBROUTINE FUNCT1(N,XC,FC)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C     ........ LOCAL ARGUMENTS ....
      DOUBLE PRECISION FC, XC(N)
C
      REAL*8 COOR(5),PE(3),YMDL(3),TOC,R1,XVP(5),TOCR
C
C     SCALAR IN COMMON
      COMMON /USED1/COOR,R1,XVP
      COMMON /USED3/TOC
      COMMON /USED2/PE
      TOC=0.0D+0
      TOCR=0.0D+0
      DO 900 I=1,N
      TOC=TOC+COOR(I)*XC(I)
      TOCR=TOCR+R1*((XC(I)-XVP(I))**2)
  900 CONTINUE
      CALL MODEL(3,7,7,XC,YMDL,PE)
C
      FC=(YMDL(1)-1.0)**2+XC(1)**2+XC(2)**2+2.0*(YMDL(2)-2.0)**2
     *+(YMDL(3)-3.0)**2+XC(3)**2+XC(4)**2+XC(5)**2-TOC+TOCR
C
      RETURN
      END
C
C
      SUBROUTINE FUNCT(XIP,YMDL,FA)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 XIP(5),YMDL(3),FA
C
      FA=(YMDL(1)-1.0)**2+XIP(1)**2+XIP(2)**2+2.0*(YMDL(2)-2.0)**2+
     &(YMDL(3)-3.0)**2+XIP(3)**2+XIP(4)**2+XIP(5)**2
C
      RETURN
      END
```

APPENDIX A1.1 FLOW CHART FOR PROGRAM MAISOPE



APPENDIX A1.1



```
C      PROGRAM ISOPEIA
C
C ****************************************************************
C *                                                              *
C * LINEARISES A PROCESS ABOUT A POINT TO DEVELOPE A MODELl OF   *
C * TYPE [Y]=[W][C]+[PE] WHERE Y=3X1                             *
C *                            W=3X5                             *
C *                            PE=3X1                            *
C *                            C=3X1                             *
C *                                                              *
C ****************************************************************
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION W(100),X(5),XVP(5),XIP(5),XUOP(5)
     &                ,YMDL(3),YREAL(3),YRLC(3),DEDV(5,3),COOR(5),
     &                 BB(8,8),AA(8,8),WKSPCE(10),Z(1),PE(3),WM(3,5)
      INTEGER IW(7),IOPT
      COMMON/USED1/COOR
      COMMON/MODE/WM
      COMMON/USED3/TOC
      COMMON/USED2/PE
C     INITIALISATION OF PARAMETERS
C
C*********************************************************************
***
C              PART I
      WRITE(*,'(24(/))')
      PRINT*,'INTEGRATED SYSTEM OPTIMISATION AND PARAMETER'
      PRINT*,'ESTIMATION'
      WRITE(*,'(12(/))')
      PRINT*,'ENTER NUMBER OF SET POINTS'
      READ*,NC
      PRINT*,'ENTER INITIAL VALUES OF SET POINTS'
      READ*,(XIP(I),I=1,NC)
      PRINT*,'ENTER GAIN'
      READ*,GK
      PRINT*,'ENTER CONVERGENCE CRITEREA'
      READ*,S
C
   5  NP=3
      LIW=10
      LW=100
      IA=7
      IAA=7
C
C
C*********************************************************************
*
C
C                   PART II
C
      OPEN(40,FILE='GRPH.DAT',STATUS='NEW')
      OPEN(20,FILE='SETPT.DAT',STATUS='NEW')
      OPEN(21,FILE='REALFUN.DAT',STATUS='NEW')
```

```
C
      DO 900 I=1,NC
      XVP(I)=XIP(I)
      XUOP(I)=XIP(I)
  900 CONTINUE
C
      ITERAT=1
C
C     MODEL IDENTIFICATION
C
  850 DO 800 I=1,NC
      XVP(I)=XVP(I)+GK*(XUOP(I)-XVP(I))
  800 CONTINUE
C
      CALL PEDERV(XVP,NC,NP,IA,IAA,WM,PE)
      CALL OPFCT(NP,IA,IAA,XVP,FA,PE,WM,NC)
      WRITE(21,*)FA
C
 400   DO 450 I=1,NC
      X(I)=XIP(I)
  450 CONTINUE
C     SOLVE OPTIMISATION PROBLEM
      IFAIL=1
      CALL E04CGF(NC,X,F,IW,LIW,W,LW,IFAIL)
      IF(IFAIL.EQ.0) GO TO 350
      IF(IFAIL.EQ.1)STOP
C
 350  IF(ITERAT.EQ.500) THEN
       GO TO 390
      ELSE
       CONTINUE
      ENDIF
      DO 300 I=1,NC
      IF(DABS(X(I)-XIP(I)).GT.S)GO TO 250
  300 CONTINUE
 390  CALL PLANT(NP,IA,IAA,X,YREAL)
      CALL MODEL1(NP,IA,IAA,X,YMDL,PE,WM,NC)
      PFI=F
      CALL RESULT(NC,NP,X,ITERAT,F,PFI,YREAL,YMDL,PE,COOR)
      WRITE(40,*)PFI
      WRITE(20,*)(XVP(I),I=1,NC)
C
      CLOSE(40,STATUS='KEEP')
      CLOSE(20,STATUS='KEEP')
C
      STOP
C
C     ARGUMENT TRANSFER
C
  250 DO 200 I=1,NC
      XIP(I)=X(I)
  200 CONTINUE
      DO 150 I=1,NC
      XUOP(I)=X(I)
```

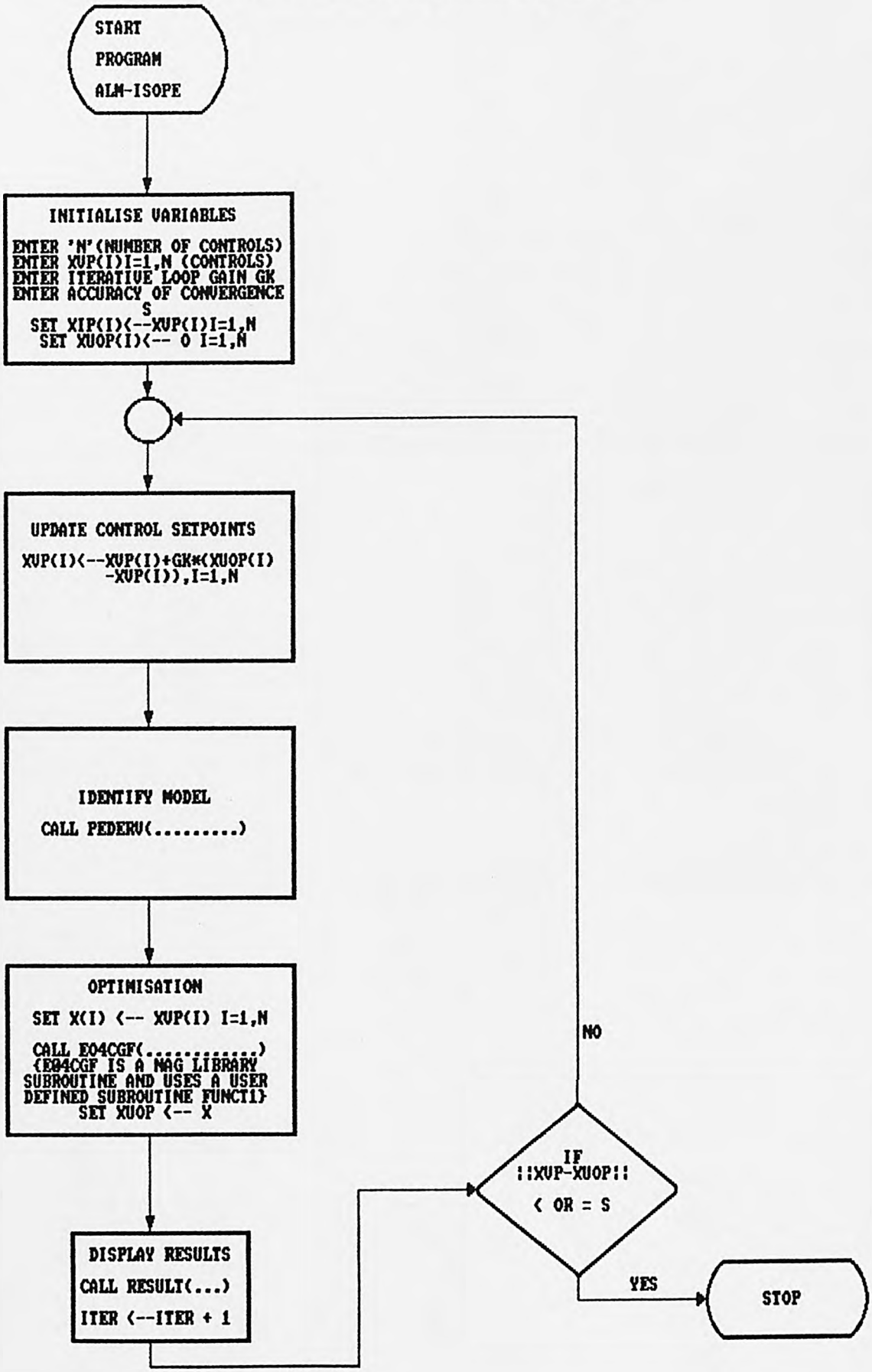
```
  150 CONTINUE
      CALL PLANT(NP,IA,IAA,XUOP,YREAL)
      CALL MODEL1(NP,IA,IAA,XUOP,YMDL,PE,WM,NC)
      CALL RESULT(NC,NP,XUOP,ITERAT,F,PFI,YREAL,YMDL,PE,COOR)
      WRITE(40,*)PFI
      WRITE(20,*)(XVP(I),I=1,NC)
C
      ITERAT=ITERAT+1
      GO TO 850
C
      STOP
      END
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE FUNCT1(NC,XC,FC)
      IMPLICIT REAL*8 (A-H,O-Z)
C     SCALAR ARGUMENTS
      REAL*8 FC,TOC
      REAL*8 WM(3,5)
      INTEGER NC,NP
C     ARRAY ARGUMENTS
      REAL*8 XC(5),COOR(5),PE(3),YMDL(3)
C
C     SCALAR IN COMMON
      COMMON /USED1/COOR
      COMMON /USED3/TOC
      COMMON /USED2/PE
      COMMON/MODE/WM
      TOC=0.0D+0
      DO 900 I=1,NC
      TOC=TOC+COOR(I)*XC(I)
  900 CONTINUE
      CALL MODEL1(3,7,7,XC,YMDL,PE,WM,NC)
C
      FC=(YMDL(1)-1.0)**2+XC(1)**2+XC(2)**2+2.0*(YMDL(2)-2.0)**2+
     &(YMDL(3)-3.0)**2+XC(3)**2+XC(4)**2+XC(5)**2-TOC
C     &-TOC
C      FC=YMDL(1)**2+1.7*(XC(1)-2.0)**2-2.8*XC(2)**2+7.3*(YMDL(3)-1.0)
C     &   **2-1.3*YMDL(2)**2+1.1*XC(3)**2+(XC(4)-1.0)**2+0.6*XC(5)
C     &   **2-TOC
      RETURN
      END
C
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE OPFCT(NP,IA,IAA,XIP,FA,PE,WM,NC)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 XIP(5),PE(3),YREAL(3),YMDL(3),FA
       REAL*8 WM(NP,NC)
      CALL MODEL1(NP,IA,IAA,XIP,YMDL,PE,WM,NC)
C     EXPRESSION OF PERFORMANCE INDEX
      FA=(YMDL(1)-1.0)**2+XIP(1)**2+XIP(2)**2+2.0*(YMDL(2)-2.0)**2
     &+(YMDL(3)-3.0)**2+XIP(3)**2+XIP(4)**2+XIP(5)**2
C      FA=YMDL(1)**2+1.7*(XIP(1)-2.0)**2-2.8*XIP(2)**2+7.3*(YMDL(3)-2.0
C     &   )**2-1.3*YMDL(2)**2+1.1*XIP(3)**2+(XIP(4)-1.0)**2+0.6*XIP(5)
C     &   **2
```

```
      RETURN
      END
C
C**********************************************************************
C
      SUBROUTINE RESULT(NC,NP,X,ITERAT,F,PFI,YREAL,YMDL,PE,COOR)
C
      DOUBLE PRECISION X(NC),YREAL(NP),PE(NP),YMDL(NP),COOR(NC)
     *,F,PFI
C
      IF (ITERAT.EQ.1) THEN
       PRINT*,'INTEGRATED SYSTEM OPTIMISATION AND PARAMETER ESTIMATION'
      ELSE
       CONTINUE
      ENDIF
      WRITE(*,'(2(/))')
      PRINT*,'ITERATION    ',ITERAT
      PRINT*,'AT POINT:'
      PRINT*,(X(I),I=1,NC)
      PRINT*,'MODEL1 OUTPUTS ARE:'
      PRINT*,(YMDL(I),I=1,NP)
      PRINT*,'PROCESS OUTPUTS ARE:'
      PRINT*,(YREAL(I),I=1,NP)
      PRINT*,'PARAMETERS ESTIMATES ARE:'
      PRINT*,(PE(I),I=1,NP)
      PRINT*,'MODIFIED PERFORMANCE INDEX:',F
      PRINT*,'REAL PERFORMANCE INDEX:',PFI
      PRINT*,'MODIFIERS ARE:'
      PRINT*,(COOR(I),I=1,NC)
      PRINT*,'************************************************************
'
C
      RETURN
      END
C**********************************************************************
*
C
C SUBROUTINE PEDERV IDENTIFIES THE LINEAR MODEL WHICH IS USED
C            SETS THE SYSTEM DERIVATIVES TO WM
C            AND MODEL IS IDENTIFIED AS Y=WM*XVP+PE
C
      SUBROUTINE PEDERV(XVP,NC,NP,IA,IAA,WM,PE)
      DOUBLE PRECISION XVP(NC),PE(NP),WM(NP,NC),PROD(3),SUM,YREAL(3),
     *       YRLD(3),EINC(10)
C
      DOUBLE PRECISION ALPHA,ALPHA2,STDN
      INTEGER NEXP(3)
      COMMON/NOIS/INCH,STDN,IFILT,ALPHA
      COMMON/FILTV/ALPHA2
      COMMON/FILT/IFYR
      COMMON/NEXPL/NEXP
C
      CALL PLANT(NP,IA,IAA,XVP,YREAL)
C
```

```
      DO 5 I=1,NC
        EINC(I)=0.01*XVP(I)
        IF(EINC(I).LE.1.0D-08)EINC(I)=0.01
        XVP(I)=XVP(I)+EINC(I)
        CALL PLANT(NP,IA,IAA,XVP,YRLD)
        DO 10 J=1,NP
         WM(J,I)=(YRLD(J)-YREAL(J))/EINC(I)
 10     CONTINUE
        XVP(I)=XVP(I)-EINC(I)
 5    CONTINUE
C
      DO 20 I=1,NP
        SUM=0.0
        DO 15 J=1,NC
         SUM=SUM+XVP(J)*WM(I,J)
 15     CONTINUE
        PROD(I)=SUM
 20   CONTINUE
C
      DO 25 I=1,NP
       PE(I)=YREAL(I)-PROD(I)
 25   CONTINUE
C
      RETURN
      END
C
C*********************************************************************
C
      SUBROUTINE MODEL1(NP,IA,IAA,XVP,YMDL,PE,WM,NC)
      REAL*8 XVP(NC),PE(NP),YMDL(NP),WM(NP,NC)
       DOUBLE PRECISION SUM,PROD(3)
C
      DO 10 I=1,NP
       SUM=0.0
       DO 5 J=1,NC
        SUM=SUM+XVP(J)*WM(I,J)
 5     CONTINUE
       PROD(I)=SUM
 10   CONTINUE
C
      DO 15 I=1,NP
       YMDL(I)=PROD(I)+PE(I)
 15   CONTINUE
C
      RETURN
      END
C
C *********************************************************************
C
C SUBROUTINE PLANT : SIMULATES STEADY STATE RESPONSE OF A SYSTEM
C
C                    INPUTS CONTROL SET-POINTS (XVP)
C                    OUTPUTS SYSTEM OUTPUTS (YREAL)
C         SUBROUTINE USES A NAG ROUTINE F04ATF FOR SOLVING THE SET
```

```
C                    OF SIMULTANEOUS EQUATIONS
C
      SUBROUTINE PLANT(NP,IA,IAA,XVP,YREAL)
C     USING F04ATF TO SOLVE LINEAR EQUATIONS
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 A(7,7),B(7),CD(7),AA(7,7),WKSPC1(10),WKSPC2(10),XVP(5),
     &       YREAL(3)
C
C
      A(1,1)=1.0
      A(1,2)=-(2.0+0.15*XVP(1))
      A(1,3)=0.0
      A(2,1)=-1.2
      A(2,2)=1.0
      A(2,3)=3.0
      A(3,1)=1.0
      A(3,2)=-1.0
      A(3,3)=1.0
C
      B(1)=1.3*XVP(1)-1.0*XVP(2)
      B(2)=1.0*XVP(3)-1.0*XVP(4)+0.1*XVP(4)**2
      B(3)=2.0*XVP(4)-1.25*XVP(5)+0.25*XVP(4)*XVP(5)
C
      IFAIL=1
      CALL F04ATF(A,IA,B,NP,CD,AA,IAA,WKSPC1,WKSPC2,IFAIL)
      IF (IFAIL.EQ.0) GO TO 20
      WRITE(1,99)IFAIL
   99 FORMAT('ERROR IN F04ATF (PLANT),IFAIL=',I2)
C      STOP
   20 DO 30 I=1,NP
      YREAL(I)=CD(I)
   30 CONTINUE
C
      RETURN
      END
C
```

APPENDIX A2.2 FLOW CHART FOR PROGRAM ALM-ISOPE



```
C ***********************************************************************
C *                                                                     *
C * PROGRAM FOR NEWTON-LIKE ALGORITHM FOR SOLVING THE DECENTRALISED     *
C * OPTIMISING CONTROL PROBLEM. SEE FLOW CHART FOR WORKING OF THE       *
C * PROGRAM                                                             *
C *                                                                     *
C ***********************************************************************
C
      PROGRAM SL-NEWT
C
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DOUBLE PRECISION EF(4), CX(7), UX(4), SK1, SK2, SK3,
     * H, G, CTR(4),EFC(7),SKC,SKG,DMDY(7,4),DYDW(4,4)
      DOUBLE PRECISION CV(7), CS(7), UW(4), U(4), T(4), PLAM(7),
     * P(4), AL(4), YREAL(4), YMDL(4),ERHOCK(11,11),EPS(11),R1,R2,
     * ERHOC(11,11),RLAM(3),DSQRTU,DSQRTC,DSQRTP,DSQRT,EP,UR1(4),
     * RLAMM(4),DRDY(7,4),DABS,BETA(3),UREAL(4),FCO,FC,CVP(7),UP(4),
     * TOTC,TOTP,UOLD(4),UXOLD(4),EPC
      INTEGER LN, IA, IAA, M ,IPROJE(4)
C
      COMMON/CONTR/CS, U, CTR
      COMMON/VARB/UW, R1, R2
      COMMON/SUBN/K
      COMMON/PRMT1/PLAM, T, P, AL
      COMMON/ER/ERHOCK
      COMMON/CVB/CV
      COMMON/RLAMMM/RLAMM,M2
      COMMON/DED/DRDY
      COMMON/NONLIN/BETA
      COMMON/PRO/CVP,UP,IPRO
C
C SET DERIVATIVES
C
      DO 1 I=1,4
         DO 1 J=1,4
            DYDW(I,J)=0.0
 1    CONTINUE
      DO 3 I=1,7
         DO 3 J=1,4
            DMDY(I,J)=0.0
 3    CONTINUE
      DYDW(1,1)=2.0
      DYDW(2,2)=1.0
      DYDW(2,3)=-3.0
      DYDW(3,2)=-1.0
      DYDW(3,3)=1.0
      DYDW(4,4)=-4.0
C
      DMDY(1,1)=1.0
      DMDY(2,1)=-1.0
      DMDY(3,2)=1.0
      DMDY(4,2)=-1.0
      DMDY(4,3)=2.0
      DMDY(5,3)=-1.0
```

```
      DMDY(6,4)=1.0
      DMDY(7,4)=2.0
C
         OPEN(11,FILE='FUNC.DAT',STATUS='UNKNOWN')
C
      DO 10 I=1,7
      PRINT*,'ENTER CX(',I,')'
      READ*,CX(I)
      CV(I) = CX(I)
      CS(I) = CX(I)
   10 CONTINUE
C
      DO 20 I=1,4
      PRINT*,'ENTER UX(',I,')'
      READ*,UX(I)
      U(I) = UX(I)
      UW(I) = UX(I)
      T(I)=0.0
      UOLD(I)=UX(I)
      UXOLD(I)=UX(I)
   20 CONTINUE
      PRINT*,'ENTER P(I),I=1,4'
      READ*,(P(I),I=1,4)
C
      PRINT*,'ENTER SK1,SK2,SK3,SK4'
      READ*,SK1,SK2,SK3,SK4
      PRINT*,'ENTER SKC,SKG,EPS,EPC'
      READ*,SKC,SKG,EP,EPC
      PRINT*,'ENTER G,H'
      READ*,G,H
C     H = 5.0D-4
      PRINT*,'ENTER R1,R2'
      READ*,R1,R2
      PRINT*,'ENTER NONLIN PARAMETERS I=1,3'
      READ*,(BETA(I),I=1,3)
      LN = 4
      IA = 7
      IAA = 7
      M2 = 1
C
      L = 0
C
      WRITE (1, 55555)
55555 FORMAT ('AUGMENTED SINGLE LOOP (ISOPE), EXAMPLE 1')
      WRITE (1, 55554) SK1, SK3, SK4
55554 FORMAT (/'SK1=',1PD12.4,2X,'SK3=',1PD12.4,2X,'SK4=',1PD12.4)
      WRITE (1, 55559) H, G, R1, R2
55559 FORMAT('H AND G',1X,2(1X,1PD12.2),3X,'R1 AND R2',1X,
     * 2(1X,1PD12.2))
C
 1500 CALL COORD(LN,IA,IAA,CV,CS,U,UW,P,PLAM,T,AL,SK4,EPS,R1,R2,EF,EFC,
     &           EP,SK2,ISELEC,SK1,SK2,M2,DYDW,DMDY,UOLD)
      CALL PFORM(CV, UW, FC, AL)
      FCO=FC
```

```
C
 1000 DO 111 I=1,7
      CS(I) = CX(I)
  111 CONTINUE
      DO 222 I=1,4
      U(I) = UX(I)
  222 CONTINUE
C
      K = 1
      CALL OPTIM(CV)
      K = 2
      CALL OPTIM(CV)
      K = 3
      CALL OPTIM(CV)
C
      YMDL(1) = CS(1) - CS(2) + 2.0*U(1) + AL(1)
      YMDL(2) = CS(3) - CS(4) + U(2) - 3.0*U(3) + AL(2)
      YMDL(3) = CS(4)*2.0 - CS(5) - U(2) + U(3) + AL(3)
      YMDL(4) = CS(6) + CS(7)*2.5 - 4.0*U(4) + AL(4)

C     CALL MODEL(LN, IA, IAA, YMDL, CS, AL)
C     CALL PLANT(LN,IA,IAA,YMDL,CS)
C
      EF(1)=U(1)-YMDL(2)
      EF(2)=U(2)-YMDL(1)
      EF(3)=U(3)-YMDL(4)
      EF(4)=U(4)-YMDL(3)
C
      DSQRTC=0.0
      DO 1001 I=1,7
       EFC(I)=CV(I)-CS(I)
 1001 CONTINUE
C
      DSQRTC=0.0
      DO 9900 I=1,7
         DSQRTC=DSQRTC+(CV(I)-CS(I))**2
C        DSQRTC=DSQRTC+(CX(I)-CS(I))**2
 9900 CONTINUE
      TOTC=FLOAT(7)
      DSQRTC=DSQRT(DSQRTC/TOTC)
      PRINT*,'DSRTQC=',DSQRTC
C
      DSQRTP=0.0
      DO 9911 I=1,4
        DSQRTP=DSQRTP+EF(I)**2
 9911 CONTINUE
      DSQRTP=DSQRT(DSQRTP/FLOAT(4))
      PRINT*,'DSQRP=',DSQRTP
C
      IF(DSQRTC.GT.G)GOTO 3456
      IF(DSQRTP.GT.H)GOTO 3456
C
      RM2 = M2
      WRITE (1, 44441) M2
```

```
      WRITE (1, 66666) FC
      WRITE (1, 99996) (CS(I),I=1,7)
      WRITE (1, 66661) (CV(I),I=1,7)
      WRITE (1, 99993) (U(I),I=1,4)
      WRITE (1, 66662) (UW(I),I=1,4)
      WRITE (1, 99992) (EF(I),I=1,4)
      WRITE (1, 33332) (P(I),I=1,4)
      WRITE (1, 33331) (T(I),I=1,4)
      WRITE (1, 33330) (AL(I),I=1,4)
      WRITE (1, 22229) (PLAM(I),I=1,7)
      WRITE (1, 22244) (CTR(I),I=1,4)
      PRINT*,'PROJECTION FLAGS'
      PRINT*,(IPROJE(I),I=1,4)
      STOP
C
 3456 RM2 = M2
      WRITE (1, 22222) M2
      WRITE (1, 66666) FC
      WRITE (1, 99996) (CS(I),I=1,7)
      WRITE (1, 66661) (CV(I),I=1,7)
      WRITE (1, 99993) (U(I),I=1,4)
      WRITE (1, 66662) (UW(I),I=1,4)
      WRITE (1, 99992) (EF(I),I=1,4)
      WRITE (1, 33332) (P(I),I=1,4)
      WRITE (1, 33331) (T(1),I=1,4)
      WRITE (1, 33330) (AL(I),I=1,4)
      WRITE (1, 22229) (PLAM(I),I=1,7)
      WRITE (1, 22244) (CTR(I),I=1,4)
      PRINT*,'EPS (I),I=1,11) ARE'
      PRINT*,(EPS(I),I=1,11)
      PRINT*,'PROJECTION FLAGS'
      PRINT*,(IPROJE(I),I=1,4)
C
      DO 101 I=1,4
       U(I)=UX(I)+SK2*(U(I)-UX(I))
 101  CONTINUE
      DO 102 I=1,7
       CS(I)=CX(I)+SK3*(CS(I)-CX(I))
 102  CONTINUE
C
      CALL NEWTON(DRDY,CV,UW,DYDW,DMDY,ERHOCK,AL,EPS,CS,U,R1,R2,EF,EFC,
     &            EP,EPC)
C
       DO 999 I=1,7
            CV(I)=CV(I)+EPS(I)
 999   CONTINUE
       DO 998 I=1,4
            P(I)=P(I)+EPS(I+7)
 998   CONTINUE
C
C     CALL PROJEC(CV,UW,U,IPROJE)
C
 3310 DO 599 I=1,7
C     CS(I)=CV(I)
```

```
C     CX(I) = CV(I)
      CX(I) = CS(I)
C     CV(I)=CV(I)+SK2*(CS(I)-CV(I))
  599 CONTINUE
      DO 699 I=1,4
      UX(I)=U(I)
C     P(I)=P(I)+SK3*EF(I)
  699 CONTINUE
C
      CALL PFORM(CV, U, FC, AL)
      WRITE(11,*)FC
C
      M2 = M2 + 1
      GO TO 1500
C
99996 FORMAT ('C',4X,7(1X,F12.6))
12345 FORMAT (6(1X,F12.5))
22229 FORMAT ('PLAM',2X,7(1X,F12.6))
66661 FORMAT ('CV',3X,7(1X,F12.6))
22244 FORMAT ('CONSTRAINTS',1X,4(1X,F12.6))
66662 FORMAT ('UW',3X,4(1X,F12.6))
22222 FORMAT ('**** AFTER',1X,I3,1X,'ITERATIONS ****')
99992 FORMAT ('EF',4X,4(1X,F12.6))
44441 FORMAT (///'CONVERGENT AFTER',I3,2X,'ITERATION '//)
33332 FORMAT ('P',4X,4(1X,F12.6))
33331 FORMAT ('T',4X,4(1X,F12.6))
33330 FORMAT ('AL',3X,4(1X,F12.6))
66666 FORMAT ('PERFORMANCE INDEX',1X,F12.6)
99993 FORMAT ('U',4X,4(1X,F12.6))
      END
C
C*********************************************************************
C
C
      SUBROUTINE OPTIM(CV)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C     ........ LOCAL ARGUMENTS......
      DOUBLE PRECISION ETA, F , RHO, STEPMX, XTOL, A(5,5), G(5),
     * C(3), RLAM(3), W(255), X(5), XL(5), XU(5), CL(1), CU(1),
     * CTR(4),RLAMM(4)
      INTEGER I, IBOUND, IFAIL, IPRINT, LA, LCLU, LIW, LW, MAXCAL,
     * N, NX, M, IW(30), MEQ, MRNGE, MINEQ
      LOGICAL LAMSET
C     ........SCALAR IN COMMON....
      DOUBLE PRECISION CS(7), U(4)
      REAL*8 P(4),R1,R2,T(4),AL(4),PLAM(7),CV(7),UW(4),ERHOC(11,11),
     &       ERHOCK(11,11)
      INTEGER K
C
C     ......... SUBROUTINE REFERENCE.......
C
      EXTERNAL FUNCT2,CON2,E04WAY,E04VAZ
      COMMON/CONTR/CS, U, CTR
      COMMON/SUBN/K
```

```
      COMMON/PRMTl/PLAM,T,P,AL
      COMMON/VAR3/UW,R1,R2
      COMMON/ER/ERHOCK
      COMMON/RLAMMM/RLAMM,M2
      IPRINT = -1
      ETA = 0.00001
      XTOL = 1.0D-16
      STEPMX = 20.0
      LCLU = 1
      IBOUND = 0
      LAMSET = .FALSE.
      MEQ = 0
      MRNGE = 0
      GO TO (100,200,300),K
  100 N = 3
      MINEQ = 1
      X(1) = CS(1)
      X(2) = CS(2)
      X(3) = U(1)
      XL(1) = -1.0D+6
      XU(1) = 1.0D+6
      XL(2) = -1.0D+6
      XU(2) = 1.0D+6
      XL(3) = 0.0D+0
      XU(3) = 0.5
      IF(M2.EQ.1)THEN
        LAMSET=.FALSE.
      ELSE
        LAMSET=.TRUE.
        RLAM(1)=RLAMM(1)
      ENDIF
      GO TO 400
  200 N = 5
      MINEQ = 2
      X(1) = CS(3)
      X(2) = CS(4)
      X(3) = CS(5)
      X(4) = U(2)
      X(5) = U(3)
      DO 40 I=1,5
      XL(I) = -1.0D+6
      XU(I) = 1.0D+6
   40 CONTINUE
      IF(M2.EQ.1)THEN
        LAMSET=.FALSE.
      ELSE
        LAMSET=.TRUE.
        RLAM(1)=RLAMM(2)
        RLAM(2)=RLAMM(3)
      ENDIF
      GO TO 400
  300 N = 3
      MINEQ = 1
      X(1) = CS(6)
```

```
      X(2) = CS(7)
      X(3) = U(4)
      XL(1) = -1.0D+6
      XU(1) = 1.0D+6
      XL(2) = 0.0D+0
      XU(2) = 1.0D+0
      XL(3) = -1.0D+6
      XU(3) = 1.0D+6
      IF(M2.EQ.1)THEN
        LAMSET=.FALSE.
      ELSE
        LAMSET=.TRUE.
        RLAM(1)=RLAMM(4)
      ENDIF
  400 LA = 5
      LIW = 30
      LW = 255
      M = MEQ + MINEQ + MRNGE
      IFAIL = 0
      CALL E04ZAF(N,M,FUNCT2,CON2,X,F,G,C,A,LA,W,LW,IFAIL)
      NX =N+MINEQ+MRNGE
      MAXCAL = 250*NX
      RHO =1.0D+0
      IFAIL = 1
      CALL E04VAF(N,MEQ,MINEQ,MRNGE,M,E04WAY,IPRINT,E04VAZ,MAXCAL,
     * ETA,XTOL,STEPMX,CL,CU,LCLU,IBOUND,XL,XU,LAMSET,X,RHO,RLAM,F,
     * G,C,IW,LIW,W,LW,IFAIL)
      GO TO(500,600,700),K
  500 CS(1) = X(1)
      CS(2) = X(2)
      U(1) = X(3)
      CTR(1) = C(1)
      RLAMM(1)=RLAM(1)
      GO TO 800
  600 CS(3) = X(1)
      CS(4) = X(2)
      CS(5) = X(3)
      U(2) = X(4)
      U(3) = X(5)
      CTR(2) = C(1)
      CTR(3) = C(2)
      RLAMM(2)=RLAM(1)
      RLAMM(3)=RLAM(2)
      GO TO 800
  700 CS(6) = X(1)
      CS(7) = X(2)
      U(4) = X(3)
      CTR(4) = C(1)
      RLAMM(4)=RLAM(1)
  800 CONTINUE
C
C 800 IF (IFAIL.NE.2) GO TO 900
C     WRITE (1, 1800) K
C     STOP
```

```
C1800 FORMAT ('K=',I1,5X,'IFAIL=2')
C
  900 NERHO=N
      CALL ERHO(CIR,K,CS,U,4,7,PLAM,P,R1,R2,ERHOC,T,NERHO,RLAM)
C
      GO TO (110,210,310),K
C
  110 DO 111 I=1,NERHO
        DO 111 J=1,NERHO
         ERHOCK(I,J)=ERHOC(I,J)
  111 CONTINUE
      GO TO 4000
C
  210 KK=3
      DO 211 I=1,NERHO
        DO 211 J=1,NERHO
         ERHOCK(I+KK,J+KK)=ERHOC(I,J)
  211 CONTINUE
      GO TO 4000
C
  310 KK=8
      DO 311 I=1,NERHO
        DO 311 J=1,NERHO
         ERHOCK(I+KK,J+KK)=ERHOC(I,J)
  311 CONTINUE
      GO TO 4000
C
 4000 RETURN
      END
C
C*********************************************************************
C
C
      SUBROUTINE FUNCT2(IFLAG,N,XC,FC,GC)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C     ........ LOCAL ARGUMENTS ....
      DOUBLE PRECISION FC, GC(N), XC(N)
C     .......SCALAR IN COMMON .....
      DOUBLE PRECISION PLAM(7), T(4), AL(4), P(4), CV(7), UW(4),
     &                 R1,R2
      INTEGER K
      COMMON/SUBN/K
      COMMON/PRMT1/PLAM, T, P, AL
      COMMON/VARB/UW, R1, R2
      COMMON/CVB/CV
C
      GO TO (100,200,300),K
C
 100  FC = (XC(3)-1.0)**4 + ((XC(1)+XC(2)-2.0)**2)*5.0
     &    + P(1)*XC(3) - P(2)*(XC(1)-XC(2)+XC(3)*2.0+AL(1))
     &    - PLAM(1)*XC(1) - PLAM(2)*XC(2)
     &   + (R1*(XC(1)-CV(1))**2) +(R1*(XC(2)-CV(2))**2)
     &   +(R2*(XC(3)-UW(1))**2)
      IF (IFLAG.EQ.0)GO TO 400
```

```
      GC(1) = 10.0*(XC(1)+XC(2)-2.0) - P(2) - PLAM(1)
     &  + 2.0*R1*(XC(1)-CV(1))
      GC(2) =P(2) - PLAM(2)+ 2.0*R1*(XC(2)-CV(2))+
     &       10.0*(XC(1)+XC(2)-2.0)
      GC(3) = 4.0*(XC(3)-1.0)**3 + P(1) - 2.0*P(2)
     &  + 2.0*R2*(XC(3)-UW(1))
      GO TO 400
  200 FC = (XC(1)-2.0)**2*2.0 + XC(2)**2 + XC(3)**2*3.0
     *  + XC(4)**2*4.0 + XC(5)**2 + P(2)*XC(4) + P(3)*XC(5)
     *  - P(1)*(XC(1)-XC(2)+XC(4)-3.0*XC(5)+AL(2))
     *  - P(4)*(2.0*XC(2)-XC(3)-XC(4)+XC(5)+AL(3))
     *  - PLAM(3)*XC(1) - PLAM(4)*XC(2) - PLAM(5)*XC(3)
     *  +(R1*(XC(1)-CV(3))**2)+(R1*(XC(2)-CV(4))**2)
     *  +(R1*(XC(3)-CV(5))**2)+(R2*(XC(4)-UW(2))**2)
     1  +(R2*(XC(5)-UW(3))**2)
      IF (IFLAG.EQ.0) GO TO 400
      GC(1) = 4.0*(XC(1)-2.0) - P(1) -PLAM(3)
     1   + 2.0*R1*(XC(1)-CV(3))
      GC(2) = 2.0*XC(2) + P(1) - P(4)*2.0 - PLAM(4)
     1  + 2.0*R1*(XC(2)-CV(4))
      GC(3) = 6.0*XC(3) + P(4) - PLAM(5)
     1  + 2.0*R1*(XC(3)-CV(5))
      GC(4) = 8.0*XC(4) + P(2) + P(4) - P(1)
     1  + 2.0*R2*(XC(4)-UW(2))
      GC(5) = 2.0*XC(5) + P(3) + 3.0*P(1) - P(4)
     1  + 2.0*R2*(XC(5)-UW(3))
      GO TO 400
  300 FC = (XC(1)+1.0)**2 + (XC(3)-1.0)**2 +(XC(2)**2)*2.5
     * + P(4)*XC(3) - P(3)*(XC(1)+2.5*XC(2)-4.0*XC(3)+AL(4))
     * - PLAM(6)*XC(1) - PLAM(7)*XC(2)
     1   +( R1*(XC(1)-CV(6))**2) +( R1*(XC(2)-CV(7))**2)
     1   +( R2*(XC(3)-UW(4))**2)
      IF (IFLAG.EQ.0) GO TO 400
      GC(1) = 2.0*(XC(1)+1.0) - P(3) - PLAM(6)
     1  + 2.0*R1*(XC(1)-CV(6))
      GC(2) = 5.0*XC(2) - 2.5*P(3) - PLAM(7)
     1   + 2.0*R1*(XC(2)-CV(7))
      GC(3) = 2.0*(XC(3)-1.0) + P(4) + P(3)*4.0
     1  + 2.0*R2*(XC(3)-UW(4))
      GO TO 400
C
  400 RETURN
      END
C
C*********************************************************************
C
      SUBROUTINE CON2(IFLAG,N,M,XC,CC,A,LA)
C     ........ LOCAL ARGUMENTS ....
      DOUBLE PRECISION CC(M),XC(N),A(LA,M)
      INTEGER K
      COMMON/SUBN/K
C
      GO TO(100,200,300),K
C
```

```
C 100 CC(1) = 1.0 - (XC(1)**2+XC(2)**2)
  100 CC(1) = 1.5 - (XC(1)+XC(2))
      IF (IFLAG.EQ.0) GO TO 400
      A(1,1) = -1.0
      A(2,1) = -1.0
      A(3,1) = 0.0
      GO TO 400
  200 CC(1) = 1.0 - (0.5*XC(1)+XC(2)+2.0*XC(3))
C
C     CC(2) = 4.0 - (4.0*XC(1)**2+2.0*XC(1)*XC(4)+0.4*XC(4)
C    &        +XC(1)*XC(3)+0.5*XC(3)**2+XC(4)**2)
C
      CC(2) = 4.5 - (4.0*XC(1)+2.0*XC(4)+XC(3))
      IF (IFLAG.EQ.0) GO TO 400
      A(1,1) = - 0.5D+0
      A(2,1) = - 1.0D+0
      A(3,1) = - 2.0D+0
      A(4,1) = 0.0D+0
      A(5,1) = 0.0D+0
      A(1,2) = -4.0
      A(2,2) = 0.0
      A(3,2) = -1.0
      A(4,2) = -2.0
      A(5,2) = 0.0
      GO TO 400
  300 CC(1) = XC(1) + XC(3) + 1.0
      IF (IFLAG.EQ.0) GO TO 400
      A(1,1) = 1.0D+0
      A(2,1) = 0.0D+0
      A(3,1) = 1.0D+0
C
  400 RETURN
      END
C
C*********************************************************************
C
      SUBROUTINE AMONIT(N,M,X,F,C,NITER,NNF,GLNORM,COND,POSDEF,RHO,
     * RLAM)
C     ..... LOCAL ARGUMENTS ....
      DOUBLE PRECISION COND,F,GLNORM,RHO,CNORM,C(M),RLAM(M),X(N)
      LOGICAL POSDEF
C     .......SCALAR IN COMMON ....
      INTEGER K
      COMMON/SUBN/K
C     ....FUNCTIO REFERENCE ...
      DOUBLE PRECISION DSQRT
      IF (NITER.GE.0) GO TO 140
      CNORM = 0.0
      GO TO(100,200,300),K
  100 IF(C(1).LT.0.0) CNORM = CNORM + C(1)**2
      CNORM = DSQRT(CNORM)
      GO TO 400
  200 IF (C(1).LT.0.0) CNORM = CNORM + C(1)**2
      IF (C(2).LT.0.0) CNORM = CNORM + C(2)**2
```

```
      CNORM = DSQRT(CNORM)
      GO TO 400
  300 IF(C(1).LT.0.0) CNORM = CNORM + C(1)**2
      CNORM = DSQRT(CNORM)
  400 WRITE (1,69990) GLNORM,CNORM
      WRITE (1,69989) RHO
      WRITE (1,69988) (RLAM(I),I=1,M)
      RETURN
  140 IF (COND.EQ.0.0) RETURN
      IF (COND.LT.1.0D+6) GO TO 180
      WRITE (1, 69993)
  180 IF(.NOT.POSDEF) WRITE(1,69991)
      RETURN
69993 FORMAT ('CONDITION NUMBER OF PROJECTED HESSIAN EXCEED 1.0D+6')
69991 FORMAT ('THE PROJECTED HESSIAN IS NOT POSITIVE DEFINITE')
69990 FORMAT ('END OF THE CYCLE GLNORM AND CNORM ARE',(1PD12.4))
69989 FORMAT ('RHO = ',1PD12.4)
69988 FORMAT ('THE LAGRANGE MULTIPLIER ARE'/(1PD12.4))
      END
C
C*********************************************************************
C
      SUBROUTINE COORD(LN,IA,IAA,CV,CS,U,UW,P,PLAM,T,AL,SK4,EPS,R1,R2,
     &                 EF,EFC,EP,SK2,ISELEC,SK1,SK3,M2,DYDW,DMDY,UOLD)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C     ......... LOCAL ARGUMENT ....
      DOUBLE PRECISION DMDY(7,4), DYDW(4,4), EINC(7), PT(4), YR1(4),
     * YR2(4), DEDV(7,4), YREAL(4), AL(4), DRDY(7,4), Q(4), DQPT(4)
C     ........SCALAR IN COMMON .....
      DOUBLE PRECISION CV(7), UW(4), P(4), PLAM(7), T(4), CS(7), U(4),
     &                 ERHOCK(11,11),EPS(11),EF(4),EFC(7),R1,R2,EP,SK2
     &                 ,SK1,SK3,H(4,4),TEMP(4,4),TEMP2(4,7),TEMP3(4,7)
     &                 ,DMDYT(4,7),DRDYT(4,7),YMODEL(4),YMDL(4),UOLD(4)
     &                 ,DSTAR(4,7)
C
      COMMON/ER/ERHOCK
      COMMON/DED/DRDY
C
C     ESTIMATION UNIT
C
      CALL PLANT(LN, IA, IAA, YREAL, CV)
      UW(1)=YREAL(2)
      UW(2)=YREAL(1)
      UW(3)=YREAL(4)
      UW(4)=YREAL(3)
C
      AL(1) = YREAL(1) - CV(1) + CV(2) - 2.0*UW(1)
      AL(2) = YREAL(2) - CV(3) + CV(4) - UW(2) + 3.0*UW(3)
      AL(3) = YREAL(3) - 2.0*CV(4) + CV(5) + UW(2) - UW(3)
      AL(4) = YREAL(4) - CV(6) - 2.5*CV(7) + 4.0*UW(4)
C
C
C     DERIVATIVE OF OUTPUT W.R.T CV
      CALL PLANT(LN,IA,IAA,YR1,CV)
```

```
C
      CALL MODEL(LN,IA,IAA,YMODEL,CV,AL)
C
      DO 350 I=1,7
      EINC(I) = 0.01*CV(I)
      IF (EINC(I).EQ.0.0D+0) EINC(I)=0.001
      CV(I) = CV(I) + EINC(I)
      CALL PLANT(LN, IA, IAA, YR2, CV)
      CALL MODEL(LN,IA,IAA,YMDL,CV,AL)
      DO 355 J=1,4
      DRDY(I,J) = (YR2(J) - YR1(J))/EINC(I)
      DMDY(I,J) = (YMDL(J)-YMODEL(J))/EINC(I)
  355 CONTINUE
      CV(I) = CV(I) - EINC(I)
  350 CONTINUE
      DO 352 I=1,7
      DO 353 J=1,4
      DEDV(I,J) = - (SK4*DRDY(I,J)-DMDY(I,J))
  353 CONTINUE
  352 CONTINUE
C
      PT(1) = - P(2)
      PT(2) = - P(1)
      PT(3) = - P(4)
      PT(4) = - P(3)
C
      DO 545 I=1,7
      SUM = 0.0D+0
      DO 560 J=1,4
      SUM = SUM + DEDV(I,J)*PT(J)
  560 CONTINUE
      PLAM(I) = SUM
  545 CONTINUE
C
      RETURN
      END
C
C*********************************************************************
*
C
      SUBROUTINE PFORM(CS, U, FC, AL)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION CS(7), U(4), AL(4)
      FC = (U(1)-1.0)**4 + (CS(1)+CS(2)-2.0)**2*5.0
     *    + (CS(3)-2.0)**2*2.0 + CS(4)**2 + CS(5)**2*3.0
     *    + U(2)**2*4.0 + U(3)**2 + (CS(6)+1.0)**2
     *    + (U(4)-1.0)**2 + CS(7)**2*2.5
      RETURN
      END
C
C*********************************************************************
C
      SUBROUTINE MODEL(LN, IA, IAA, YMDL, CS, AL)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
```

```
      DOUBLE PRECISION YMDL(4), CS(7), AL(4),YREAL(4),U(4)
C
      CALL PLANT(LN,IA,IAA,YREAL,CS)
C
      U(1)=YREAL(2)
      U(2)=YREAL(1)
      U(3)=YREAL(4)
      U(4)=YREAL(3)
C
      YMDL(1) = CS(1) - CS(2) + 2.0*U(1) + AL(1)
      YMDL(2) = CS(3) - CS(4) + U(2) - 3.0*U(3) + AL(2)
      YMDL(3) = CS(4)*2.0 - CS(5) - U(2) + U(3) + AL(3)
      YMDL(4) = CS(6) + CS(7)*2.5 - 4.0*U(4) + AL(4)
C
      RETURN
      END
C
C*********************************************************************
C
      SUBROUTINE FUNCT1(IFLAG,N,XC,FC)
      REAL*8 XC(N),FC,CVP(7),UP(4)
      COMMON/PRO/CVP,UP,IPRO
C
      GO TO (700,800,900),IPRO
C
  700 FC=(CVP(1)-XC(1))**2+(CVP(2)-XC(2))**2
      GO TO 400
C
  800 FC=0.0
      FC=(CVP(3)-XC(1))**2+(CVP(4)-XC(2))**2+(CVP(5)-XC(3))**2
      GO TO 400
C
  900 FC=0.0
      FC=(CVP(6)-XC(1))*2+(CVP(7)-XC(2))**2
C
  400 RETURN
      END
C
C *******************************************************************
C
      SUBROUTINE CON1(IFLAG,N,M,XC,CC)
C
      REAL*8 XC(N),CC(M),CVP(7),UP(4)
      COMMON/PRO/CVP,UP,IPRO
C
      GO TO(800,900,1000)IPRO
C
  800 CC(1) = 1.0 - (XC(1)**2+XC(2)**2)
      GO TO 400
  900 CC(1) = 1.0 - (0.5*XC(1)+XC(2)+2.0*XC(3))
      CC(2) = 4.0 - (4.0*XC(1)**2+2.0*XC(1)*UP(2)+0.4*UP(2)
     &        +XC(1)*XC(3)+0.5*XC(3)**2+UP(2)**2)
      GO TO 400
 1000 CC(1) = XC(1) + UP(4) + 0.5
```

```
C
 400  RETURN
      END
C
C ***************************************************************
C
      SUBROUTINE PLANT(LN, IA, IAA, YREAL, CS)
      DOUBLE PRECISION A(7,7), B(7), CD(7), AA(7,7), WKSPC1(30),
     * WKSPC2(10), YREAL(4), CS(7),BETA(3)
      INTEGER LN, IA, IAA, IFAIL
      COMMON/NONLIN/BETA
      A(1,1) = 1.0D+0
      A(1,2) = (-2.0-BETA(1)*CS(1))
      A(1,3) = 0.0D+0
      A(1,4) = 0.0D+0
      A(2,1) = -1.2D+0
      A(2,2) = 1.0D+0
      A(2,3) = 0.0D+0
      A(2,4) = 3.0D+0
      A(3,1) = 1.0D+0
      A(3,2) = 0.0D+0
      A(3,3) = 1.0D+0
      A(3,4) = -1.0D+0
      A(4,1) = 0.0D+0
      A(4,2) = 0.0D+0
      A(4,3) = 4.2D+0
      A(4,4) = 1.0D+0
C
      B(1) = 1.3*CS(1) - CS(2)
      B(2) = CS(3) - CS(4) + CS(4)**2*BETA(2)
      B(3) = CS(4)*2.0 - CS(5)*1.25 + CS(4)*CS(5)*BETA(3) + 0.1
      B(4) = 0.8*CS(6) + CS(7)*2.5
C
C     B(1) = 1.3*CS(1) - CS(2)
C     B(2) = CS(3) - CS(4)*2.0
C     B(3) = CS(4)*2.0 - CS(5)*2.0
C     B(4) = 0.8*CS(6) + CS(7)*2.0
C
      IFAIL = 1
      CALL F04ATF(A, IA, B, LN, CD, AA, IAA, WKSPC1, WKSPC2, IFAIL)
      IF (IFAIL.EQ.0) GO TO 20
      WRITE (1, 90000) IFAIL
90000 FORMAT (/'ERROR IN F04ATF - IFAIL =',I2/)
   20 DO 30 I=1,LN
      YREAL(I) = CD(I)
   30 CONTINUE
C
      RETURN
      END
C
C ***************************************************************
*
C
      SUBROUTINE ERHO(CTR,K,CS,U,NU,NCS,PLAM,P,R1,R2,ERHOC,T,
```

```
     *                NERHO,RLAM)
C
      REAL*8 CTR(4),CS(7),U(4),PLAM(7),T(4),P(4),R1,R2,ERHOC(11,11),
     *       RLAM(3)
      INTEGER K,NERHO
C
C VARIABLES FOR THIS SUBROUTINE ONLY
C
      REAL*8 L(11,11),ERHOC2(11,11),G(1,10),TEMP(10,10),TEMP2(10,10),
     &       SUM,TEM,G2(2,10),TEMPI(10,10),TEMP2I(10,10),GT(10,1),
     &       L1(11,11),UNIT(11,11),WKSPCE(11)
C
C CHECK FOR SUBSYSTEM
C
      DO 1  I=1,11
       DO 1 J=1,11
        L(I,J)=0.0
        L1(I,J)=0.0
  1   CONTINUE
      DO 9000 I=1,10
            G(1,I)=0.0
            GT(I,1)=0.0
            G2(1,I)=0.0
            G2(2,I)=0.0
 9000  CONTINUE
C
      GO TO (10,20,30),K
C
C K = 1 SUBSYSTEM I
C
 10   L(1,1)=10.0
      L(1,2)=10.0
      L(1,3)=0.0
      L(2,1)=10.0
      L(2,2)=10.0
      L(2,3)=0.0
      L(3,1)=0.0
      L(3,2)=0.0
      L(3,3)=12.0*(U(1)-1.0)**2
C
      DO 101 I=1,NERHO
                L(I,I)=L(I,I)+R1*2.0
 101  CONTINUE
C
C INVERT L(N,N)
C
      NL=NERHO
      CALL INVERT(L,11,NL)
C
C     IF(RLAM(1).LE.0.0.OR.CTR(1).GT.0.0)THEN
      IF(RLAM(1).LE.0.0)THEN
       DO 102 I=1,NERHO
        DO 102 J=1,NERHO
         ERHOC(I,J)=L(I,J)
```

```
 102   CONTINUE
C
      ELSE
C
       G(1,1)=+1.0
       G(1,2)=+1.0
       G(1,3)=0.0
       DO 11 I=1,NERHO
          GT(I,1)=G(1,I)
 11    CONTINUE
       SUM=0.0
       DO 103 I=1,NERHO
        SUM=0.0
        DO 104 J=1,NERHO
         SUM=SUM+L(I,J)*GT(J,1)
 104    CONTINUE
        TEMP(I,1)=SUM
 103   CONTINUE
C
       SUM=0.0
       DO 105 I=1,NERHO
        SUM=SUM+G(1,I)*TEMP(I,1)
 105   CONTINUE
       TEM=SUM
       TEM=1.0/TEM
C
       DO 106 I=1,NERHO
         TEMP(I,1)=TEMP(I,1)*TEM
 106   CONTINUE
C
       DO 107 I =1,NERHO
        SUM=0.0
        DO 108 J =1,NERHO
         SUM=SUM+G(1,J)*L(J,I)
 108    CONTINUE
        TEMP2(1,I)=SUM
 107   CONTINUE
C
       DO 109 I=1,NERHO
        DO 109 J=1,NERHO
         ERHOC2(I,J)=TEMP(I,1)*TEMP2(1,J)
 109   CONTINUE
C
       DO 110 I=1,NERHO
        DO 110 J=1,NERHO
         ERHOC(I,J)=L(I,J)-ERHOC2(I,J)
 110   CONTINUE
C
      ENDIF
C
      GO TO 40
C
C K = 2 SUBSYSTEM II
C
```

```
  20  L(1,1)=4.0
      L(1,3)=0.0
      L(1,4)=0.0
      L(2,2)=2.0
      L(3,1)=0.0
      L(3,3)=6.0
      L(3,4)=0.0
      L(4,1)=0.0
      L(4,4)=8.0
      L(5,5)=2.0
C
      DO 201 I=1,NERHO
                L(I,I)=L(I,I)+R1*2.0
 201  CONTINUE
C
C INVERT L(N,N)
C
      NL=NERHO
      CALL INVERT(L,11,NL)
C
      IF(RLAM(1).LE.0.0.AND.RLAM(2).LE.0.0)THEN
C     IF((RLAM(1).LE.0.0.AND.RLAM(2).LE.0.0).OR.(CTR(2).GT.0.0
C    *   .AND.CTR(3).GT.0.0))THEN
       DO 202 I=1,NERHO
        DO 202 J=1,NERHO
         ERHOC(I,J)=L(I,J)
 202   CONTINUE
C
      ELSE
C
       IF(RLAM(1).LE.0.0.AND.RLAM(2).GT.0.0)THEN
C      IF((RLAM(1).LE.0.0.AND.RLAM(2).GT.0.0).OR.(CTR(2).GT.0.0
C    *    .AND.CTR(3).LE.0.0))THEN
           NGA=1
           G2(1,1) = +4.0
           G2(1,2) = 0.0
           G2(1,3) = +1.0
           G2(1,4) = +2.0
           G2(1,5) = 0.0
       ELSE
        IF(RLAM(1).GT.0.0.AND.RLAM(2).LE.0.0)THEN
C       IF((RLAM(1).GT.0.0.AND.RLAM(2).LE.0.0).OR.(CTR(2).LE.0.0
C    *     .AND.CTR(3).GT.0.0))THEN
C
           G2(1,1)=+0.5
           G2(1,2)=+1.0
           G2(1,3)=+2.0
           G2(1,4)=0.0
           G2(1,5)=0.0
           NGA=2
        ELSE
C
         IF(RLAM(1).GT.0.0.AND.RLAM(2).GT.0.0)THEN
           NGA=3
```

```
              G2(1,1)=+0.5
              G2(1,2)=+1.0
              G2(1,3)=+2.0
              G2(1,4)=0.0
              G2(1,5)=0.0
              G2(2,1) = +4.0
              G2(2,2) = 0.0
              G2(2,3) = +1
              G2(2,4) = +2.0
              G2(2,5) = 0.0
C
            ENDIF
           ENDIF
          ENDIF
C
C         NCTR=2
C
          IF(NGA.EQ.1)THEN
           NCTR=1
          ELSE
           IF(NGA.EQ.2)THEN
            NCTR=1
           ELSE
            NCTR=2
           ENDIF
          ENDIF
C
          SUM=0.0
          DO 203 I=1,NERHO
           SUM=0.0
           DO 204 J=1,NCTR
             SUM=0.0
             DO 205 KKK=1,NERHO
                SUM=SUM+L(I,KKK)*G2(J,KKK)
 205         CONTINUE
             TEMP(I,J)=SUM
 204       CONTINUE
 203      CONTINUE
C
C G2(2,5)*TEMP(5,2)=TEMPI(2,2)
C
         DO 206 I=1,NCTR
           SUM=0.0
           DO 207 J=1,NCTR
             SUM=0.0
             DO 208 KKK=1,NERHO
               SUM=SUM+G2(I,KKK)*TEMP(KKK,J)
  208        CONTINUE
             TEMPI(I,J)=SUM
  207      CONTINUE
  206    CONTINUE
C
C INVERT TEMPI
C
```

```
      IF(NCTR.EQ.1)THEN
       TEMPI(1,1)=1.0/TEMPI(1,1)
      ELSE
       CALL INVERT(TEMPI,10,2)
      ENDIF
C
C MULTIPLY TEMP(5,2)*TEMPI(2,2)=TEMP2I(5,2)
C
      DO 209 I=1,NERHO
        SUM=0.0
        DO 210 J=1,NCTR
          SUM=0.0
          DO 211 KKK=1,NCTR
            SUM=SUM+TEMP(I,KKK)*TEMPI(KKK,J)
  211     CONTINUE
          TEMP2I(I,J)=SUM
  210   CONTINUE
  209 CONTINUE
C
C MULTIPLY G2(2,5)*L(5,5)=TEMP2(2,5)
C
      DO 212 I=1,NCTR
        SUM=0.0
        DO 213 J=1,NERHO
          SUM=0.0
          DO 214 KKK=1,NERHO
            SUM=SUM+G2(I,KKK)*L(KKK,J)
  214     CONTINUE
          TEMP2(I,J)=SUM
  213   CONTINUE
  212 CONTINUE
C
C ERHOC2(5,5)=TEMP2I(5,2)*TEMP2(2,5)
C
      DO 215 I=1,NERHO
        SUM=0.0
        DO 216 J=1,NERHO
          SUM=0.0
          DO 217 KKK=1,NCTR
            SUM=SUM+TEMP2I(I,KKK)*TEMP2(KKK,J)
  217     CONTINUE
          ERHOC2(I,J)=SUM
  216   CONTINUE
  215 CONTINUE
C
C
      DO 218 I=1,NERHO
       DO 218 J=1,NERHO
        ERHOC(I,J)=L(I,J)-ERHOC2(I,J)
 218  CONTINUE
C
     ENDIF
C
     GO TO 40
```

```
C
C K = 3 SUBSYSTEM III
C
 30   L(1,1)=2.0
      L(2,2)=5.0
      L(3,3)=2.0
C
      DO 301 I=1,NERHO
                  L(I,I)=L(I,I)+R1*2.0
 301  CONTINUE
C
C INVERT L(N,N)
C
      NL = NERHO
      CALL INVERT(L,11,NL)
C
C     IF(RLAM(1).LE.0.0.AND.CTR(3).GT.0.0)THEN
      IF(RLAM(1).LE.0.0)THEN
       DO 302 I=1,NERHO
        DO 302 J=1,NERHO
         ERHOC(I,J)=L(I,J)
 302   CONTINUE
C
      ELSE
C
       G(1,1)=-1.0
       G(1,2)=0.0
       G(1,3)=-1.0
       DO 31 I=1,NERHO
          GT(I,1)=G(1,I)
 31    CONTINUE
       SUM=0.0
       DO 303 I=1,NERHO
        SUM=0.0
        DO 304 J=1,NERHO
         SUM=SUM+L(I,J)*GT(J,1)
 304    CONTINUE
        TEMP(I,1)=SUM
 303   CONTINUE
C
      SUM=0.0
      DO 305 I=1,NERHO
       SUM=SUM+G(1,I)*TEMP(I,1)
 305  CONTINUE
      TEM=SUM
      TEM=1.0/TEM
C
      DO 306 I=1,NERHO
        TEMP(I,1)=TEMP(I,1)*TEM
 306  CONTINUE
C
      DO 307 I =1,NERHO
       SUM=0.0
       DO 308 J =1,NERHO
```

```
          SUM=SUM+G(1,J)*L(J,I)
 308      CONTINUE
          TEMP2(1,I)=SUM
 307     CONTINUE
C
         DO 309 I=1,NERHO
          DO 309 J=1,NERHO
           ERHOC2(I,J)=TEMP(I,1)*TEMP2(1,J)
 309     CONTINUE
C
         DO 310 I=1,NERHO
          DO 310 J=1,NERHO
           ERHOC(I,J)=L(I,J)-ERHOC2(I,J)
 310     CONTINUE
C
       ENDIF
C
  40     RETURN
         END
C
C*********************************************************************
C
C
       SUBROUTINE NEWTON(DRDY,CV,UW,DYDW,DMDY,ERHOCK,AL,EPS,CS,U,R1,R2,
      &                  EF,EFC,EP,EPC)
C
       REAL*8 ZSTAR(11,7),DSTAR(4,7),DRDY(7,4),CV(7),UW(4),DMDY(7,4),
      *       DYDW(4,4),DSTART(7,4),H(4,4),DT(7,4),EPS(11),CS(7),
      *       CMDT(11,4),GDASH(11,4),SUM,TEMP1(11,4),TEMP2(11,11),U(4),
      *       ERHOCK(11,11),XDASH(11,11),CDASH(7,11),AL(4),TEMP3(11,11),
      *       TEMP4(11,11),TEMP5(11,11),HT(4,4),XZ(11),AA(20,20),UR1(4),
      *       BB(20,20),WKSPCE(20),R1,R2,KSTAR(4,7),EF(4),EFC(7),UR2(4),
      *       EINC(11),EP,EPC
C
       DO 1000 I=1,4
        DO 1000 J=1,4
         H(I,J)=0.0
 1000 CONTINUE
C
       H(1,2)=1.0
       H(2,1)=1.0
       H(3,4)=1.0
       H(4,3)=1.0
C
       CALL ZSTARI(DRDY,ZSTAR)
C
       DO  9999 I=1,7
                DO  9999 J=1,4
                         KSTAR(J,I)=DRDY(I,J)
 9999 CONTINUE
C
C
C TRANSPOSE DSTAR TO DSTART
C
```

```
      DO 1 I=1,7
       DO 1 J=1,4
        DSTART(I,J)=DMDY(I,J)-DRDY(I,J)
C       DSTART(I,J)=DSTAR(J,I)
        DSTAR(J,I)=DSTART(I,J)
  1   CONTINUE
C
C
C MULTIPLY DSTART*HT
C
      DO 2 I=1,4
         DO 2 J=1,4
           HT(J,I)=H(I,J)
   2  CONTINUE
      DO 3 I=1,7
          SUM=0.0
          DO 4 J=1,4
              SUM=0.0
              DO 5 KKK=1,4
                  SUM=SUM+DSTART(I,KKK)*HT(KKK,J)
  5           CONTINUE
              DT(I,J)=SUM
  4       CONTINUE
  3   CONTINUE
C
C PERFORM OPERATION CM (BY ADDING ROWS OF ZEROS
C
      KKK=1
      DO 6 I=1,11
          IF(I.EQ.3.OR.I.EQ.7.OR.I.EQ.8.OR.I.EQ.11)THEN
               DO 7 J=1,4
                   CMDT(I,J)=0.0
  7            CONTINUE
          ELSE
               DO 71 J=1,4
                   CMDT(I,J)=DT(KKK,J)
  71           CONTINUE
               KKK=KKK+1
          ENDIF
  6   CONTINUE
C
C
C EVALUATE GDASH......
C
      DO 8 I=1,11
          DO 8 J=1,4
              GDASH(I,J)=0.0
  8   CONTINUE
C
      URl(1)=U(1)-(CS(3)-CS(4)+U(2)-3.0*U(3))
      URl(2)=U(2)-(CS(1)-CS(2)+2.0*U(1))
      URl(3)=U(3)-(CS(6)+2.5*CS(7)-4.0*U(4))
      URl(4)=U(4)-(2.0*CS(4)-CS(5)-U(2)+U(3))
C
```

```
      KKK=0
      KKKKK=0
      DO 40 I=1,11
          IF(I.EQ.3.OR.I.EQ.7.OR.I.EQ.8.OR.I.EQ.11)THEN
               KKKKK=KKKKK+1
               EINC(I)=U(KKKKK)*0.001
               IF(EINC(I).LT.1.0D-10)EINC(I)=0.0001
               U(KKKKK)=U(KKKKK)+EINC(I)
               UR2(1)=U(1)-(CS(3)-CS(4)+U(2)-3.0*U(3))
               UR2(2)=U(2)-(CS(1)-CS(2)+2.0*U(1))
               UR2(3)=U(3)-(CS(6)+2.5*CS(7)-4.0*U(4))
               UR2(4)=U(4)-(2.0*CS(4)-CS(5)-U(2)+U(3))
               DO 41 J=1,4
                GDASH(I,J)=(UR2(J)-UR1(J))/EINC(I)
   41          CONTINUE
               U(KKKKK)=U(KKKKK)-EINC(I)
          ELSE
               KKK=KKK+1
               EINC(I)=CS(KKK)*0.0001
               IF(EINC(I).LT.1.0D-08)EINC(I)=0.001
               CS(KKK)=CS(KKK)+EINC(I)
               UR2(1)=U(1)-(CS(3)-CS(4)+U(2)-3.0*U(3))
               UR2(2)=U(2)-(CS(1)-CS(2)+2.0*U(1))
               UR2(3)=U(3)-(CS(6)+2.5*CS(7)-4.0*U(4))
               UR2(4)=U(4)-(2.0*CS(4)-CS(5)-U(2)+U(3))
               DO 42 J=1,4
                GDASH(I,J)=(UR2(J)-UR1(J))/EINC(I)
   42          CONTINUE
               CS(KKK)=CS(KKK)-EINC(I)
          ENDIF
   40 CONTINUE
C
C PERFORM -GDASH(I,J)-CMDT(I,J)=TEMP1(I,J)
C
      DO 9 I=1,11
          DO 9 J=1,4
             TEMP1(I,J)=-GDASH(I,J)-CMDT(I,J)
   9  CONTINUE
C
      DO 10 I=1,11
           DO 10 J=1,7
                IF(I.EQ.3.OR.I.EQ.7.OR.I.EQ.8.OR.I.EQ.11)THEN
                     ZSTAR(I,J)=R2*ZSTAR(I,J)*2.0
                ELSE
                     ZSTAR(I,J)=R1*ZSTAR(I,J)*2.0
                 ENDIF
  10  CONTINUE
C
C ASSEMBLE MATRIX TEMP2 = ZSTAR¦TEMP1
C
      DO 11 I=1,11
           DO 12 J=1,11
                IF(J.GE.8)THEN
                     TEMP2(I,J)=TEMP1(I,J-7)
```

```
                ELSE
                     TEMP2(I,J)=ZSTAR(I,J)
                ENDIF
   12       CONTINUE
   11 CONTINUE
C
C MULTIPLY ERHOCK*TEMP2 = XDASH
C
      DO 13 I=1,11
           SUM=0.0
           DO 14 J=1,11
                SUM=0.0
                DO 15 KKK=1,11
                     SUM=SUM+ERHOCK(I,KKK)*TEMP2(KKK,J)
  15            CONTINUE
                XDASH(I,J)=SUM
  14       CONTINUE
  13  CONTINUE
C
C SELECT CERTAIN ROWS OF XDASH TO GIVE CDASH
C
      KKK=1
      DO 16 I=1,11
           IF(I.NE.3.AND.I.NE.7.AND.I.NE.8.AND.I.NE.11)THEN
                DO 17 J=1,11
                     CDASH(KKK,J)=XDASH(I,J)
   17           CONTINUE
                KKK=KKK+1
           ENDIF
   16 CONTINUE
C
C
C                          -         -
C                         | I       0 |
C ASSEMBLE MATRIX TEMP3 = |           |
C                         | HDSTAR  0 |
C                          -         -
C
      DO 18 I=1,7
           DO 18 J=1,7
                IF(I.EQ.J)THEN
                     TEMP3(I,J)=1.0
                ELSE
                     TEMP3(I,J)=0.0
                ENDIF
  18  CONTINUE
C
      DO 19 I=1,7
           DO 19 J=1,4
                TEMP3(I,J+7)=0.0
  19  CONTINUE
C
      DO 20 I=1,4
           DO 20 J=1,4
```

```
                TEMP3(I+7,J+7)=0.0
  20  CONTINUE
C
      DO 21 I=1,4
           SUM=0.0
           DO 22 J=1,7
                SUM=0.0
                DO 23 KKK=1,4
                     SUM=SUM+H(I,KKK)*DSTAR(KKK,J)
  23            CONTINUE
                TEMP3(7+I,J)=SUM
  22       CONTINUE
  21  CONTINUE
C
C                        -           -
C                       |-CDASH       |
C PERFORM TEMP4=TEMP3+  |             |
C                       | GDASH*XDASH |
C                        -           -
C                          -         -
C ASSEMBLE MATRIX TEMP5=!-CDASH      !
C                       ! GDASH*XDASH!
C                        -         -
C
      DO 24 I=1,7
           DO 24 J=1,11
                TEMP5(I,J)=-CDASH(I,J)
  24  CONTINUE
C
      DO 25 I=1,4
           SUM=0.0
           DO 26 J=1,11
                SUM=0.0
                DO 27 KKK=1,11
                     SUM=SUM+(GDASH(KKK,I)*XDASH(KKK,J))
  27            CONTINUE
                TEMP5(I+7,J)=SUM
  26       CONTINUE
  25  CONTINUE
C
C PERFORM TEMP4=TEMP3+TEMP5
C
      DO 28 I=1,11
           DO 28 J=1,11
                TEMP4(I,J)=TEMP3(I,J)+TEMP5(I,J)
  28  CONTINUE
C
C TEMP4 = SIDASH(V,P) [IN FORMULA]
C
C CALCULATE NEWTON STEP BY SOLVING TEMP4*EPS=XZ
C EPS= NEWTON STEP
C      -     -
C XZ=-! CV-CS !
C     ! UW-U  !
```

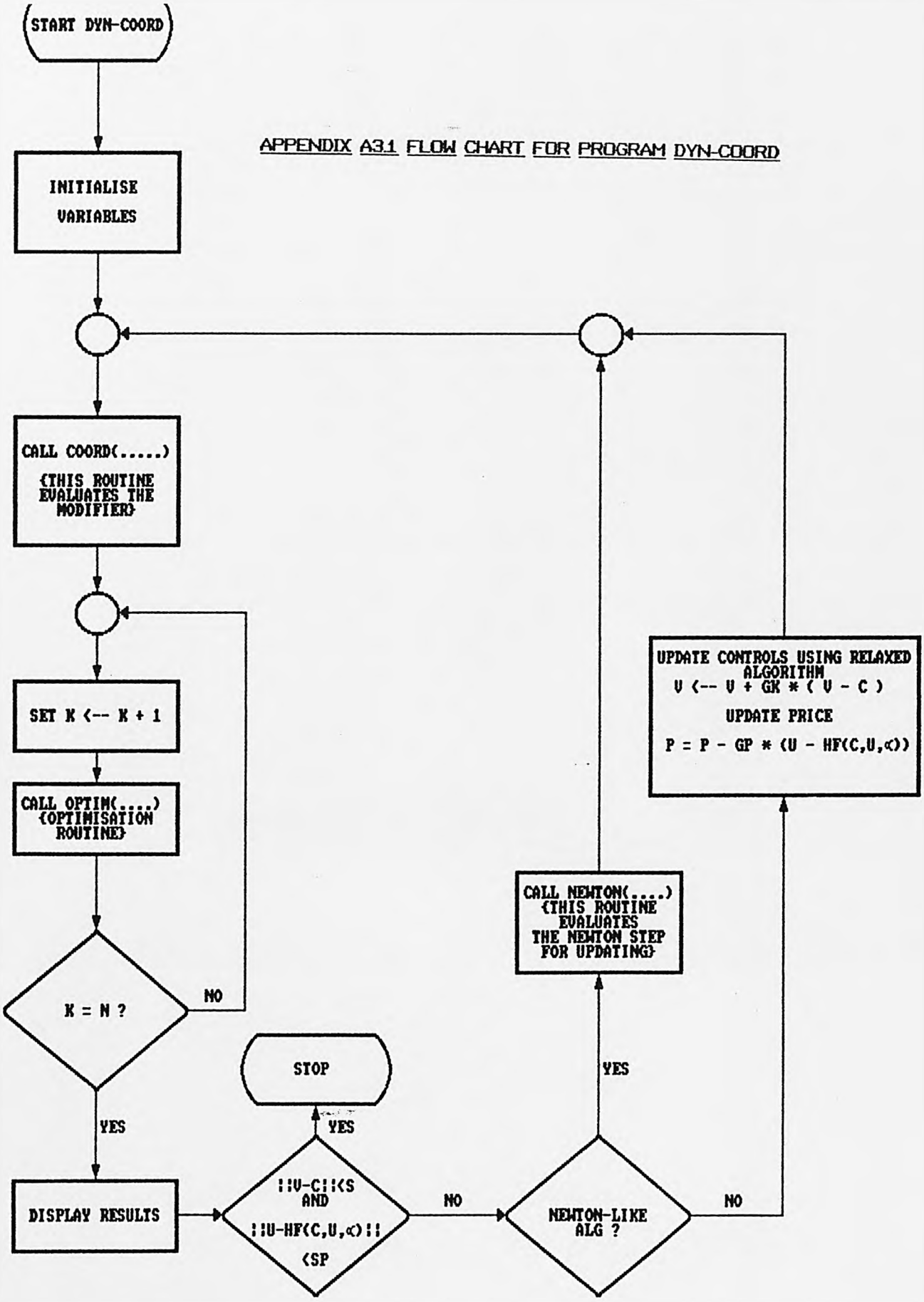
```
C      -     -
C
       KKKK=1
       KKK=1
       DO 29 I=1,11
            IF(I.LT.8)THEN
                 XZ(I)=-EFC(KKK)*EPC
                 KKK=KKK+1
            ELSE
                 XZ(I)=-EF(KKKK)*EP
                 KKKK=KKKK+1
            ENDIF
  29   CONTINUE
C
C SOLVE AS PER ABOVE
C
       IFAIL=1
       CALL F04ATF(TEMP4,11,XZ,11,EPS,AA,20,WKSPCE,BB,IFAIL)
       IF(IFAIL.NE.0)THEN
                PRINT*,'IFAIL IN NEWTON STEP ,IFAIL=',IFAIL
       ENDIF
C
C
C RETURN EPS AS SOLUTION OF THIS SUBROUTINE
C THIS IS THE NEWTON STEP
C EPS(1) TO EPS(7) FOR C
C EPS(8) TO EPS(11) FOR PRICE
C
       RETURN
       END
C
C**********************************************************************
C
C SUBROUTINE FOR EVALUATING Z*
C CALLED FROM COORDINATOR
C
       SUBROUTINE ZSTARI(DRDY,ZSTAR)
       REAL*8 ZSTAR(11,7),DRDY(7,4)
C
C LOCAL VARIABLES
C
       REAL*8 H(4,4),Z(7),DRDYU(4,7),DRDYT(4,7)
C
C INITIALISING MATRICES
C
       DO 1 I=1,4
        DO 1 J=1,4
         H(I,J)=0.0
 1     CONTINUE
C
       DO 10 I=1,7
         DO 10 J=1,4
            DRDYT(J,I)=DRDY(I,J)
 10    CONTINUE
```

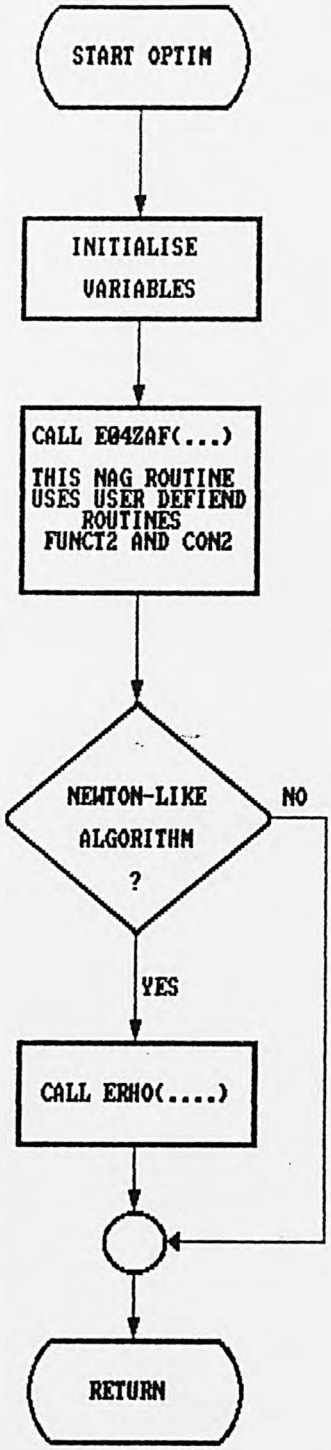
```
C
      DO 2 I=1,11
       DO 2 J=1,7
        ZSTAR(I,J)=0.0
 2    CONTINUE
C
      H(1,2)=1.0
      H(2,1)=1.0
      H(3,4)=1.0
      H(4,3)=1.0
C
C MULTIPLYING H*DRDY
C
      IFAIL = 1
      CALL F01CKF(DRDYU,H,DRDYT,4,7,4,Z,7,7,IFAIL)
      IF(IFAIL.NE.0)THEN
       PRINT*,'IFAIL IN F01CKF IN ZSTARI,IFAIL=',IFAIL
       STOP
      ENDIF
C
      ZSTAR(1,1)=1.0
      ZSTAR(2,2)=1.0
      ZSTAR(4,3)=1.0
      ZSTAR(5,4)=1.0
      ZSTAR(6,5)=1.0
      ZSTAR(9,6)=1.0
      ZSTAR(10,7)=1.0
      DO 3 I=1,7
       ZSTAR(3,I)=DRDYU(1,I)
       ZSTAR(7,I)=DRDYU(2,I)
       ZSTAR(8,I)=DRDYU(3,I)
       ZSTAR(11,I)=DRDYU(4,I)
 3    CONTINUE
C
      RETURN
      END
C
C*********************************************************************
C
C SUBROUTINE DSTARI FOR DSTAR
C  CALLED FROM COORDINATOR VIA SUBROUTINE NEWTON
C
      SUBROUTINE DSTARI(DSTAR,CV,UW,DRDY,DYDW,DMDY,KSTAR)
      REAL*8 DSTAR(4,7),CV(7),UW(4),DRDY(7,4),DMDY(7,4),DYDW(4,4),
     &       DRDYT(4,7),DMDYT(4,7),DYDWT(4,4),KSTAR(4,7)
C
C LOCAL VARIABLES
C
      REAL*8 H(4,4),TEMP(4,7),SUM,TEMP2(4,7),Z(7)
C
C DYDW =-F/UW TRANPOSE
C DRDY = F*/CV
C DMDY = F/CV
C
```

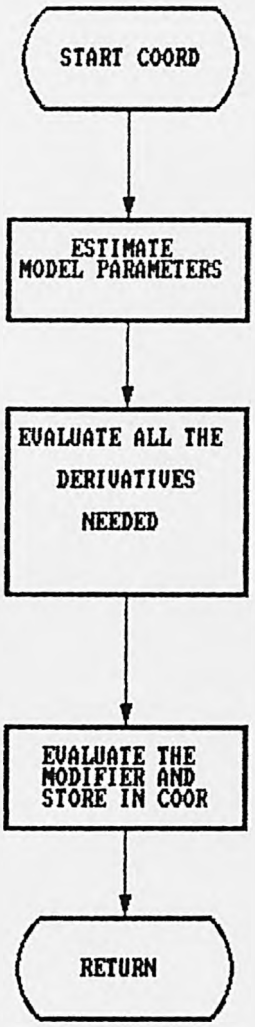
```
C TRANSPOSE ALL MATRICES
C
      DO 10 I=1,7
        DO 10 J=1,4
         DRDYT(J,I)=DRDY(I,J)
         DMDYT(J,I)=DMDY(I,J)
 10   CONTINUE
C
C     DO 11 I=1,4
C       DO 11 J=1,4
C         DYDWT(J,I)=-DYDW(I,J)
C11   CONTINUE
C
      DO 1 I=1,4
       DO 1 J=1,4
        H(I,J)=0.0
 1    CONTINUE
C
      H(1,2)=1.0
      H(2,1)=1.0
      H(3,4)=1.0
      H(4,3)=1.0
C
      DO 20 KKK=1,4
        SUM=0.0
        DO 2 I=1,4
          SUM=0.0
          DO 3 J=1,4
            SUM=SUM+DYDW(KKK,J)*H(J,I)
 3        CONTINUE
          TEMP(KKK,I)=SUM
 2      CONTINUE
 20   CONTINUE
C
      IFAIL=1
C     CALL F01CKF(TEMP2,TEMP,KSTAR,4,7,4,Z,7,7,IFAIL)
      CALL F01CKF(TEMP2,TEMP,DRDYT,4,7,4,Z,7,7,IFAIL)
      IF(IFAIL.NE.0)THEN
       PRINT*,'IFAIL IN F01CKF OF DSTAR,IFAIL=',IFAIL
       STOP
      ENDIF
C
      DO 4 I=1,4
       DO 5 J=1,7
C       DSTAR(I,J)=DMDYT(I,J)+TEMP2(I,J)-KSTAR(I,J)
          DSTAR(I,J)=DMDYT(I,J)+TEMP2(I,J)-DRDYT(I,J)
 5     CONTINUE
 4    CONTINUE
C
      RETURN
      END
C
C*********************************************************************
C
```

```
C
      SUBROUTINE INVERT(A,N,NL)
C
C
C     INVERSE OF MATRIX A(N,N) INTO A(N,N) BY GAUSS-JORDAN METHOD
C
C     A IS UNCHANGED ON EXIT
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 A(N,N),B(20,20),C(20,20)
C
C     SET UP B AS UNIT MATRIX AND COPY A INTO C
C
      DO 20 I=1,NL
           DO 10 J=1,NL
                C(I,J)=A(I,J)
                B(I,J)=0.0
   10      CONTINUE
           B(I,I)=1.0
   20 CONTINUE
C     SELECT COL K FOR ELIMINATION
      DO 60 K=1,NL
C     DO ALL ROWS EXCEPT ROW K
           DO 40 I=1,NL
                IF (I.EQ.K) GO TO 40
                     Z=C(I,K)/C(K,K)
                     DO 30 J=1,NL
                          C(I,J)=C(I,J)-C(K,J)*Z
                          B(I,J)=B(I,J)-B(K,J)*Z
   30                CONTINUE
   40      CONTINUE
C     AND DIVIDE ROW K BY C(K,K)
           Z=C(K,K)
           DO 50 J=1,NL
                C(K,J)=C(K,J)/Z
                B(K,J)=B(K,J)/Z
   50      CONTINUE
   60 CONTINUE
C
      DO  100 I=1,NL
        DO 100 J=1,NL
             A(I,J)=B(I,J)
 100  CONTINUE
C
      RETURN
      END
C
C ***************************************************************************
C
      SUBROUTINE PROJEC(CV,UW,U,IPROJE)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      REAL*8 CV(7),UW(4),U(4),CC(4)
C     ........ LOCAL ARGUMENTS......
      DOUBLE PRECISION ETA, F , RHO, STEPMX, XTOL, A(5,5), G(5),
```

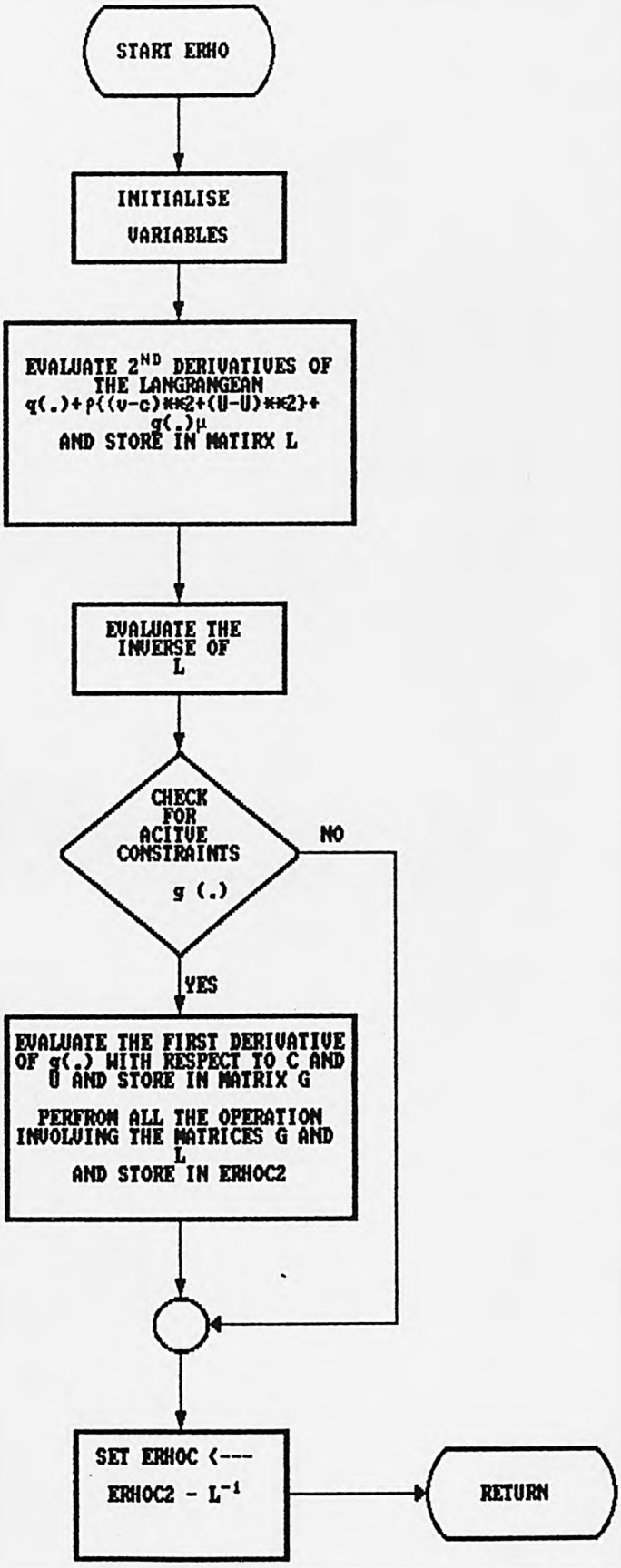
APPENDIX A3.1 FLOW CHART FOR PROGRAM DYN-COORD

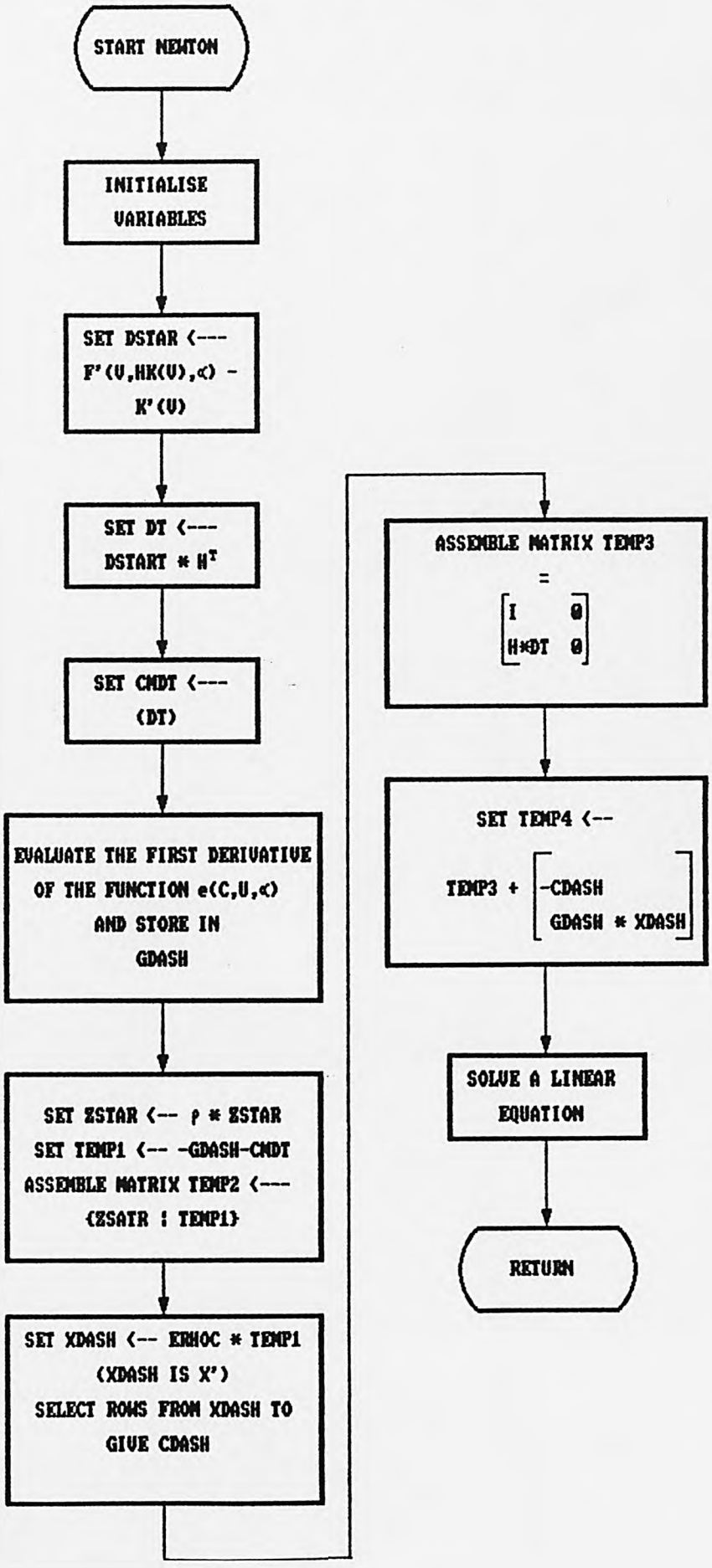



START OPTIM
INITIALISE VARIABLES
CALL E04ZAF(...)
THIS NAG ROUTINE USES USER DEFIEND ROUTINES FUNCT2 AND CON2
NEWTON-LIKE ALGORITHM ?
NO
YES
CALL ERHO(....)
RETURN


START COORD
ESTIMATE MODEL PARAMETERS
EVALUATE ALL THE DERIVATIVES NEEDED
EVALUATE THE MODIFIER AND STORE IN COOR
RETURN

APPENDIX A3.1




START NEWTON
INITIALISE VARIABLES
SET DSTAR <--- F'(U,HK(U),α) - K'(U)
SET DT <--- DSTART * H^T
SET CNDT <--- (DT)
EVALUATE THE FIRST DERIVATIVE OF THE FUNCTION e(C,U,α) AND STORE IN GDASH
SET ZSTAR <-- ρ * ZSTAR
SET TEMP1 <-- -GDASH-CNDT
ASSEMBLE MATRIX TEMP2 <--- {ZSATR ¦ TEMP1}
SET XDASH <-- ERHOC * TEMP1
(XDASH IS X')
SELECT ROWS FROM XDASH TO GIVE CDASH
ASSEMBLE MATRIX TEMP3 = [I 0; H*DT 0]
SET TEMP4 <-- TEMP3 + [-CDASH; GDASH * XDASH]
SOLVE A LINEAR EQUATION
RETURN

```
C     ISOPE BASED ON NORMAL LANGRANGIAN
C     A TWO LEVEL STRUCTURE WITH COORDINATION PASSIVE
C     WITH I/O FEEDBACK
C
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DOUBLE PRECISION CX(7), SK1, SK2, SK3,H, G, CTR(4),CV(7), CS(7),
     *                 UW(4), PLAM(7), AL(4), YREAL(4), YMDL(4),DSQTRC,
     *                 DSQRT,DABS,A(9),LAGM(4),LAGMO(4),U(4),RHO1,RHO2,
     *                 RHO3,YMODEL(4)
C
      INTEGER LN, IA, IAA, M ,IPROJEC(4)
C
      COMMON/CONTR/CS, U, CTR,LAGM ,M2
      COMMON/VARB/CV, UW, R1, R2
      COMMON/SUBN/K
      COMMON/PRMT1/PLAM,AL
      COMMON/PLAPAR/A
C
      OPEN(12,FILE='FUNC1.DAT',STATUS='UNKNOWN')
C
      DO 10 I=1,7
      PRINT*,'ENTER C(',I,')'
      READ*,CX(I)
C     CX(I) = 0.01
      CV(I) = CX(I)
      CS(I) = CX(I)
      PLAM(I)=0.0
      COORO(I)=0.0
   10 CONTINUE
C
      PRINT*,'ENTER A(I),I=1,9'
      READ*,(A(I),I=1,9)
C
      PRINT*,'ENTER SK1,SK2,SK3'
      READ*,SK1,SK2,SK3
      PRINT*,'ENTER G'
      READ*,G
C
      DO 11 I=1,4
        LAGM(I)=0.0
        LAGMO(I)=0.0
 11   CONTINUE
      LN = 4
      IA = 7
      IAA = 7
      M2 = 1
      L = 0
C
 1500 DO 88 I=1,7
      CV(I) = CV(I) + SK1*(CS(I) - CV(I))
   88 CONTINUE
C
      DO 87 I=1,3
         LAGM(I)=LAGMO(I)+SK3*(LAGM(I)-LAGMO(I))
```

```
         LAGMO(I)=LAGM(I)
 87   CONTINUE
C
          CALL COORD(LN,IA,IAA,CV,CS,UW,PLAM,AL,SK2,A,LAGM)
C
      CALL PFORM(CV, UW, FC, AL)
      WRITE(12,*)FC
C
 1000 DO 111 I=1,7
      CS(I) = CX(I)
  111 CONTINUE
C
      K = 1
      CALL OPTIM
      K = 2
      CALL OPTIM
      K = 3
      CALL OPTIM
C
      CALL MODEL(YREAL,CS,AL,YMODEL)
      DSQRTC=0.0
      DO 1001 I=1,7
       DSQRTC=DSQRTC+(CS(I)-CV(I))**2
 1001 CONTINUE
      DSQRTC=DSQRT(DSQRTC/7.0)
      PRINT*,'DSQRTC=',DSQRTC
C
 2311 IF (DSQRTC.GT.G) GO TO 3456
C
      RM2 = M2
      WRITE (1, 44441) M2
      WRITE (1, 66666) FC
      WRITE (1, 99996) (CS(I),I=1,7)
      WRITE (1, 66661) (CV(I),I=1,7)
      WRITE (1, 66662) (UW(I),I=1,4)
      WRITE (1, 33330) (AL(I),I=1,4)
      WRITE (1, 22229) (PLAM(I),I=1,7)
      WRITE (1, 22244) (CTR(I),I=1,4)
      STOP
C
 3456 RM2 = M2
      WRITE (1, 22222) M2
      WRITE (1, 66666) FC
      WRITE (1, 99996) (CS(I),I=1,7)
      WRITE (1, 66661) (CV(I),I=1,7)
      WRITE (1, 66662) (UW(I),I=1,4)
      WRITE (1, 33330) (AL(I),I=1,4)
      WRITE (1, 22229) (PLAM(I),I=1,7)
      WRITE (1, 22244) (CTR(I),I=1,4)
C
      DO 599 I=1,7
      CX(I) = CS(I)
  599 CONTINUE
C
```

```
      M2 = M2 + 1
      GO TO 1500
C
99996 FORMAT ('C',4X,7(1X,F12.6))
12345 FORMAT (6(1X,F12.5))
22229 FORMAT ('PLAM',2X,7(1X,F12.6))
66661 FORMAT ('CV',3X,7(1X,F12.6))
22244 FORMAT ('CONSTRAINTS',1X,4(1X,F12.6))
66662 FORMAT ('UW',3X,4(1X,F12.6))
22222 FORMAT ('**** AFTER',1X,I3,1X,'ITERATIONS ****')
99992 FORMAT ('EF',4X,4(1X,F12.6))
44441 FORMAT (///'CONVERGENT AFTER',I3,2X,'ITERATION '//)
33332 FORMAT ('P',4X,4(1X,F12.6))
33331 FORMAT ('T',4X,4(1X,F12.6))
33330 FORMAT ('AL',3X,4(1X,F12.6))
66666 FORMAT ('PERFORMANCE INDEX',1X,F12.6)
99993 FORMAT ('U',4X,4(1X,F12.6))
      END
C
      SUBROUTINE OPTIM
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C     ........ LOCAL ARGUMENTS......
      DOUBLE PRECISION ETA, F , RHO, STEPMX, XTOL, A(5,5), G(5),
     * C(3), RLAM(3), W(255), X(5), XL(5), XU(5), CL(1), CU(1),
     * CTR(4),LAGM(4)
      INTEGER I, IBOUND, IFAIL, IPRINT, LA, LCLU, LIW, LW, MAXCAL,
     * N, NX, M, IW(30), MEQ, MRNGE, MINEQ
      LOGICAL LAMSET
C     ........SCALAR IN COMMON....
      DOUBLE PRECISION CS(7), U(4)
      INTEGER K
C
      EXTERNAL E04WAY, CON2, E04VAZ, FUNCT2
      COMMON/CONTR/CS, U, CTR,LAGM,M2
      COMMON/SUBN/K
C
      IN=0
      IPRINT = -1
      ETA = 0.001
      XTOL = 1.0D-8
      STEPMX = 4.0D+0
      LCLU = 1
      IBOUND = 0
      LAMSET = .FALSE.
      MEQ = 0
      MRNGE = 0
  1   GO TO (100,200,300),K
  100 N = 2
      MINEQ = 1
      X(1) = CS(1)
      X(2) = CS(2)
      XL(1) = -1.0D+6
      XU(1) = 1.0D+6
      XL(2) = -1.0D+6
```

```
      XU(2) = 1.0D+6
      RHO=1.0
      IF(IN.NE.0.AND.M2.NE.1)THEN
        LAMSET=.TRUE.
        RLAM(1)=LAGM(1)
      ENDIF
      GO TO 400
  200 N = 3
      MINEQ = 1
      X(1) = CS(3)
      X(2) = CS(4)
      X(3) = CS(5)
      DO 40 I=1,3
      XL(I) = -10.0
      XU(I) = 10.0
   40 CONTINUE
      RHO=1.0
      IF(IN.NE.0.AND.M2.NE.1)THEN
        LAMSET=.TRUE.
        RLAM(1)=LAGM(2)
      ENDIF
      GO TO 400
  300 N = 2
      MINEQ = 1
      X(1) = CS(6)
      X(2) = CS(7)
      X(3) = U(4)
C SUBSYSTEM 3
      XL(1) = -1.0D+6
      XU(1) = 1.0D+6
      XL(2) = 0.0D+0
      XU(2) = 1.0D+0
      RHO=1.0
      IF(IN.NE.0.AND.M2.NE.1)THEN
        LAMSET=.TRUE.
        RLAM(1)=LAGM(3)
      ENDIF
  400 LA = 5
      LIW = 30
      LW = 255
      M = MEQ + MINEQ + MRNGE
      IFAIL = 0
      CALL E04ZAF(N,M,FUNCT2,CON2,X,F,G,C,A,LA,W,LW,IFAIL)
      NX =N+MINEQ+MRNGE
      MAXCAL = 250*NX
C     RHO =1.0D+0
      IFAIL = 1
      CALL E04VAF(N,MEQ,MINEQ,MRNGE,M,E04WAY,IPRINT,E04VAZ,MAXCAL,
     * ETA,XTOL,STEPMX,CL,CU,LCLU,IBOUND,XL,XU,LAMSET,X,RHO,RLAM,F,
     * G,C,IW,LIW,W,LW,IFAIL)
C
      GO TO(500,600,700),K
  500 CS(1) = X(1)
      CS(2) = X(2)
```

```
      CTR(1) = C(1)
      LAGM(1)=RLAM(1)
      GO TO 800
  600 CS(3) = X(1)
      CS(4) = X(2)
      CS(5) = X(3)
      CTR(2) = C(1)
      CTR(3) = C(2)
      LAGM(2)=RLAM(1)
      GO TO 800
  700 CS(6) = X(1)
      CS(7) = X(2)
      CTR(4) = C(1)
      LAGM(3)=RLAM(1)
C 800 IF (IFAIL.NE.2) GO TO 900
C     IF(IN.NE.2)THEN
C       IN=IN+1
C       GO TO 1
C     ENDIF
C     PRINT*,'K=',K,'OPTIM NUMBER=',IN
C 900 RETURN
  800 RETURN
      END
C
      SUBROUTINE FUNCT2(IFLAG,N,XC,FC,GC)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C     ........ LOCAL ARGUMENTS ....
      DOUBLE PRECISION FC, XC(N)
     *                 ,GC(N)
C     .......SCALAR IN COMMON .....
      DOUBLE PRECISION PLAM(7), AL(4), CV(7), UW(4),YM(4)
      INTEGER K
      COMMON/SUBN/K
      COMMON/PRMT1/PLAM,AL
      COMMON/VARB/CV, UW, R1, R2
      GO TO (100,200,300),K
C
 100  YM(1)=XC(1)-XC(2)-1.00*UW(1)+AL(1)
      FC=5.0*XC(1)**2+5.0*XC(2)**2- PLAM(1)*XC(1)-PLAM(2)*XC(2)
     *   +0.01*(YM(1)-1.0)**2
      IF (IFLAG.EQ.0)GO TO 400
      GC(1)=10.0*XC(1)-PLAM(1)+0.02*(YM(1)-1.0)
      GC(2)=10.0*XC(2)-PLAM(2)-0.02*(YM(1)-1.0)
      GO TO 400
C
  200 YM(2)=XC(1)-XC(2)-1.0*UW(2)+1.00*UW(3)+AL(2)
      FC = (XC(1)-2.0)**2*2.0 +XC(2)**2 + XC(3)**2*3.0- PLAM(3)*XC(1)
     *    - PLAM(4)*XC(2)- PLAM(5)*XC(3)+0.01*(YM(2)-2.0)**2
      IF (IFLAG.EQ.0) GO TO 400
      GC(1) = 4.0*(XC(1)-2.0) -PLAM(3)+0.02*(YM(2)-2.0)
      GC(2) = 2.0*XC(2)  - PLAM(4)-0.02*(YM(2)-2.0)
      GC(3) = 6.0*XC(3) - PLAM(5)
      GO TO 400
C
```

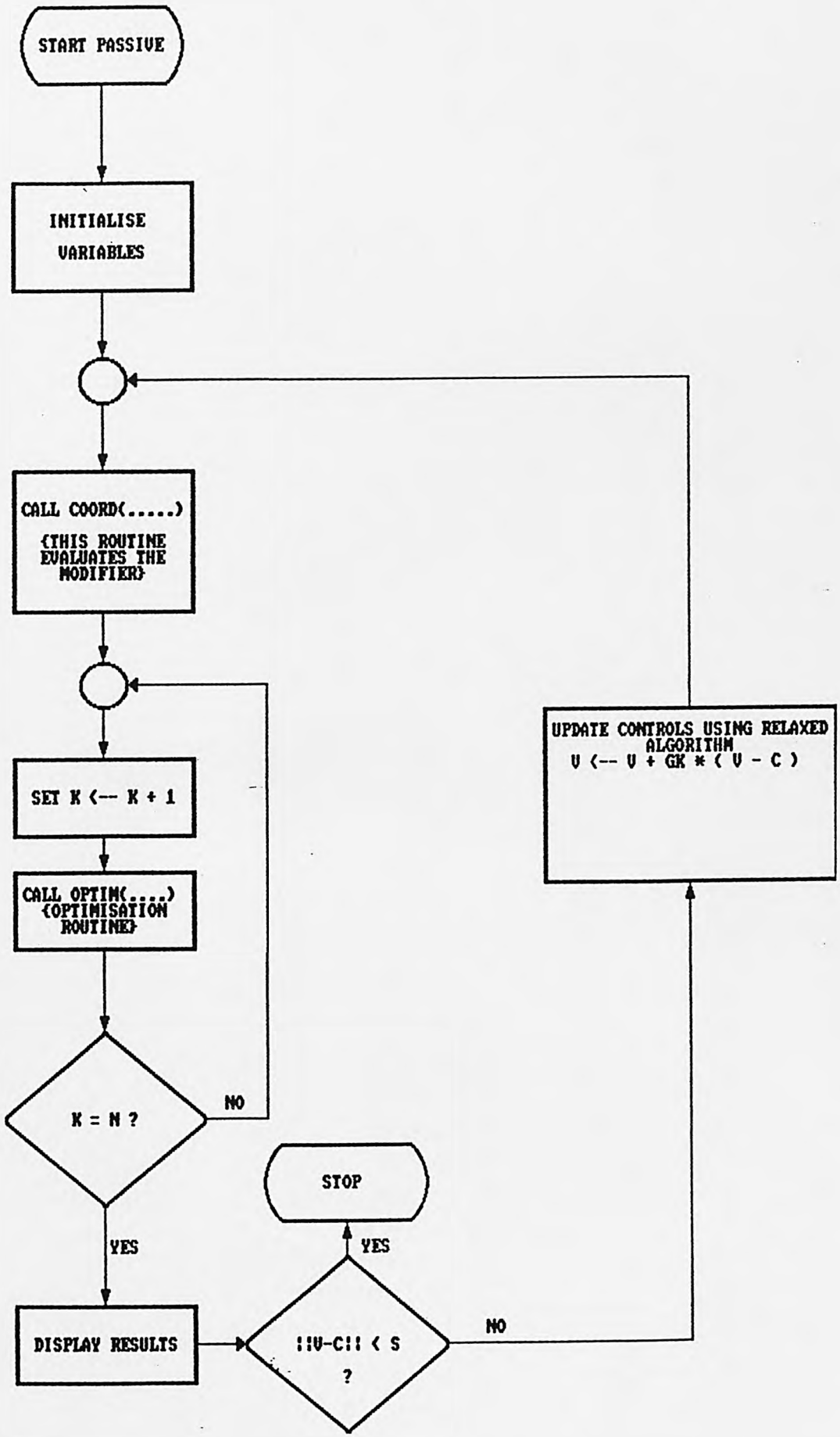
```
  300 YM(4)=XC(1)+2.5*XC(2)-1.00*UW(4)+AL(4)
      FC = (XC(1)+1.0)**2+XC(2)**2*2.5-PLAM(6)*XC(1)-PLAM(7)*XC(2)
     *    +0.01*(YM(4)-3.0)**2
      IF (IFLAG.EQ.0) GO TO 400
      GC(1) = 2.0*(XC(1)+1.0) - PLAM(6)+0.02*(YM(4)-3.0)
      GC(2) = 5.0*XC(2) - PLAM(7)+0.02*(YM(4)-3.0)*2.5
  400 RETURN
      END
C
      SUBROUTINE CON2(IFLAG,N,M,XC,CC,A,LA)
C     ..... LOCAL ARGUMENTS ....
      DOUBLE PRECISION CC(M),XC(N),A(LA,M)
     *                 ,R1,R2,CV(7),UW(4)
      INTEGER K
C
      COMMON/SUBN/K
      COMMON/VARB/CV, UW, R1, R2
C
      GO TO(100,200,300),K
  100 CC(1) = 2.0 - (XC(1)**2+XC(2)**2)
      IF (IFLAG.EQ.0) GO TO 400
      A(1,1) = - 2.0*XC(1)
      A(2,1) = - 2.0*XC(2)
      GO TO 400
  200 CC(1) = 1.2 - (0.5*XC(1)+XC(2)+2.0*XC(3))
      IF (IFLAG.EQ.0) GO TO 400
      A(1,1) = - 0.5D+0
      A(2,1) = - 1.0D+0
      A(3,1) = - 2.0D+0
      GO TO 400
  300 CC(1) = XC(1) + XC(2) + 2.0
      IF (IFLAG.EQ.0) GO TO 400
      A(1,1) = 1.0D+0
      A(2,1) = 1.0D+0
  400 RETURN
      END
C
      SUBROUTINE AMONIT(N,M,X,F,C,NITER,NNF,GLNORM,COND,POSDEF,RHO,
     * RLAM)
C
      DOUBLE PRECISION COND,F,GLNORM,RHO,CNORM,C(M),RLAM(M),X(N)
      LOGICAL POSDEF
      INTEGER K
      COMMON/SUBN/K
C
      RETURN
      END
C
      SUBROUTINE COORD(LN,IA,IAA,CV,CS,UW,PLAM,AL,SK2,A,LAGM)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C     ..... LOCAL ARGUMENT ....
      DOUBLE PRECISION DMDY(7,4),EINC(7), YR1(4), YR2(4), DEDV(7,4),
     *                 YREAL(4), AL(4), DRDY(7,4), QY(7), CCY(7),
     *                 CV(7), UW(4), PLAM(7), CS(7),SK2,A(9),YM1(4),
```

```
     *                H(4,4),TEMP(4,4),TEMP2(7,4),TEMP3(7,4),YM2(4)
     *                ,DCQYM(4),CCYM(4,4),LAGM(4),DQY(4),DQYY(7)
C
C     ESTIMATION UNIT
C     DERIVATIVE OF OUTPUT W.R.T CV
C
      CALL PLANT(LN,IA,IAA,YR1,CV)
      UW(1)=YR1(2)
      UW(2)=YR1(1)
      UW(3)=YR1(4)
      UW(4)=YR1(3)
C
      AL(1) = YR1(1) - CV(1) + CV(2) + 1.00*UW(1)
      AL(2) = YR1(2) - CV(3) + CV(4) +1.0*UW(2) - 1.00*UW(3)
      AL(3) = YR1(3) - 2.0*CV(4) + CV(5) +1.5*UW(2) -0.50*UW(3)
      AL(4) = YR1(4) - CV(6) - 2.5*CV(7) + 1.00*UW(4)
C
      CALL MODEL(YR1,CV,AL,YM1)
C
      DO 1 I=1,7
           EINC(I) = 0.0001*CV(I)
           IF (EINC(I).EQ.0.0D+0) EINC(I)=0.0001
           CV(I) = CV(I) + EINC(I)
           CALL PLANT(LN, IA, IAA, YR2, CV)
           CALL MODEL(YR2,CV,AL,YM2)
           DO 2 J=1,4
                DRDY(I,J) = (YR2(J) - YR1(J))/EINC(I)
                DRDY(I,J)=DRDY(I,J)*SK2
                DMDY(I,J) = (YM2(J) - YM1(J))/EINC(I)
  2        CONTINUE
           CV(I) = CV(I) - EINC(I)
  1   CONTINUE
C
      DQY(1)=0.02*(YM1(1)-1.0)
      DQY(2)=0.02*(YM1(2)-2.0)
      DQY(3)=0.0
      DQY(4)=0.02*(YM1(4)-3.0)
C
      DO 21 I=1,7
         S=0.0
         DO 22 J=1,4
            S=S+(DMDY(I,J)-DRDY(I,J))*DQY(J)
  22     CONTINUE
         DQYY(I)=S
  21  CONTINUE
C
      DO 3 I=1,4
           DO 3 J=1,4
                H(I,J)=0.0
                CCYM(I,J)=0.0
  3   CONTINUE
      H(1,2)=1.0
      H(2,1)=1.0
      H(3,4)=1.0
```

```
      H(4,3)=1.0
C
      DCQYM(1)=0.02*(YM1(1)-1.0)*(-1.00)
      DCQYM(2)=0.02*(YM1(2)-2.0)*(-1.0)
      DCQYM(3)=0.02*(YM1(2)-2.0)*(1.00)
      DCQYM(4)=0.02*(YM1(4)-3.0)*(-1.00)
C
      DO 5 I=1,7
         TEMP2(I,1)=DRDY(I,2)
         TEMP2(I,2)=DRDY(I,1)
         TEMP2(I,3)=DRDY(I,4)
         TEMP2(I,4)=DRDY(I,3)
    5 CONTINUE
C
      DO 7 I=1,7
         S=0.0
         DO 8 J=1,4
            S=S+DCQYM(J)*TEMP2(I,J)
    8    CONTINUE
         QY(I)=S
    7 CONTINUE
C
      DO 11 I=1,7
         PLAM(I)=-QY(I)+DQYY(I)
   11 CONTINUE
C
      RETURN
      END
C
      SUBROUTINE PLANT(LN, IA, IAA, YREAL, CS)
      DOUBLE PRECISION A(7,7), B(7), CD(7), AA(7,7), WKSPC1(30),
     * WKSPC2(10), YREAL(4), CS(7)
      INTEGER LN, IA, IAA, IFAIL
C
      A(1,1) = 1.0D+0
      A(1,2) = 0.02
      A(1,3) = 0.0D+0
      A(1,4) = 0.0D+0
      A(2,1) = 0.5
      A(2,2) = 1.0D+0
      A(2,3) = 0.0D+0
      A(2,4) = -0.05
      A(3,1) = 1.0D+0
      A(3,2) = 0.0D+0
      A(3,3) = 1.0D+0
      A(3,4) = -0.09
      A(4,1) = 0.0D+0
      A(4,2) = 0.0D+0
      A(4,3) = 0.08
      A(4,4) = 1.0D+0
C
      B(1) = 1.5*CS(1) - CS(2)
      B(2) = CS(3) - CS(4) + CS(4)**2*0.1
      B(3) = CS(4)*2.0 - CS(5)*1.30 - CS(4)*CS(5)*0.25 + 0.1
```

```
      B(4) = 1.3*CS(6) + CS(7)*2.5
C
      IFAIL = 1
      CALL F04ATF(A, IA, B, LN, CD, AA, IAA, WKSPC1, WKSPC2, IFAIL)
      IF (IFAIL.EQ.0) GO TO 20
      WRITE (1, 90000) IFAIL
90000 FORMAT (/'ERROR IN F04ATF - IFAIL =',I2/)
   20 DO 30 I=1,LN
      YREAL(I) = CD(I)
   30 CONTINUE
C
      RETURN
      END
C
C ***********************************************************************
***
C
      SUBROUTINE PFORM(CS, U, FC, AL)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION CS(7), U(4), AL(4)
C
      FC=5.0*CS(1)**2+5.0*CS(2)**2+2.0*(CS(3)-2.0)**2+CS(4)**2
     *   +3.0*CS(5)**2+(CS(6)+1.0)**2+2.5*CS(7)**2
     *   +0.01*(U(2)-1.0)**2+0.01*(U(1)-2.0)**2+0.01*(U(3)-3.0)**2
C
      RETURN
      END
C
      SUBROUTINE MODEL(YR,C,AL,YM)
      REAL*8 YR(4),YM(4),C(7),AL(4)
C
C     YM(1)=C(1)-C(2)-0.02*YR(2)+AL(1)
C     YM(2)=C(3)-C(4)-0.5*YR(1)+0.05*YR(4)+AL(2)
C     YM(3)=2.0*C(4)-C(5)-YR(1)+0.09*YR(4)+AL(3)
C     YM(4)=C(6)+2.5*C(7)-0.08*YR(3)+AL(4)
C
      YM(1)=C(1)-C(2)-1.00*YR(2)+AL(1)
      YM(2)=C(3)-C(4)-1.0*YR(1)+1.00*YR(4)+AL(2)
      YM(3)=2.0*C(4)-C(5)-1.5*YR(1)+0.50*YR(4)+AL(3)
      YM(4)=C(6)+2.5*C(7)-1.00*YR(3)+AL(4)
C
      RETURN
      END
```

APPENDIX A4.1 FLOW CHART FOR PROGRAM PASSIVE



APPENDIX B

Nomenclature and value of constants in the fuel-gas mixing system given in chapter 4.

Symbol	Definition
A_W	Effective cross sectional area of water (0.96 m^2)
A_1	Effective cross sectional area of Butene (2.1 m^2)
C_p	Specific heat at constant pressure of the 5 components (3.5, 0.55, 0.45, 0.406, 0.388 KCAL/Kg ^oC)
C_v	Specific heat at constant volume of the 5 components (2.48, 0.44, 0.375, 0.359, 0.349 KCAL/Kg^oC)
C_{pb}	Specific heat of butene
C_{ps}	Specific heat of steel
E_2	Total internal energy of gas mixer (KCAL)
h_i	Specific enthalpy of the i^{th} component (KCAL/Kg)
L_{vap}	Level of Butene in vapouriser
L_{st}	Level of steam condensate
M_{vap}	Mass of butene in vapouriser
M_2	Mass of gas mixer
M_s	Mass of Vapouriser
P_{vap}	Pressure in Vapouriser (psig)
P_2	Pressure in gas mixer
T_{vap}	Temperature in vapouriser
T_{4s}	Temperature of liquid butene
T_2	Temperature in gas mixer
Q	Heat transferred from steam to liquid butene
K_q	Heat transfer coefficient for vapouriser
q_{st}	Rate of steam flow into vapouriser (Kg/s)
q_i	Flow of the various components into the gas mixer
q_{cond}	Rate of flow of condensate from the vapouriser
r	Gas Constants (0.57, 0.0713, 0.038, 0.0259, 0.0204 KCAL/Kg ^oC)
X	Matrix of composition vectors X = (X1, X6, X7, X8, X9, X2)
h_{fgw}, h_{fgb}	Latent heats of water and butene
ρ_W, ρ_b	Density of water and butene (1000.0, 600.0 Kg/m^3)

ρ_2	Gas density of gas mixier
K_i	Flow canstant of valve i