Hierarchical Identification of Large-Scale System Models

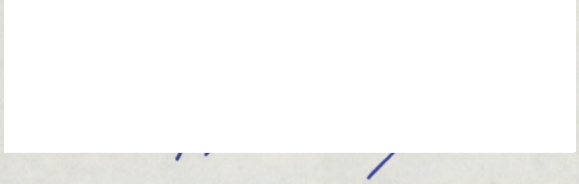
by

Boris R. Jankovic

Dissertation submitted in compliance with the requirements for the Doctor's Degree in Technology in the Department of Electrical Engineering (Light Current) at Technikon Natal

This Dissertation represents my own work

APPROVED FOR FINAL SUBMISSION



Promoter: Professor Vladimir B. Bajic, Pr.Eng., D.Eng.Sc.(EE)

Durban, 24th February, 1997

Acknowledgments

Many people have contributed to make this thesis possible.
I express my sincere gratitude to my promoter, Professor Vladimir Bajic, who was a permanent source of support and encouragement. His help has been unstinting, particularly in those moments when problems encountered seemed to be insurmountable. I am especially grateful for his strict and maximally constructive critiques of my research results tempered with invariably friendly approach.
I am particularly indebted to Professor K. Maciej Przyluski from the Institute of Mathematics of the Polish Academy of Sciences, who scrutinized the final draft of the thesis and whose suggestions contributed to numerous improvements of the final text.
Also, Mr. Meredith McLeod has carefully checked the text and has made numerous suggestions for improvements and I thank him sincerely.
Miss Catherine Radhakrishun has helped with typing some of the most awkward parts of text.
I thank Technikon Natal for making the necessary tools and equipment for this research available to me.
The Library of the University of Witwatersrand has kindly allowed me to use their resources for the period of two years and I thank them for that.
Finally, I thank Gerda and Tanya for their understanding, patience and support throughout this adventure during which I have taken too much of their time.

Abstract

In this study we propose a new concept and methodology of hierarchical identification. The need for such a methodology comes from the fact that identification of large-scale systems (LSSs) by one-shot approach may be numerically very complex. The analysis of LSSs is, in general, not approached by the one-shot methodologies normally associated with non-LSSs. The proposed method of hierarchical identification can be therefore viewed as an extension of LSS methodologies to system identification. LSS methodology aims at breaking up the initial, complex problem into a set of smaller size subproblems. The measure of complexity for optimization-based identification routines is the number of parameters that enter the optimization routines simultaneously. In our hierarchical identification method we aim at minimizing this number.

In our dealing with identification of LSSs we assume that systems can be modelled successfully in wide range of operating conditions by linear models. An additional assumption is that input-output signals used for identification are sufficiently good to enable identification. Internal functioning of the system to be modelled by identification is assumed to be unknown. In order to be able to apply the LSS methodology in system identification, identification is performed by linear composite models. The objective is to enable identification of certain portions of the composite model independently and in parallel. This constitutes Phase 1 of the hierarchical identification method. In Phase 2 interactions among subsystem models that were obtained in Phase 1 are determined in such a way that the resulting composite model optimally matches the given input-output sequences. The composite model structure is chosen in such a way that the number of parameters in any of the phases is as small as possible. In this way some conceptual difficulties are overcome, the most important one being that the LSS methodology is not directly applicable for identification. This is because there is initially nothing to decompose, because the model is only the end result of the identification process.

Although conceptually similar, separate methods are envisaged for hierarchical identification of SISO and MIMO systems. Because SISO systems have only one input and one output, only artificial (mathematical) decomposition can be applied. MIMO models, however, offer a wider variety of decomposition strategies. In addition, we may get some insight into the internal structure of the system by analyzing given input-output signal sequences. The identification procedure obtained in such cases is also a two-phase hierarchical one.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Although modelling and system identification have been well defined and investigated, the aim of this introduction is to briefly review some of the basic concepts in these topics. This is necessary in order to point out some problems in system identification and how we address these problems with our conceptually new approach to identification of system models utilizing the concept of large-scale dynamic systems. This new approach is the main subject of this study.

1.1 Models of Physical Systems and Reality

Our quest for understanding nature is often motivated by the need to predict how certain natural phenomena will behave in future. In other words "to understand a phenomenon is to be able to predict it and influence it in predictable ways" (Sargent 1983). The main tool in this quest for predictability is the concept of model. A model synthesizes quantitative and/or qualitative aspects of observed reality. As there are many ways of describing physical reality, a concept of model must be a broad and flexible one. Models are generally classified as mental (or intuitive), and mathematical. Mental models are used by humans (often subconsciously) when executing certain actions. An action of stopping the car by using brakes is a typical example. The aim of the mental model used in this action is to

predict the stopping distance when a particular force is applied to the brake pedal. When a driver uses brakes, he does not apply the force on the brake pedal in an action following some quick solution of differential equations involving frictions, masses, disk sizes, heat dissipation etc., but instead according to the past experience and some (visual) feedback, estimates whether the applied force will be enough to stop the vehicle on time. When the conditions change in a such a way that no previous experience will guide the driver through the process of stopping the car, the results are often very undesirable. On the other hand, mathematical models are mathematical objects that should imitate the behaviour of the modelled physical object. The closer the correspondence between the imitated behaviour and the actual behaviour of the physical object, the better the model, we say. Mathematical models are our (mathematical) descriptions of reality. As such they must not be confused with reality. The relation between mathematics and reality is a fascinating subject (more on this can be found, from a mathematician's point of view in, for example, Hatcher 1968; similarly, an introductory treatment of the relation between perception, logic and reality can be found in Shaw 1981). In this study, we take a more pragmatic approach where models just describe the (observed) reality more or less accurately, and where distinction between that reality and the model is always borne in mind. Related to this approach is an issue of the so-called true model, the notion that will be referred to frequently in this text. We can think of a "true model" as the one that is the result of some modelling limit process where the end result is the model that describes the behaviour of the physical system without any discrepancy. For example, Ohm's law, which relates physical quantities such as voltage difference across the resistor and the current through it, is known to describe them with limited accuracy due to the nonlinearities involved and also due to the effects of the environment, such as temperature, humidity, etc. One may then create a more accurate (and complex) nonlinear model that would describe the observations better. We can then build more and more complex models whose (external) behaviour imitates more and more closely the observed behaviour of the physical system. The limit of this sequence we will call

the "true model" and we will assume that such limit model (or "true model") will be able to describe the relevant phenomenon exactly. Hence, the true model is a fiction that is often necessary in order to evaluate the quality of a certain model. The quality of a particular model we can then express by some measure of deviation of that model from the true model.

There are various types of models and they can be obtained in more than one way. For example, mental models can be obtained either from the past experience or from the process of learning. Of those types of models that are mentioned above, we will in this study consider only mathematical models in the strict sense, those presented in forms of differential and difference equations. They come in a great variety of forms such as linear, nonlinear, continuous, discrete, etc.

1.2 Mathematical Modelling and System Identification

This section deals with question of how a mathematical model can be obtained. Roughly speaking, there are two ways of generating these models: the so called mathematical modelling and system identification. Because the distinction between the two is not very clear, in what follows we describe what is meant by these terms in this study.

1.2.1 Mathematical Modelling (MM)

Mathematical modelling is often denoted as modelling from first principles. We assume that the internal functioning of a particular process that we wish to model is known. Then we apply the laws of physics governing this internal functioning (usually ignoring or being unaware of the fact that these laws are also models that are more or less accurate descriptions of physical observations) and eventually we get the model of the system. As laws of physics that govern behaviour of dynamic systems are often in the form of differential equations, we may get the model of the system in that form. An example of

this could be found in planetary mechanics, which is built very successfully from Newton's laws of dynamics and gravity. In this study we will be dealing with modelling of the so called dynamic systems, i.e. those that are, roughly speaking, governed by the models in the form of differential or difference equations. For rigorous treatment of the concept of dynamic systems see, for example, Mesarović and Takahara (1989).

1.2.2 System Identification (SI)

The background of system identification is a process in which models are built from the measured input and output data, which are obtained from the system under observation that we intend to model. Here we do not assume that it is exactly known what is happening within a system. Systems that are subject to SI are in many cases dynamic systems used in control engineering. However, SI has much broader application, as biological, socioeconomic and other systems could all be subjected to SI for modelling purposes. For example, similar modelling methodology was applied as early as in the sixties in the study and analysis of finite machines (Conway 1971).

1.2.3 Relation Between Modelling and Identification

The first approach, MM, is logical and elegant. One is able to see the true power of science through application of modelling from first principles. We are able, based on our knowledge of physics, to predict which direction a particular process will take. This all sounds ideal, but in reality, this approach ends up in many difficulties. There are few reasons for this. First of all, we often do not have all the necessary information about the observed process, so we may get an inaccurate model. Also, we may build our model on certain assumptions which may hold only for certain conditions. These conditions are often not fulfilled during the normal operation of the system. On the other hand, certain processes are very difficult to model from the laws of physics despite the fact that they may be deterministic and that there are known laws guiding their behaviour. Examples of these can be found in many textbooks on physics and engineering, where one frequently

encounters sentences like "if we ignore friction" or "if the change in angle is small enough", and so on (a few examples of this can be found in, for instance, Ogata 1970, Palm 1986, Franklin *et al.* 1991). In real life we cannot ignore such details and when it is attempted to include them the model, we find that there is no simple way to describe them. One way to account for these details may be to model the related phenomena as random disturbances (Ljung 1987, Söderström and Stoiça 1989). Therefore, in many cases, at best, we may hope to obtain a model of limited accuracy. However, this approach is often very useful in qualitative analysis of physical systems. For example, by studying Bernoulli's equation, we may conclude that in a typical aircraft wing cross-section the difference in pressures will arise between the upper and the lower side of the wing when there exists relative airflow, resulting in aircraft take-off. However, from this principal idea to a good aerodynamic model of the wing dynamics is a long way.

From these considerations, we can conclude that the question as to which approach to choose is largely a philosophical one. In many cases we deal with systems that are complex, whose exact *modus operandi* is not known and where only input-output records exist. To model such a system, we should follow the route of SI. Of course, if we can get at least some insight into the nature of the observed process, it can help building a model of that process. Therefore, the two approaches should not be viewed as separate, but rather as complementary.

Depending on how much we know about the system that is to be modelled, identification procedures are often termed white, grey and black-box identifications (Hsia 1977). In the case of white-box identification, it is assumed that the system is known to us in great detail, and modelling from first principles can be utilized. The most frequent case is when some knowledge about the system does exist, but there is not enough insight into the system's structure that will enable direct modelling from first principles. This type of identification is referred to as grey-box identification. On the other extreme, when there is no knowledge about the system, we talk about black-box identification. An important point is that the parameters of models obtained by black-box identification may not have

any physical interpretation (Glover and Willems 1974, Ljung 1987).

How much should we be worried about this lack of physical interpretation? The answer depends mainly on what the purpose of the obtained model is. The models obtained by black-box identification should enable us to estimate the behaviour of the modelled process for some given inputs, but, in general, they are not able to tell us what the systems are made of and how they are internally structured.

1.3 The Concept of Large-Scale Systems

The identification methods that will be presented in this text are developed specifically with large-scale systems (LSSs) methodology in mind (Šiljak 1983). This class of systems is becoming increasingly important as objects studied by science and engineering are ever more complex. The class of systems that can be considered as LSSs would include systems as diverse as power systems, economic systems, ecological systems, etc. They are, of course, not limited to some large physical objects - for instance, software (see Sage 1987). A natural question that one may ask is when should a (dynamic) system be regarded as a large-scale one. This may suggest that any definition of the term "large-scale" could be subjective and/or intuitive. It should be noted that terms "large-scale system" and "complex system" are frequently used as synonyms. Simon (1962) described a complex system as "one made up of large number of parts that are interacting in a non-simple way". Šiljak (1983) considers a dynamic system to be complex when the following conditions are fulfilled:

a) the system model has big dimensionality - the system is large;

b) there exist information structure constraints - there are difficulties in locating where decisions are made within the system, and

c) there exists uncertainty due to the complexity - the internal and external nature of the system may not be exactly known, as well as the nature of interactions among

various parts of it.

Mahmoud *et al.* (1985) define a large-scale system as "... system that contains a number of interdependent constituents which serve particular functions, share resources and are governed by a set of interrelated goals and constraints".

These definitions reveal that it is not only dimensionality that characterizes LSSs, although it is an always present and the most distinctive characteristics of such systems (Šiljak 1983). Information flow in the system, possibly different or even mutually exclusive objectives of different components that make up a LSS and even identification of what is input, output and system states, are all problems associated with consideration of LSSs (Šiljak 1983). System dead-times may get another meaning in LSSs as they may be of such magnitude that one can talk about asynchronous system operation due to the propagation of information to and from different subsystems (Mahmoud *et al.* 1985).

Analysis and design of LSSs pose a lot of problems. Because computational difficulties grow much faster then the system size, one would first encounter the problems in this area when dealing with a large, complex problem. This means that the analysis and design of LSSs cannot be approached in the same way as ordinary (non large-scale) systems. One of the most distinguishing characteristics of LSSs is the fact that "one-shot" approach methods cannot, in general, be used with great success in the study of such systems (Šiljak 1978, 1983). Numerical difficulties in analysis of even relatively low-order multivariable systems may be significant (Young and Wang 1987). It is for this reason that analysis of a typical LSS is not performed directly. There are several techniques devised for dealing with LSS problems and most of them are based on some problem simplification techniques. For example, models of complex dynamic systems can be approximated by some simpler, models. Problem simplification techniques that we will mention here are model order reduction, aggregation and decomposition (partition). These techniques can be combined in order to create simpler models (by partitioning or aggregating the models we also get reduced order models).

1.3.1 Model Order Reduction

Strictly speaking, aggregation and decomposition principles could also fall into this category as the result of their application is also a reduced order approximation of the original, complex model (Hickin and Sinha 1980, Šiljak 1983). However, aggregation and decomposition in particular, are applicable to more general problems than those confined to the areas of model analysis. Therefore, by model reduction in this section we will denote methods aimed at reducing the complexity of dynamic models by the reduction in model order. These methods are commonly known as model order reduction techniques (Elrazaz and Sinha 1979, Parnebo and Silverman 1982).

A general overview of model reduction can be found in Anderson and Liu (1989). As the order of the model is the most important measure of the complexity of a linear model, the aim of these techniques can be stated as this: approximate a given high order linear model by some lower-order model in such a way that a certain matching criterion is satisfied. This criterion is measure of performance or "goodness" of such approximation.

The earliest and most natural methods are based on the so-called eigenvalue retention problem (Davison 1966, Mahapatra 1977, Enright and Kamel 1980). The idea is to approximate the high-order model by retaining dominant system modes in a lower-order model and discarding the less important ones. The method was extended for the case of complex eigenvalues in Elazraz and Sinha (1979). Some performance criterion is usually given in advance and the task is then to choose the highest new model order m in such a way that by discarding appropriate eigenvalues λ_{m+1}, λ_{m+2}, ... , the criterion is still satisfied. It is argued in Hickin and Sinha (1975, 1980) that all eigenvalue preservation methods are special cases of aggregation. This will be discussed in the next section.

Another interesting approach is the one suggested by Ouyang *et al.* (1987). In their approach, the modes that have dominant energy contribution are retained in lower-order models instead of those with largest eigenvalues. Using these modes, the denominator of the transfer function of the reduced model can be formed. The parameters of the numerator are then determined by using frequency response matching techniques.

Yet another approach is based on Markov series (Hickin and Sinha 1980, Chen 1984). Markov series can be obtained by expansion of a strictly proper transfer function into a Laurent series. Truncating this series to the few first terms only, we get a partial realization of the original Markov series, which can be then used to create the transfer function of the reduced order model. This reduction yields reduced order models with good transient matching, at the cost of the steady-state characteristics (Hickin and Sinha 1980). Details on how the steady-state characteristics can be improved and the related notion of partial realizations of transfer function matrices in multivariable models are given also in Hickin and Sinha (1980).

The matching criterion is usually chosen to be represented by some norm involving the difference between outputs of the original high order model and its lower-order approximation. This criterion is often taken as integral square error between the output of the original model and the reduced order model which is then minimized by the choice of parameters of the reduced order model (Meier and Luenberger 1967, Wilson 1970). This criterion is known as model output error criteria (Hsia 1977). Another approach is based on quadratic criterion minimizing the equation error. This approach has some numerical advantages over the output error criterion as the minimization of equation error criterion leads to a linear problem (Obinata and Inooka 1976, Eitelberg 1982). Some general issues relating to the effects of applying reduced order models in the synthesis of controllers can be found in Allemong and Kokotović (1980), Doram (1987) and Kreisselmeier and Mavenkamp (1988).

It is worth noting that all of these methods were developed for models in open-loop operation. If the original high-order model operating in closed-loop is replaced by reduced order one, the question is how well its performance will be matched in this case. This question is not easy to answer for the general case, but it is reasonable to assume that the better the model approximation, the better will be its closed-loop performance (Šiljak 1983). On the other hand, when it is known that reduced order model will be placed in the closed loop, the approximation criterion for reducing the given open-loop high-order

model can often be relaxed as such configuration is less sensitive to model inaccuracies (Pollard and Brosilow 1985).

We can conclude that the natural habitat for applicability of model order reduction techniques lies mostly within single input, single output models, as parameterisation of these model is uniquely defined by its order (as the structure of system matrices or polynomials defining models as transfer functions are uniquely defined - this will be studied in details in Chapter 2). In the case of multi-input, multi-output models, reduction of just the order of such models may not be enough as in addition to the model order, model parameterisation also depends on the dimension on input and output vectors. In these cases, aggregation and decomposition techniques should be applied (Šiljak 1978).

1.3.2 Aggregation

The method of aggregation has an aim of reducing the number of system variables in the resulting model. This is achieved by aggregating or grouping certain states together in order to form a new set of states. The principle of aggregation goes back to the original work of Lyapunov (see collection of works, Lyapunov 1956) in the context of stability analysis. The method itself, in the form usually utilized, was initially developed by economists for the study of a market, which is LSS with large number of states (Aoki 1978). It was observed that some of the states may behave in a similar way and that they can be somehow grouped together, thus forming subgroups of states (an example of this can be found in Bajić and Petrović 1980). For example, the state of a national economy can be described with many parameters, such as economic growth of each of the sectors, balance of payments, interest rates, unemployment rate, etc. By means of aggregation, the state of an economy can be described by a much smaller set of variables. For example, "the most important aggregates for monetary policy purposes remain the M3 money supply and the total amount of bank credit extended to the private sector" (Stals 1996). All monetary policy indicators (states) are therefore mapped (aggregated)

into only two. The new model in which these subgroups assume the role of the new states is then called an aggregate model. In this way we deal with aggregates in a reduced dimension state space instead of the original models which have much larger dimension of the relevant state space. To be able to perform aggregation successfully, the selection of variables to be grouped together is obviously crucial. Intuitively speaking, the more physical insight into the system available to us, the better the chance that aggregation will be successful.

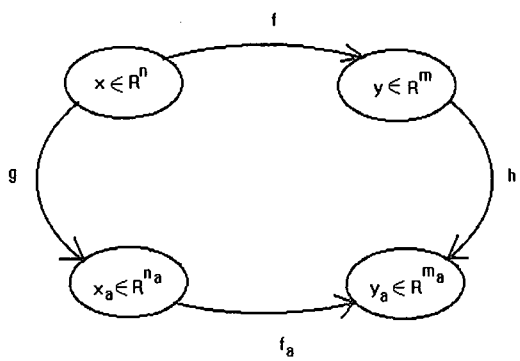


Figure 1-1: Aggregation mapping

An excellent overview of aggregation, and corresponding notion of disaggregation, is presented in Aoki (1971, 1978) and Mahmoud *et al.* (1985). The aggregation process can be presented in the following way. Let f be a mapping such that $f : R^n \ni \mathbf{x} \mapsto \mathbf{y} \in R^m$. Our goal is to find some transformations $g : R^n \ni \mathbf{x} \mapsto \mathbf{x}_a \in R^{n_a}$, $h : R^m \ni \mathbf{y} \mapsto \mathbf{y}_a \in R^{m_a}$ and $f_a : R^{n_a} \ni \mathbf{x}_a \mapsto \mathbf{y}_a \in R^{m_a}$ (Fig. 1-1) such that $n_a < n$ and $m_a < m$ and where the objectives regarding the behaviour of the reduced order model and the original model are fulfilled. The process of mapping f into f_a is called aggregation.

If the relation

$$h[f(\mathbf{x})] = f_a[g(\mathbf{x})] \tag{1.1}$$

holds for every $\mathbf{x} \in R^n$ we talk about perfect aggregation. In many cases aggregation will yield only approximate mappings when (1.1) holds approximately. In order for

aggregation to be successful, we need some measure of deviation of aggregate model from the original model. This could be, as usual, some suitably defined distance in R^{m_a} between the original mapping and its aggregate,

$$d(\mathbf{x}) = \|h[f(\mathbf{x})] - f_a[g(\mathbf{x})]\|. \tag{1.2}$$

To have the aggregate as close as possible to the original mapping, this distance has to be minimized by suitable choice of g, h and f_a. The concept of aggregation can be easily applied to linear models as they describe mappings from the input sequence into the state variables or output variables by means of linear transformations. In principle, through aggregation we create models that are less complex and more suitable for the analysis. The question that arises now is how can we extract the original system's states from the aggregate model. This can be achieved through disaggregation, where we need to find some inverse mappings $g^{-1}(\mathbf{x}_a)$ and $h^{-1}(\mathbf{y}_a)$ that are ideally inverses of the aggregation mappings.

We can be more specific when we talk about aggregation of dynamic systems. Let us consider a linear, time invariant, dynamic system given by

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} = \mathbf{C}\mathbf{x} \end{cases} \tag{1.3}$$

where $\mathbf{x}$ is an n-component state vector, $\mathbf{u}$ is a p-component input vector and $\mathbf{y}$ is a q-component output vector. Matrix $\mathbf{A}$ is an $n \times n$, $\mathbf{B}$ is $n \times p$ matrix and $\mathbf{C}$ is a $q \times n$ matrix. We can now define the aggregate of the state vector $\mathbf{x}$, denoted as $\mathbf{x}_a$, in the following way

$$\mathbf{x}_a = \mathbf{M}_a\mathbf{x} \tag{1.4}$$

where $\mathbf{M}_a$ is $n_a \times n$ aggregation matrix, $n_a < n$. Thus, aggregation in dynamic systems is achieved by modifying the state vector, while, frequently, dimensions of input and output vectors are left unchanged (Aoki 1978). The objective in case of such aggregation

would be that outputs of the reduced-order model and the original model behave as similarly as possible for the given set of inputs. That would imply that aggregation is particularly useful when the number of states of the dynamic system given by (1.3) is significantly higher than the number of inputs and outputs. To obtain a dynamic equation for aggregate model, relation (1.4) is used in conjunction with (1.3), and after some elementary matrix manipulations we get

$$\dot{\mathbf{x}}_a = \mathbf{F}\mathbf{x}_a + \mathbf{G}\mathbf{u} \tag{1.5}$$

providing that

$$\mathbf{F}\mathbf{M}_a = \mathbf{M}_a\mathbf{A} \tag{1.6}$$

holds and that $\mathbf{G} = \mathbf{M}_a\mathbf{B}$. Conditions under which a unique solution for matrix $\mathbf{F}$ exists can be found in Aoki (1978) and Lancaster and Tismenetsky (1985). It can also be shown (Aoki 1978) that the model given by (1.5) inherits the stability property of the model given by (1.3).

Another very important feature of aggregation is that it preserves the system eigenvalues (Mahmoud *et al.* 1985). Therefore, the eigenvalue λ_i of the matrix $\mathbf{A}$ of the model given by (1.3) will retain its value in the aggregate model matrix $\mathbf{F}$ in (1.5), while the corresponding eigenvectors will be linearly transformed. If the set of eigenvalues of the original model matrix $\mathbf{A}$ is denoted as E and the set of an aggregated model matrix $\mathbf{F}$ by E_a then the following relation will hold

$$\forall \lambda_i \in E_a \Longrightarrow \lambda_i \in E \tag{1.7}$$

or, in other words, eigenvalues of $\mathbf{F}$ are also eigenvalues of $\mathbf{A}$. The converse is not true in general as $E_a \subseteq E$ and, because in order for aggregation to be meaningful, matrix $\mathbf{F}$ should always be of a smaller dimension than the matrix $\mathbf{A}$, resulting in a smaller eigenvalue set. This is the reason why model reduction based on the eigenvalue retention

can be viewed as a special case of aggregation.

Aggregate models are, as already hinted, approximations of the original models. Therefore, it is important to somehow optimize the aggregation models in such way that certain error criteria are minimized. Some optimization criteria can be found in Aoki (1978, 1971), Mahmoud *et al.* (1985) and Drenick(1975). As an illustration, let the number of outputs in the original model and in the aggregated model is kept the same. This is often the case as the input-output characteristic of the original model should be preserved in its aggregate. Then the following criterion may be used

$$J = \sum_{i=1}^{p} \sum_{j=1}^{q} \int_0^T (y^{ij} - y_a^{ij})^2 dt \tag{1.8}$$

where p is the number of inputs, q is the number of outputs, and y^{ij} is the i-th output due to the j-th input, provided that all other inputs are zero, while subscript a refers to the aggregated model outputs.

Another important feature of aggregation is that through this process the original structure of the LSS can be preserved. This can be achieved by applying aggregation to each of the subsystems making up the LSS. In this way the nature of interactions among the subsystems is preserved, as only the subsystems themselves are aggregated. Should some subsystems be perceived as behaving in a similar way, they can be aggregated (grouped) in such way that new subsystems are formed, incorporating some of the subsystem interactions (Šiljak 1978). The end result is that the number of subsystems and interactions in the final aggregate model is smaller than in the initial case.

Once again, reconstructing the states or output vectors from the aggregated model is done by disaggregation. Under some conditions, perfect disaggregation is also possible. Details can be found in Aoki (1978).

In practice, aggregation is a process guided by insight into the internal structure of the system. Model reduction methods from the previous section inherently tend to remove any physical meaning associated with the original model's states. One way of preserving

the character of the original model in the aggregate model is proposed in Coxson (1985). The original state vector is partitioned in some way (for instance, by grouping similarly behaved states together), components in each of the groups are averaged and then, the so obtained averages from each of the partitions are taken as the new states of the aggregate state vector. Aggregation methods for the models in the form of a transfer function matrix can be found in Hickin and Sinha (1980).

1.3.3 Decomposition

A third method for reducing the dimensionality of the problem is that of decomposition. Decomposition is a relatively broad term as it encompasses various methods of partitioning a complex problem into a set of smaller and less complex problems. The aim of decomposition is to simplify and speed up the analysis of complex problems. This method is widely used in many areas of science and engineering. There are many papers reported in literature utilizing the decomposition principle in the study of LSSs, both for structural and numerical analysis (Šiljak 1978, Brucoli *et al.* 1987, Drenick 1975, Malinowski *et al.* 1985, Parnebo and Silverman 1982, Pearson 1971, Prywes 1984, Singh *et al.* 1986, Sundareshan 1976, Grujić *et al.* 1987).

The decomposition process normally consists of two steps, the first consisting of breaking up the initial problem into simpler ones and the second of establishing the relations (interactions) among them. In that way we can analyze subproblems separately. When we include interactions among the subsystems, we obtain the composite model that should preferably have the same characteristics as an initial, complex one. For dynamic systems, the decomposition method is particularly useful for weakly coupled systems (Gajić *et al.* 1990, Kecman 1988), as subsystems have high degree of autonomy. How we decompose (partition) the system will depend on the aim of the decomposition as well as on the amount of *a priori* knowledge about the system. Sometimes we can decompose the system based on some physical considerations, in which case the obtained subsystems will have a physical interpretation. Certain methods are developed based on this approach,

particularly for the analysis of electrical networks (Šiljak 1978, Vittaco and Michel 1978, Milić and Bajić 1988, Bajić 1989, 1996). The problem is, however, that such methods are not universal in the sense that they cannot be successfully applied to other classes of problems, different from those for which they were developed.

In other cases, decomposition has solely an aim of breaking up the large numerical problem into a set of smaller, easier subproblems. In this case, a large-scale dynamic system can be decomposed into a set of smaller size dynamic systems artificially. With such decomposition we do not attempt to assign any physical meaning to subsystems obtained in such a way and only the final model is subject to interpretation. This type of decomposition is termed mathematical decomposition (Šiljak 1978). Mathematical decomposition is very important in the case when there is very little physical guidance on how to decompose the original problem. This type of decomposition will be of particular importance in this study. More on the formal treatment of decomposition techniques will be given in Chapter 3.

1.3.4 Identification of Large-Scale Systems

One particular aspect of LSSs analysis is their identification from input-output sets. Such identification will, of course, share all the problems associated with analysis of LSSs. As the end result of the identification process is a model of the observed system, it is clear that the standard aggregation-decomposition approach is not directly applicable, as there is initially nothing to decompose. Therefore we are forced to apply "classic" one-shot methods in identification of LSSs. This is clearly a very difficult task, as the issues of both numerical stability and enormous computational effort are very prominent. If the identification method for a LSS is based on optimization, then the parameter estimation techniques will lead to a large-scale optimization problem. This is important to note, since in this study we base the identification on the optimization approach.

One of the major problems in parameter estimation is the number of parameters that enter the optimization routines simultaneously. It is clear that the larger the number

of parameters the bigger are the computational difficulties, both in terms of processing time and numerical stability. The number of arithmetic operations per iteration is of the order $n^3 + mn$ and $n^2 + mn$ for Newton and quasi-Newton methods respectively, where n is the number of variables and m the number of constraints (Gill and Murray 1974). Required memory space is of the order $n^2 + mn$ for both techniques. However, nothing precise can be said about the required number of iterations due to the problems inherent in optimization algorithms.

Although memory considerations are becoming of less importance, the computation time required may be excessive. This is of particular significance for real time applications of identification. In addition to the excessive computational time required, complex functions which represent an error criterion contain potentially large number of local minima, saddle points, narrow valleys etc., which makes choice of the initial set of parameters a non-trivial exercise. This makes automation of identification difficult as it very often requires an interactive procedure for correction of initial values of parameters. The decision as to whether the obtained minimum is global (or good enough) also has to be done manually, i.e. interactively. Thus, identification of LSSs may take too long, and as such, may not be efficient when models are required within a given time interval, for example in self-adaptive control of large-scale and relatively fast changing systems.

1.4 Problem Statement: Hierarchical Identification of LSSs

Now we can define the problem to be investigated in this study. This is:

- **Development of methodology for identification of processes on the basis of LSS philosophy that will enable significant reduction in the computational effort required.**

Efforts will be made to achieve the efficient identification of LSSs suitable for practical implementation. The main ingredient of all such methods will be minimized computational effort as a consequence of the minimized number of parameters that have to be determined simultaneously in any particular stage of the identification process.

The hypothesis associated with the problem stated above is that the black-box model of a LSS can be identified utilizing the LSSs methodology.

Assumptions are that:

1. the input-output records to be used for identification contain sufficient information for determination of linear models, and

2. no information is available about the internal structure of the observed system.

The delimitation adopted is that only linear models will be used in this study.

The benefits of this approach will be that the black-box model of a LSS based on methods developed in this study could be obtained much faster, broadening the scope of the applicability of identification methods. The way of achieving this is through the methods of hierarchical identification, which will permit the utilization of parallel processing.

The initial idea of this methodology of hierarchical identification that is fully based on LSS philosophy was suggested to the author by Bajić (1994), and some initial results on this methodology are presented in Janković and Bajić (1996a, 1996b, 1997a, 1997b, 1997c, 1997d) and Bajić and Janković (1997). The methods that are developed are presented in Chapter 3 for single-input, single-output (SISO) systems and then in Chapter 4 for multivariable systems.

In order to present these methods, it is necessary to describe very briefly some classical approaches to SI of both SISO and multi-input, multi-output (MIMO) systems. This is done in the next chapter.

Chapter 2

Overview of System Identification

System identification is today a well matured area. There are many approaches to SI that are analyzed in literature and the scope of their applicability is well known. This chapter has the aim of presenting some very basic SI methodology in order to highlight some of the difficulties in SI, which are addressed by our new approach of hierarchical identification developed in Chapters 3 and 4.

2.1 Basic Steps In System Identification

As already mentioned, SI consists of a procedure for building models of systems from input-output data. In this chapter we aim to briefly describe the essence of the main SI methods. There exists an extensive literature on both the theory and application of SI which can be consulted for any in-depth study of SI. There are some excellent books that treat this subject systematically (Ljung 1987, Söderstöm and Stoiça 1989, Hsia 1977).

Briefly stated, to perform SI it is necessary:

1. to obtain input-output records (either from the results of an experiment, if it is possible to perform one on the systems, or from the data collected during the normal operation of the system);

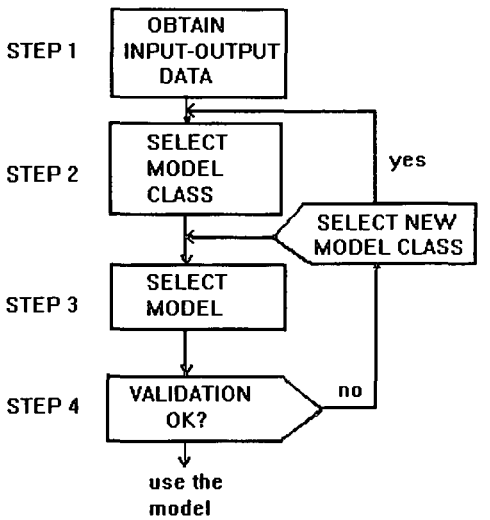


Figure 2-1: Steps in SI

2. to decide on the class or set of models that will be considered as potential candidates for the final model;

3. to choose from this set of models some models which we will try to fit to recorded data by adjusting their parameters, and

4. to validate the obtained model.

This procedure is schematically shown in Fig. 2-1. The algorithm of this procedure often makes a loop because if step 4 fails, we go back to (usually) step 3 or 2 (sometimes we may need "better" input-output record - this will be discussed later, and then we may have to repeat the experiment from step 1, if possible).

It was mentioned earlier that the amount of *a priori* knowledge about the system to be modelled may have significant influence on the SI procedure. Therefore, there are two distinct cases. The first one is where there is very little or no knowledge about the internal structure of the system; then we are talking about black box identification. The black box problem is obviously a difficult one, so we normally resort to certain assumptions that will make the identification problem easier. In the other case, it is assumed that the

internal structure is known to us, therefore enabling us to choose a proper model, and all that we need to do is to estimate parameters of such a model. In this case the problem of identification is reduced to the problem of parameter estimation.

The above mentioned four basic SI steps are briefly described in what follows.

2.1.1 Input-Output Data

Not every input-output record is good enough for the purpose of SI. Outputs may be too corrupted by noise, inputs may be recorded for too short a time interval or their spectra may not be rich enough. The presence of noise obviously hides part of the output variation due to an input and identification is then more difficult. The duration of input signals should be longer then the dominant time constant of the system (Godfrey 1993). The minimal length of input sequence for identification of a class of models can be found in Sontag (1980). The input signal should also be persistently exciting, which, roughly speaking, means that it can excite every relevant mode of the system, or alternatively, that the bandwidth of the signal is not shorter then the bandwidth of the system (Godfrey 1993). By system mode, in the case of linear time-invariant state-space models, we will understand a component of the output that corresponds to a particular eigenvalue of the model. Signals like white noise and the pseudo-random binary sequence owe their popularity to the richness of their spectra. More in-depth cover on these topics can be found in Godfrey (1993) and Goodwin and Payne (1977). Sometimes identification has to be performed from the input-output sequences obtained during the normal operation of the system. These sequences may not carry enough information for identification purposes and the accuracy of the models obtained by identification from such records may not be good. Some suggestions in this respect can be found in Janković and Bajić (1997a, 1997c).

2.1.2 Choice of the Model Class

As the output of SI is a model of the system, we must choose one to start with. This decision is not easy. First of all, as input-output records are normally in some tabular form, the question is whether to search for continuous or discrete models. Because input-output records are in fact sampled values of (possibly) continuous signals, we may also call them input-output (I/O) sequences. The choice of discrete models is a popular one and the majority of reported work in literature is for this class of models. They are also easier to implement on digital computers, as they are modelled via difference equations. However, one can successfully fit sampled data into the continuous models (Söderström and Stoiça 1989). On the other hand, continuous models are more natural to us as most of the dynamic physical phenomena are modelled via differential equations which yield continuous models. The parameters of such models may have physical meaning. For more about the relation between discrete and continuous models, see Rao and Sinha (1991).

While the choice of discrete or continuous models can sometimes be a matter of taste, some other choices are more governed by the practical issues and actual knowledge of the system. The most important one is whether linear or nonlinear models are to be used. If the simplification of the system description is a governing factor, linear models may be used. If the nonlinearities are in the essence of a system's *modus operandi*, a nonlinear model must be used, no matter how this may complicate the analysis. However, in most cases, boundaries between applicability of linear and nonlinear models are not as sharp. For instance, an essentially nonlinear system may have a linear range of operation, and in this range it can be satisfactorily described by linear models. When using such models, it must be clearly understood in which operating range they are applicable. To illustrate this dilemma, linear vs. nonlinear, a proponent of linear systems would point out that linearity works well in many cases and mathematical treatment of such systems is well founded and powerful. On the other hand, a proponent of nonlinear systems would point out that virtually all of the processes in nature are nonlinear, therefore justifying the

employment of nonlinear models, despite the lack of elegant mathematical theory for treatment of such models. As they are both right in their claims, we will not pursue this issue any further, and in this text predominantly linear systems will be considered, although we acknowledge that sometimes they won't do well if nonlinear phenomena take considerable effect.

System identification is performed in either a probabilistic (stochastic) or a deterministic framework (Ljung 1987). Utilization of a deterministic model framework implies the assumption that in principle, parameters of the model and input-output signal set can be described by functions in the ordinary mathematical sense. More on deterministic identification of dynamic systems can be found in, for example, Heij (1989). In the case of a stochastic framework, such assumption is not used. In this text that framework will be deterministic, although the effects of noise will not be ignored. On the other hand, the core of the ideas related to the hierarchical identification, which is the main contribution of this study, is essentially independent of any particular identification approach.

In addition to these, other considerations (Ljung 1987, Söderström and Stoiça 1989) deal with issues such as parametric vs. nonparametric models, time domain models vs. frequency domain ones, time-varying vs. time invariant etc. Within the class of linear systems, both input-output and state-space descriptions are used for SI. Hierarchical identification of single-input, single-output and multi-input, multi-output dynamic systems will be treated separately as there are certain differences in approach to them.

If the input-output data is first recorded, and SI is accomplished after that, we talk about off-line identification. If we update the values of parameter estimates while obtaining new, fresh, input-output records, we talk about on-line identification (Hsia 1977). The latter method can be used to estimate parameters of time-variable systems. In such cases, normally, more recent data will carry more weight in the identification process (Niv and Fisher 1996).

2.1.3 Choice of Models Within a Model Class

Once we have decided on the class of models, we must be more specific in our choice in order that some fitting of the members of that class to I/O sequence can be done. In the case of single-input, single-output (SISO) linear models this will translate to the choice of model order, as this is the main measure of model complexity within a linear framework. There is no general rule for determination of the model order other then trial; some specific examples can be found in Hirshberg and Merhav (1996). In practice, we start from a low-order model and see how well its output matches the output data. If we are not happy with the results of fitting, we take a higher order model, and so on. The question that arises is when to stop. Normally, one would stop on a particular model order when any further increase in model order brings insignificant improvement in model output error. Another aspect of the model choice is the number of free parameters that will enter the optimization routines simultaneously if SI is done via an optimization approach. It is intuitively clear that this number should be as small as possible. This problem is central to the development of our new identification approach, presented in Chapters 3 and 4.

A note regarding the model order is worth mentioning here: just as from the discussion about the relation between model and system we have concluded that there can be no true model, it can also be said that the notion of the "true" model order is somewhat misleading. It would be right to say only that in the linear range of the operation of the system, a model of particular order describes the I/O observations better than a model of some other order.

After all the considerations regarding the model choice, we have to estimate the parameters of such model. If optimization based SI is used, we define some error criterion that we consequently try to minimize in order to get the parameterized model that will best fit the observed I/O data. Numerous parameter estimation techniques are used for this purpose. We mention here two of the most popular: least-squares method and maximum likelihood method (Ljung 1987, Söderström and Stoiça 1989, Hsia 1977,

Goodwin and Payne, 1977). However, there are other, non-optimization based parameter estimation techniques, such as those used in process control, on which some of the well known controller tuning techniques, like for example, Ziegler-Nichols, Cohen-Coon, CHR, etc., are based (see De Carvalho 1993).

2.1.4 Model Validation

Model validation deals with an issue of whether the obtained model is a good one. How good a model is will largely depend on the purpose of its use (Ljung 1987, Poola *et al.* 1994). If we need only a rough model, we may be able to get one relatively quickly. Such models are often used in regulation, where closed loop is employed to reduce the negative effect of model imprecision (Pollard and Brosilow 1985) or to improve transient properties of the controlled system. If we need a high precision model, then we have to validate the obtained model more carefully.

Model validation is performed in several steps. First, it should be noted that the obtained model is good for the I/O sequence on which it was built because it was adjusted to that sequence. That is not a guarantee that it will work well with some other input signals. The main reason for this is that the original I/O sequence may not have been persistently exciting and therefore some of the system modes may not be detected. That in turn may manifest as a poor result for some other input signal which places more emphasis on other, previously undetected modes. Therefore, the obtained model should be simulated for some other inputs. If the I/O sequence is long enough, a useful trick is to save, for validation purposes, a portion of that sequence that was not used for identification (Ljung 1987).

The validation process and model comparison will in general depend on the validating criterion, which may be, for instance, some measure of misfit of the model outputs, like integral square error (ISE), where the error is defined as the difference between model output and measured output sequence.

One other issue that should be kept in mind when evaluating a particular model is its

complexity. We, naturally, should choose the simplest model possible that satisfactorily describes the system. Almost always, we wish to reduce the complexity of the obtained model. For instance, we may look for pole-zero cancellations in linear system models governed by transfer functions. Very often our concern is that the selected model may be overparameterised. Identification of such models may lead to computational difficulties (Söderström and Stoiça 1989). Because of this, it is very important to select the minimal dimension model from the set of acceptable models, and a number of research papers deal with this (Bingulac 1976, Buddin 1971, Denham, 1974). This subject will be studied later in more detail.

The result of the identification process should be, ideally, the "true" model. Validation is a process that should confirm that the model sequence obtained as a result of identification converges to the true model. Another important notion is the one of identifiability (Söderström and Stoiça 1989, Ljung 1987, Grewal and Glower 1976, Söderstrom *et al.* 1976, Brown and Norton 1982). It is an important concept dealing with questions such as whether a model sequence obtained by identification will converge to the true model as the length of the I/O sequence approaches infinity. It also deals with an issue of whether the parameter set for the so obtained true model is unique. Although this is a theoretical problem, it emphases indirectly the importance of a persistently exciting I/O sequence as a prerequisite for system identifiability, as it is not possible to obtain enough information about the system from its output in the case when I/O sequence is not persistently exciting, in which case not all of the model parameters will converge to their true values no matter how long the duration of the input signal. Simply stated, good input-output records will allow identification of *all* of the parameters of the "true" model. In fact, there is a direct relation between the maximal number of parameters that can be estimated and the quality of the input signal (Mayne 1972).

2.1.5 Some Additional Problems in SI

In addition to the difficulties mentioned above, it is worth mentioning that SI shares some general computational difficulties with other topics in science and engineering. This has to do with whether the problem is ill-conditioned or well-conditioned, and also whether the computational method is numerically stable or unstable (Chen 1984). In the first case, a small change in data will bring large changes in solution, showing sensitivity to data. An example of this would be identification from an impulse response (Audley and Lee 1976). Numerically unstable methods amplify the accumulated errors. This is more likely to happen with complex numerical problems (Šiljak 1978). Conditioning of a problem and numerical stability are two separate problems, and one should always use numerically stable methods if possible. However, the conditioning of a problem is tied to the model used and sometimes we may have to consider different model representations when an ill-conditioned situation occurs (Audley and Lee 1974).

As an example, we can look at what happens with discretization of a continuous-time model. Discretization of a model through z-transform means that the s-plane is mapped into the z-plane in such a way that the left half of the s-plane is mapped to a region within the unit circle centered at the origin of the z-plane, and right half of the s-plane is mapped to the area of the z-plane outside the unit circle. The imaginary axis in the s-plane corresponds to the unit circle in the z-plane. As the sampling frequency increases, the mapped area in the z-plane shrinks around the point (1,0) on the real axis (Rao and Sinha 1991). What this means is that system poles and zeros of the continuous model which may be well apart in the s-plane, now become very close in the z-plane, resulting in inherent numerical difficulties associated with high-sampling rate discrete models (Rao and Sinha 1991, Nunnes and Goodwin 1991). The way of overcoming these difficulties is, for instance, to apply some other discretization technique, like δ-transform, in which case the mapped area in the δ-plane expands with the increase in sampling rate (Nunnes and Goodwin 1991).

Another example would include the problem of identifying the system from its im-

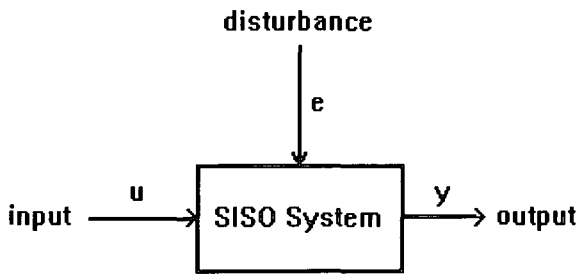


Figure 2-2: SISO system

pulse response. This is a well known ill-posed problem (see Audley and Lee 1974 where it is proposed to circumvent the problem by choosing another representation, the λ-representation).

In this text, all proposed methods are applied to continuous models. This should be viewed only as an illustration of parallel identification methodology, which is, as already mentioned, independent of the particular identification methods, meaning that discrete time and nonlinear models may be used equally well.

2.2 Identification of SISO Systems

A SISO system can be presented as in Figure 2-2. It has one measurable input and one measurable output. It is customary in systems theory to denote as input signals those inputs that are controllable by us. Inputs that are beyond our control are called disturbances. The effect of disturbances on the output is frequently studied within a stochastic framework, as the values of disturbance signals cannot be surely predicted.

There are several ways of describing SISO systems. Both parametric and non-parametric descriptions are used for this purpose (Rubio 1971, Chen 1984, Ljung 1987).

In this section, we will consider only parametric description. There are several parametric descriptions of linear models (Dickinson *et al.* 1974, Chen 1984, Ljung 1987), and the two that will be used in this text are transfer function and state-space description.

2.2.1 Transfer Function Models

It is well known that for continuous, linear, time-invariant system, input and output are linearly related in the complex frequency domain (Papoulis 1977, Lynn 1982). We translate input-output relations from time to frequency domain via, for example, Laplace or Fourier transforms. In the case of discrete model, some or all variables are either sampled value of continuous signals, or are discrete by the nature of the process (Oppenheim and Schaffer 1975, p.7). In this case we move from discretized time to frequency domain via z-transform, δ-transform, etc. Once a suitable transfer function structure is selected, we can take on the task of determining the parameters of such model.

In the case of the model from Figure 2.1 we can write

$$y = f(u, e, t) \tag{2.1}$$

where t is time, while y is output signal sequence, u is an input signal sequence, e is a signal sequence representing the effect of disturbances and f is a suitable operator. In the discrete model case, we can write for a simple linear model where no disturbances are present, that

$$y(t) + a_1 y(t-1) + \ldots + a_n y(t-n) = b_1 u(t-1) + \ldots + b_m u(t-m) \tag{2.2}$$

where expression $u(t-m)$ denotes the value of the signal u at the moment shifted backwards (from the current moment) by m sampling units. Expression 2.2 can be written in a more compact form as

$$A(q^{-1})y = B(q^{-1})u \tag{2.3}$$

where $A(q^{-1})$ and $B(q^{-1})$ are polynomials of order n and m, respectively. The shift operator q^{-1} is defined as $q^{-1}f(k) = f(k-1)$.

To identify a system with model (2.3), we need to estimate coefficients of the poly-

nomials $A(q^{-1})$ and $B(q^{-1})$. These coefficients are called parameters of the model (2.3) and therefore the identification process is reduced to the one of parameter estimation. It is common to refer to these coefficients in a form of the parameter vector

$$\Theta = (a_1\ a_2\ \ldots\ a_n\ b_1\ b_2\ \ldots\ b_m). \tag{2.4}$$

We see from (2.4) that we need to find $n + m$ parameters to uniquely determine the model.

Depending on how we model the disturbances, we can get different transfer function models. Due to the assumption of linearity, output of the system y will consist of the sum of the component due to the input signal u and the component due to the disturbance signal e. The disturbance signal e can be modelled in several ways. One way is to consider it as a white noise (Goodwin and Payne 1977, Ljung 1987, Söderström and Stoiça 1989). White noise is a random process with zero mean and Gaussian distribution. In a discrete framework it is a sequence of independent and identically distributed random variables with mean value equal to zero. In reality, the assumption of independence and Gaussian distribution is rarely fulfilled, but white noise makes for an easier analysis than coloured noise. Another way of modelling the disturbance signal is to describe it as a uniform, but bounded quantity. This is justified by the fact that the magnitude of a disturbance signal is limited by some upper boundary value.

The effects of disturbances are often modelled by adding them directly to the output. Sometimes signal e is considered as passing through some filter, and the resulting signal is then added to the output.

Some of the models that are in common usage are described below.

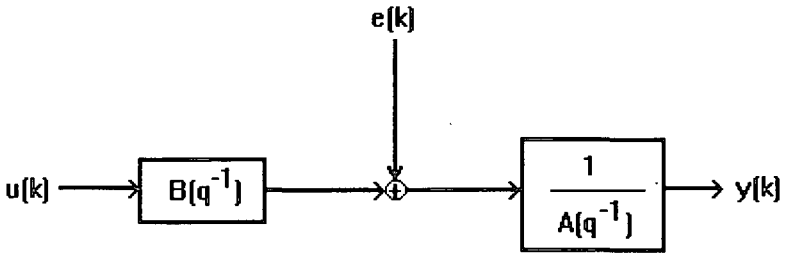


Figure 2-3: ARX model

Auto-Regressive Model With Exogenous Variable (ARX)

The ARX model is presented in Figure 2-3.. Here, input $u(t)$ is referred to as the exogenous variable. It is described as

$$A(q^{-1})y(t) = B(q^{-1})u(t) + e(t). \tag{2.5}$$

In this case disturbances are modelled in such a way that the disturbance term is added directly to the output of the model. The parameter vector is the same as the one given by (2.4).

Finite Impulse Response (FIR) Model

The very important special case of ARX model is the FIR model. The general form of the difference equation relating input sequence $u(n)$ and output sequence $y(n)$ is

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{r=0}^{M} b_r u(n-r). \tag{2.6}$$

The finite impulse response model is obtained for the case $N = 0$, for which (2.6) becomes

$$y(n) = \frac{1}{a_0}\left[\sum_{r=0}^{M} b_r u(n-r)\right]. \tag{2.7}$$

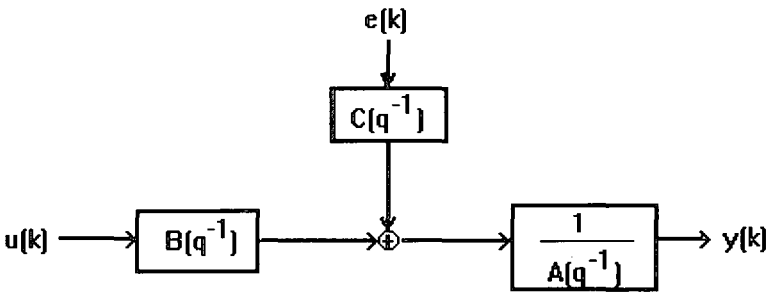


Figure 2-4: ARMAX model

Because this equation is not recursive, the response sequence $y(n)$ will always be of finite duration, as opposed to the case when N from (2.6) is greater than zero, in which case the response is of infinite duration (Oppenheim and Schafer 1975, Božić 1979, Lynn 1982, Vanlandingham 1985).

Autoregressive Moving Average With Exogenous Variable (ARMAX) Model

The ARMAX model is more general than the ARX model. It enables greater flexibility with regard to the description of both input and disturbance terms and has the form

$$A(q^{-1})y(t) = B(q^{-1})u(t) + C(q^{-1})e(t). \tag{2.8}$$

The parameter vector is

$$\Theta = (a_1\ a_2\ \ldots\ a_n\ b_1\ b_2\ \ldots\ b_m\ c_1\ c_2\ \ldots\ c_p). \tag{2.9}$$

The ARMAX model is presented in Figure 2-4.

Other combinations of polynomials would bring more complex model structure. However, it can be seen that in all cases polynomial A is a common factor at the point of output for both input and disturbance. If we compare the equations (2.3) and (2.5), we can see that disturbance term e acts like an "error" w.r.t. the equation given by (2.3),

and these models are often called equation error models (Ljung 1987).

There is another class of transfer function models called output error models (Ljung 1987). In this case, the disturbance (or "error") term directly affects the output, thus acting as an output error. For such a model, we can write analogously to (2.1), that

$$y = f(u,t) + e. \tag{2.10}$$

In equation (2.10) the operator $f(u,t)$ will represent the model and the term e represents the difference between the measured (system) output sequence and the model output sequence. Generalization of the output-error model class is the so-called Box-Jenkins model (Ljung 1987, Söderström and Stoiça 1989), which is given as

$$y(t) = \frac{B(q^{-1})}{F(q^{-1})}u(t) + \frac{C(q^{-1})}{D(q^{-1})}e(t). \tag{2.11}$$

General Model Structure

By combining the general case for output error and equation error models, a general model structure for transfer function models is obtained as

$$A(q^{-1})y(t) = \frac{B(q^{-1})}{F(q^{-1})}u(t) + \frac{C(q^{-1})}{D(q^{-1})}e(t) \tag{2.12}$$

Obviously, by manipulating polynomials in this general case, all of the special cases can be derived. The model given by (2.12) is rarely used, as it contains a large number of parameters. We would often prefer to start with something like an ARX model and see if such a model would yield satisfactory behaviour.

2.2.2 State-Space Models

State-space (or state-variable) representation for linear, continuous, time-invariant models is of the form

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t). \end{cases} \tag{2.13}$$

In this representation, $\mathbf{x}(t)$ is the n-component state vector, $\mathbf{u}(t)$ is a p-component input vector and $\mathbf{y}(t)$ is a q-component output vector. Matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ are of dimensions $n \times n$, $n \times p$, $q \times n$ and $q \times p$, respectively. The equation given by (2.13) is said to be n-dimensional. This representation can describe even those states that are not directly reachable from the input, or whose effect is not seen at the output. Therefore, this representation is more general than the transfer function representation. Using this representation it is also possible to effectively describe time-varying systems (Chen 1984). More details for this representation can be found in any introductory text in control theory, for example Fortmann and Hiltz (1977).

2.2.3 Parameter Estimation

Once we decide on the appropriate model structure, all we need to do is to determine the parameter vector. In other words, we try to fit the model to the measured input-output sequences. This procedure is depicted in Figure 2-5. We see that input into the procedure is a model from a class and I/O sequence, and the output is a parameter vector, defining the resulting model. We recall once again that the identification process will yield a unique set of parameters in a given model class if the system whose I/O sequence we use is identifiable by that sequence. It was mentioned in Chapter 1 that the most popular estimation techniques are those based on the least-squares and maximum likelihood methods. First we deal with the least-squares and maximum likelihood methods, and subsequently briefly describe parameter estimation using optimization techniques for minimizing an arbitrary error function.

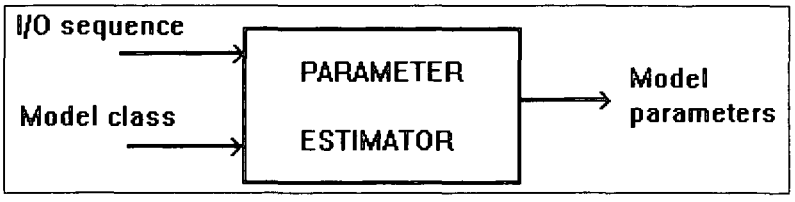


Figure 2-5: Parameter estimation procedure

Least-Squares Method

Least-squares method (Goodwin and Payne 1977, Hsia 1977, Whittle 1983) is the most widely used method for parameter estimation, and its application goes far beyond system identification (Whittle 1983). To illustrate the method, suppose that a line has to be drawn through a set of given points in plane (interpolation problem) in such a way that it is as close as possible to those points. To be able to do this we need to define what we mean by "close". If we define this closeness (interpolation error) as the sum of the squares of the distances (in the usual geometrical sense of distance between a line and a point in a plane) between each point from the given set and the line, then it is necessary to find parameters of the line such that this error is minimal.

Suppose that variable y linearly depends on n inputs $x_1, x_2, ..., x_n$

$$y(t) = \theta_1 x_1(t) + \ \ldots \ + \theta_n x_n(t). \tag{2.14}$$

We want to obtain estimates for constant parameters $\theta_1, \theta_2, ..., \theta_n$, so that the modelling error ε will be minimal in the least squares sense. Derivation of the least-squares method can be found in Appendix A. Here we state the final result for the least squares estimator (LSE), obtained by using m measurements:

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.15}$$

In this relation $\hat{\theta}$ is the required n-component parameter vector estimate, $\mathbf{y}$ is the m-

component measured output vector, and matrix $\mathbf{X}$ is an $m \times n$ matrix

$$\mathbf{X} = \begin{bmatrix} x_1^1 & x_2^1 & . & . & . & x_n^1 \\ x_1^2 & x_2^2 & . & . & . & x_n^2 \\ . & . & & & & . \\ . & . & & & & . \\ . & . & & & & . \\ x_1^m & x_2^m & . & . & . & x_n^m \end{bmatrix} \tag{2.16}$$

containing the inputs in m measurements, where x_i^j denotes j-th measurement of i-th input.

It is worth investigating some basic statistical properties of LSE. This is important as we always operate on data that is more or less corrupted by noise. We show here that LSE is an unbiased estimate.

An estimate $\hat{\theta}$ is said to be unbiased if its statistical expectation $E(\hat{\theta})$ is equal to θ, where θ is a true vector of parameters (Söderström and Stoiça 1989). This can also be stated in the following way. When the number of measurements is increasing, the estimate $\hat{\theta}$ is becoming closer to its true value θ, so that

$$\lim_{m \to \infty} \hat{\theta} = \theta. \tag{2.17}$$

Let the vector $\boldsymbol{\varepsilon}$ represent an equation error of an estimate (see Appendix A). This error exists due to the disturbances in the I/O signals as well as the model inaccuracies. If this disturbance is taken to be white noise, then the vector $\boldsymbol{\varepsilon}$ representing this disturbance is a stationary random vector with zero mean, i.e. its expectation is zero. Furthermore, $\boldsymbol{\varepsilon}$ is assumed to be uncorrelated with $\mathbf{y}$ and $\mathbf{X}$ (Hsia 1977). It is now necessary to show that under this condition (2.17) will hold. To show that, if $\mathbf{y}$ from Eq. A.7 (Appendix

A) is substituted in (2.15) and then we take the expectation of both sides, we get

$$E(\hat{\theta}) = E(\theta) + E((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)E(\boldsymbol{\varepsilon}). \tag{2.18}$$

As the expectation of noise vector $E(\varepsilon)$ is zero, the second term in (2.18) is also zero, and (2.17) follows directly. Other statistical properties of LSE can be found, for instance, in Hsia (1977), Goodwin and Payne (1978) and Whittle (1983). For application of LSE in the presence of model uncertainty, see Bai and Nagpal (1995). The consistency of LSE method when used for identification is discussed in Ljung (1977).

The LSE method just described is sometimes called ordinary LSE, as all components of $\boldsymbol{\varepsilon}$ have the same weight. One would then guess that there exists a more general LSE method in which we can assign different weighting factors to the error components. This is particularly useful if we are tracking time-variable dynamics, where more recent error components are associated with bigger weighting factors (Hsia 1977). Multi-output LSE is very similar to the single output LSE and derivation of the method can be found in Hsia (1977).

In this work optimization methods will be used for the parameter estimation. The main reason for the introductory treatment of LSE is because some methods for speeding up the identification procedures can be derived from LSE, including those suitable for parallel processing. Such methods can be viewed, to an extent, as the alternative to our new approach of hierarchical identification. However, our approach has some distinct advantages that will be discussed later.

Maximum Likelihood Method

This method considers measured outputs as results of a stochastic process. The aim now is to estimate the parameters in a such a way that the probability of a given outcome (in this case measured output vector values) is maximized. To do this, we regard ε as a random process and assume that its probability density function $p(\varepsilon, \theta)$ is known. We

now look for the estimate of θ that will maximize the likelihood of those error samples. The issue is how to define the likelihood function that we need to maximise. This function is usually taken as a joint probability density function of each error sample

$$f(\varepsilon, \theta) = \prod_{i=1}^{m} p(\varepsilon_i, \theta) \tag{2.19}$$

providing that ε_i are uncorrelated. The best estimate for θ is given then as

$$\hat{\theta}_{ml} = \max_{\theta} f(\varepsilon, \theta). \tag{2.20}$$

The form of (2.20) will obviously depend on the probability density function to be used in (2.19). For details see Ljung (1987).

Optimization Methods

In a deterministic framework, LSE is certainly the most popular and easy to use method. We have seen that it also exhibits good statistical behaviour. In many cases, though, we may wish to select some other, more suitable error criterion. In that case we have to minimize that particular error function, and LSE is then no longer applicable. In this case we apply other optimization procedures. Roughly speaking, the goal of optimization methods is to minimize an arbitrary function which depends on certain parameters by tuning the values of these parameters. Optimization methods are very powerful indeed, and are widely used in many areas of mathematics and science (Fletcher 1980). They are an indispensable tool for many types of identification and they will be employed extensively in this study for black-box identification when system structure and statistical nature of the signals are not known.

There are two categories of optimization procedures. One category comprises gradient methods, in which we descent towards the (local) minimum following the gradients of the function to be minimized (Burley 1974, Fletcher 1980). The other category includes the so called non-gradient techniques where information about direction of search for

the minimum is obtained via a numerical experiment (Burley 1974). The most popular method in this category is the simplex method, but there are also other, less efficient methods, like bisection. Gradient methods are in general superior to non-gradient methods, but the simplex method will in certain cases do just as well, particularly if the function exhibits some particularly nasty behaviour.

2.2.4 Examples

In what follows several examples are given in order to illustrate some of the problems that appear in practice in identification of continuous time models.

Example 2.1 The first example relates to the identification of a process whose step response is given in Figure (2-6). This looks like the response of a first order system. In such a case, where signal to noise ratio is high, we can establish the values of constants τ, K and L in the input-output model

$$\frac{Y(s)}{U(s)} = \frac{K}{\tau s + 1} e^{-Ls} \tag{2.21}$$

almost from geometrical considerations, as ratio K/τ is equal to the slope of the tangent of unit step response at point $t = 0$ and K itself can be determined from the steady-state response. Dead time L is also easy to estimate from the step response graph.

If the signal to noise ratio is low, then it is difficult to use this reasoning to obtain the model (2.21). We have to take noise into consideration, and in this case we may not be able to estimate the parameters properly by similar considerations.

Model (2.21) can be also used instead of higher order, minimum-phase models that do not exhibit any oscillatory behaviour in their response. Large industrial processes (like chemical processes) are often reasonably successfully described by the model (2.21). More details can be found in Aström and Hagglund (1988).

It should be noted that such, geometrical approach to SI is non-optimization based.

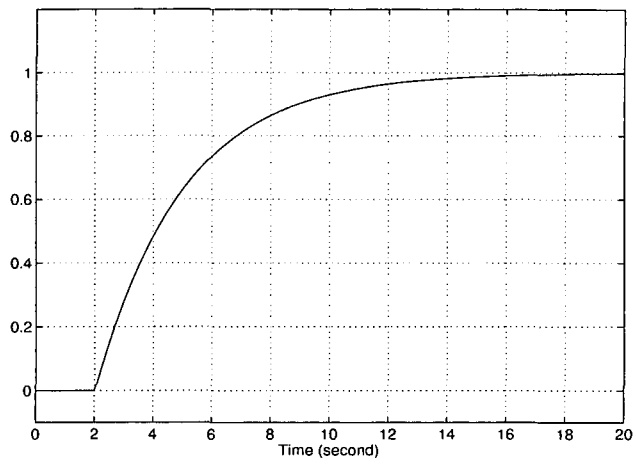


Figure 2-6: Step response for a first order model

This is not the only non-optimization based method for estimating model parameters. For example, another interesting method of a similar nature is given in Yuwana and Seborg (1982).

Example 2.2 We shall now illustrate the SI procedure using another example. Throughout this research, modelling and simulations were performed on *Matlab* numeric computation software, and *Simulink*, the software for simulation and integration of differential equations. In order to illustrate SI methods, we create a model in *Simulink*, whose input-output pairs are then considered as measured input-output sequences. In this case, deterministic, linear, continuous, time-invariant models will be used to fit this "measured" sequence. The model for this example is presented in Figure 2-7. We assume that both input and output sequences are corrupted by noise. Furthermore, we will model this noise as coloured noise (see Figure). Finally, there is a dead-time of 1 second in the system. I/O sequence is obtained from signals at the points A and B when the unit step signal is applied.

Based on this I/O sequence, we will try to identify a model that would adequately capture this I/O sequence and that will pass the validation test. Input and output

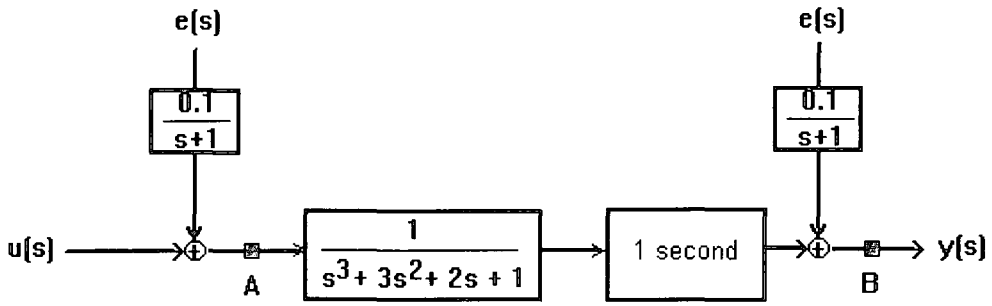


Figure 2-7: Model used to generate I/O sequence

signals are presented in Fig. (2-8). We are now looking for the transfer function that will properly describe the system dynamics. The first issue is model order as we "do not know" the model order in advance (see discussion on model order in Section 1). First, we observe that this system cannot be very successfully modelled by a first order model, as it contains damped oscillatory component. Therefore, the lowest model order we shall start with will be 2. We now look for the model of the form

$$Y(s) = G(s)e^{-sL}U(s) \tag{2.22}$$

where $G(s)$ is the transfer function without dead-time (where G is to be determined) and L is the apparent dead time of the system which also has to be estimated. To estimate the parameters of the model (2.22) we will use an optimization procedure. The function that we will minimize is a measure of difference between model output and measured output. This approach is called, as mentioned previously, model error approach. This error function is partly a function of the model parameters and partly a function of noise, similar to the ARMAX model or some of its derivatives. The two most frequently used error functions are the integral square error (ISE)

$$ISE = \sum_{i=1}^{m}(y_i^m - y_i)^2 \tag{2.23}$$

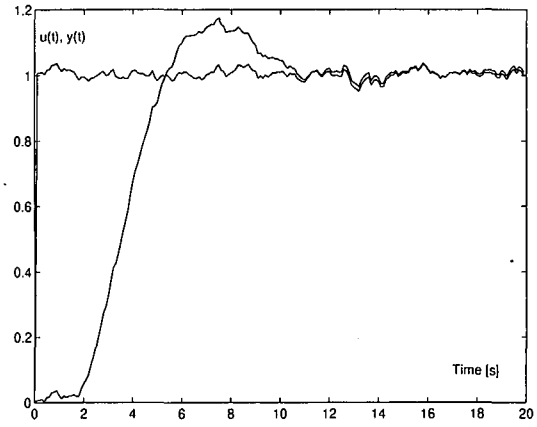


Figure 2-8: Step response of the system in Example 2

and the integral absolute error (IAE)

$$IAE = \sum_{i=1}^{m} abs(y_i^m - y_i) \tag{2.24}$$

where m is the number of measurements, y_i is the i-th component in measured output vector and y_i^m is i-th component of model output vector. We will be using $\sqrt{ISE}$ as minimizing criterion, although the error of the model will sometimes in this study also be expressed in terms of IAE. As we cannot explicitly relate the effect of the noise on the output, we will, instead of considering the whole process as stochastic, try to fit our model to the system's I/O data in such a way as to average the output error due to the noise. This is reasonable when disturbances have zero mean. As always in the case of optimization problems, we need to choose a set of initial values for parameters and the optimization method. In this particular case we use the simplex optimization algorithm. Integration step is 0.1[s]. The result is the following model:

$$y(s) = \frac{0.4013}{s^2 + 0.6547s + 0.4013} e^{-1.4137s}. \tag{2.25}$$

The meaning and importance of the dead-time term will be discussed later. Here we

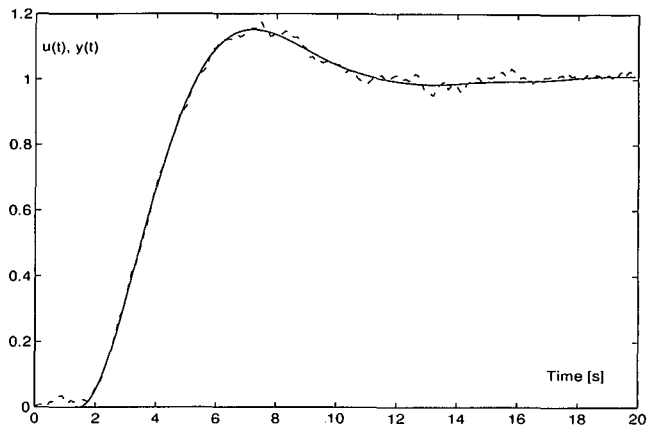


Figure 2-9: Step response of the system and the model

only observe that the estimated dead-time is somewhat bigger than the one in the model given in Fig. 2-7 used to generate I/O sequences. The error of the obtained second-order model, expressed in $\sqrt{ISE}$, is 0.1891 and the number of floating point operations that were executed during the optimization process was 235404. The model and system output are drawn in Fig. 2-9. The model error (difference between the model and the systems output) is shown in Fig. 2-10.

We now want to validate the obtained model. This can be done in the following way. We simulate the response of the model to some arbitrary input signal and compare the output of this model to the output of the original model for the same input. Both the disturbances at the input and output of the original model used to generate the outputs are kept. Responses of the system and this model to white noise are presented in Figure 2-11.

The $\sqrt{ISE}$ between model and system output is now 0.2274. This figure, drawing (2-11) and steady-state response give us confidence in the model described by (2.25).

Now we want to see if increasing the model order will bring us any benefit. We shall use the same model structure as in the previous case, the one described with (2.22). The

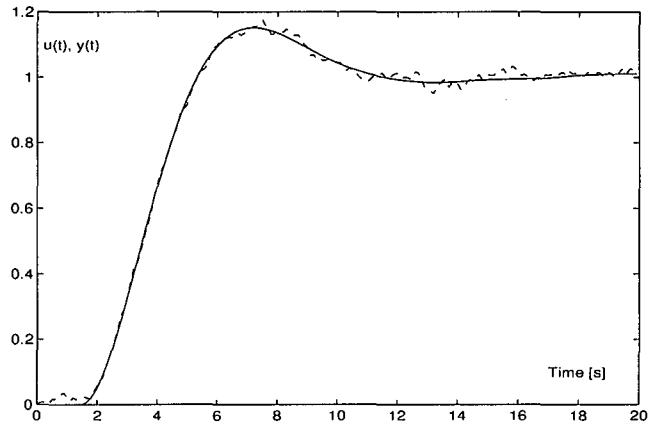


Figure 2-9: Step response of the system and the model

only observe that the estimated dead-time is somewhat bigger than the one in the model given in Fig. 2-7 used to generate I/O sequences. The error of the obtained second-order model, expressed in $\sqrt{ISE}$, is 0.1891 and the number of floating point operations that were executed during the optimization process was 235404. The model and system output are drawn in Fig. 2-9. The model error (difference between the model and the systems output) is shown in Fig. 2-10.

We now want to validate the obtained model. This can be done in the following way. We simulate the response of the model to some arbitrary input signal and compare the output of this model to the output of the original model for the same input. Both the disturbances at the input and output of the original model used to generate the outputs are kept. Responses of the system and this model to white noise are presented in Figure 2-11.

The $\sqrt{ISE}$ between model and system output is now 0.2274. This figure, drawing (2-11) and steady-state response give us confidence in the model described by (2.25).

Now we want to see if increasing the model order will bring us any benefit. We shall use the same model structure as in the previous case, the one described with (2.22). The

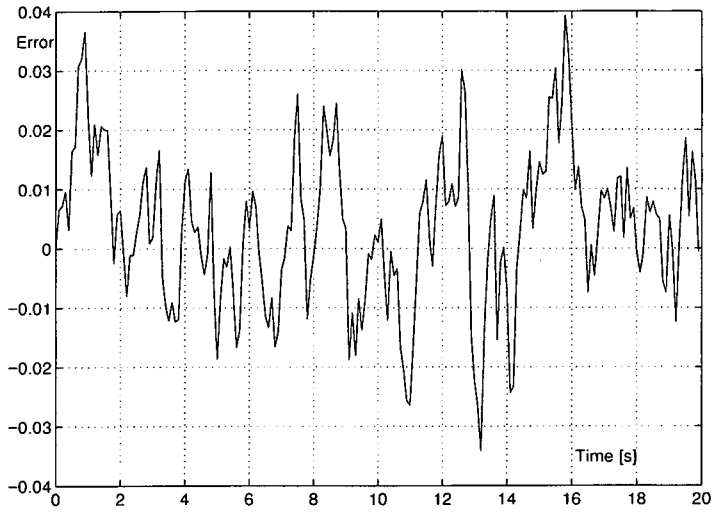


Figure 2-10: Output error for the second order model

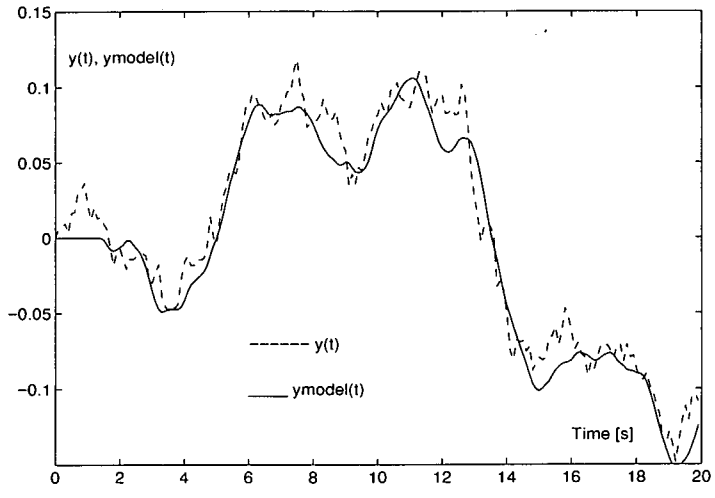


Figure 2-11: Comparison of system and model outputs for white noise

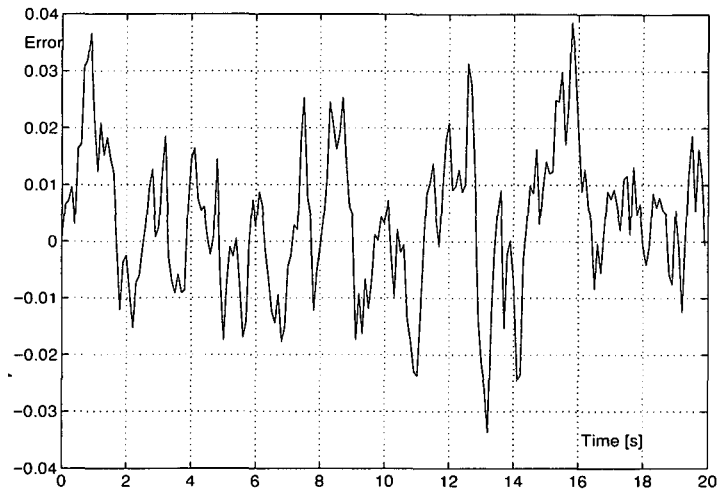


Figure 2-12: Output error for the third-order model

model order is now 3, and using the same approach for parameter estimation we get

$$y(s) = \frac{1.2053}{s^3 + 3.5911s^2 + 2.3298s + 1.2053} e^{-1.1133s}. \tag{2.26}$$

We see now that apparent dead-time of the system is closer to the true value of 1. Model error is shown in Fig. 2-12. However, $\sqrt{ISE}$ has dropped only marginally to 0.1868, which is only about 1 percent better, but the number of floating point operations has risen to 1484353, which is a more than sixfold increase. Now we proceed with model validation in the same way as in the second-order model case. We assume that disturbances in the original model are still present and unchanged, so that identification will again be performed on noisy input-output data. This time we get $\sqrt{ISE}$ as 0.2142, which is also only marginally better than for the second-order model.

The question of whether we should have stopped at a model order of two in this particular case may be affirmative as the integral absolute error was already relatively small for the second-order model, so there was no hope of reducing it drastically. But, in our search for a more precise model, we could have carried on, and obtained more precise, higher-order models.

Another issue is whether more precise models could be obtained by increasing the model order over 3. In this case one can argue that, since the "true" model order is three, no improvement is possible. But, as input-output data is corrupted by the noise, higher order models may capture a portion of the noise dynamics better. However, one must be cautious when increasing the model order, firstly, because of disproportional increase in numerical difficulties and, secondly, because such models may have very close poles and zeros in the resulting transfer function, meaning that model order increase is then not particularly beneficial. Such poles and zeros may cancel, thus making the effort of obtaining them completely unnecessary. In this particular example, the "true" model is indeed a third-order model, and any higher order models would not contain any additional useful information about the system, even if their corresponding model error is smaller than the one in the third-order model case. As a rule, the number of free parameters should be as small as possible. This principle is known as parsimony (Söderström and Stoiça 1989). In general, it should be noted that trying to capture the effects of noise with high-order deterministic methods is often unnecessary as all that is achieved is minimization of modelling error for that particular input and output pair. For some other signals, higher order models may have no significant advantage over good, relatively low order models.

In conclusion, we can say that modelling by high-order models in the presence of noise in I/O sets is not particularly beneficial. Also, the computational effort increases disproportionately compared to the increase in model order (i.e. number of parameters).

Example 2.3 The third example is taken from *The Simulink User's Guide* (The MathWorks Inc. 1993). The model that we will use to generate the I/O sequence represents a three-channel autopilot system. We obtained the I/O sequences from the unit step response of such system, which is presented in Fig. 2-13. There are two reasons why this example is important. Firstly, the original model is of order 74, which is quite high. Secondly, we will be able to assess the importance of an interactive identification approach,

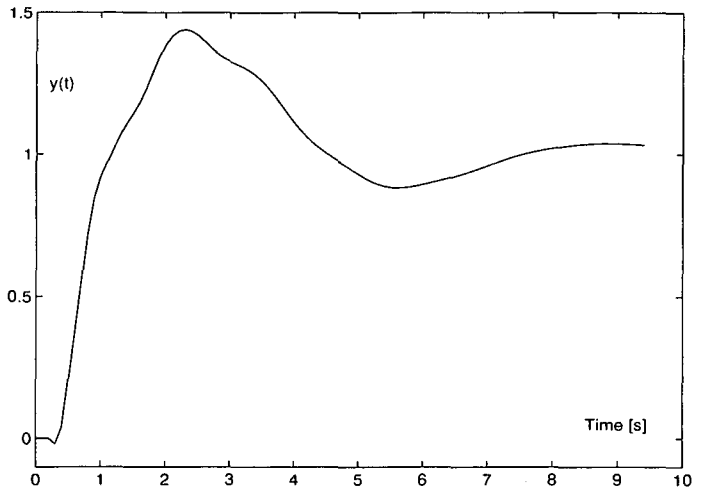


Figure 2-13: Step response for autopilot model

one that requires human intervention.

To find the model that would fit this I/O sequence, we will use the same approach as in Example 2.2. The input-output sequences are 9.4[s] in duration. We see that this response has some "humps" at approximately 1[s], 2[s] and 3.3[s] and it reaches the steady state at the end of the observation (test) interval. The presence of humps indicates that the behaviour of this system is richer than the one exhibited by the system of order 2. This means that it will not be possible to capture the system dynamics well by a second-order model. But, as we wish to compare the cost of obtaining a model and its merit, we may as well start with the model of order 2. Comparison between the system (in our case the original model of order 74) and the (identified) model outputs for models of order 2, 3 and 4 are shown in Fig. 2-14, 2-15 and 2-16. Model error is shown for models of order 3, 4, 5, and 6 in Fig. 2-17, 2-18, 2-19 and 2-20.

From these figures it can be seen that models of order 2 and 3 do not capture the system modes corresponding to these "humps" mentioned above, but the fourth-order model does so. Therefore, we can say that in this case models of order 2 and 3 are not good enough, and that the fourth-order model fits the I/O sequence much better. We can

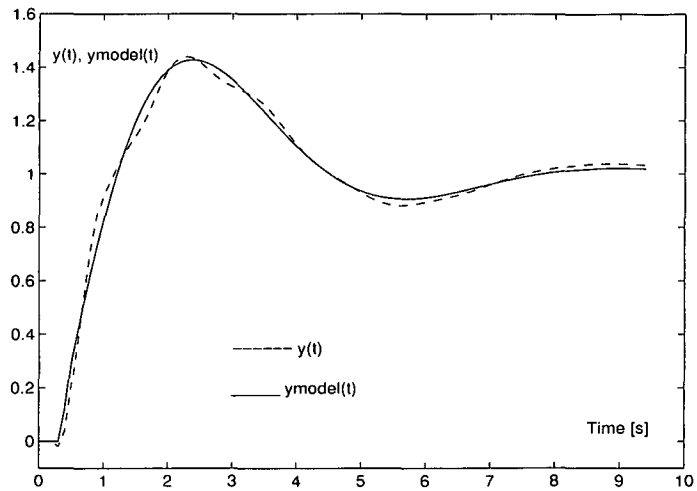


Figure 2-14: Step response of second-order model

order	init. values	$\sqrt{ISE}$	IAE	Kflops	dead-time	validation
2	1	0.2950	1.94	1939	0.00	7.5255
3	3	0.1854	1.30	4985	0.18	5.7731
4	3	0.0824	0.59	15503	0.09	2.3976
5	4	0.0809	0.57	48253	0.07	2.3834
6	11	0.0691	0.45	188106	0.11	2.3744

Table 2.1:

here conclude that sometimes purely numeric criteria for model evaluation are not good enough, and that some sort of visual judgement is often very useful. Model validation is performed by comparing the responses of the original model and the model obtained in the identification, to the white noise. This is expressed in terms of IAE.

All of the obtained models are given in Appendix B. The identification results are summarized in Table 2.1.

It is very interesting to see how disproportionately computational effort increases as the system order goes up. The number of floating point operations (as measured by Matlab's *flops* command) increases almost exponentially with the increase in the size

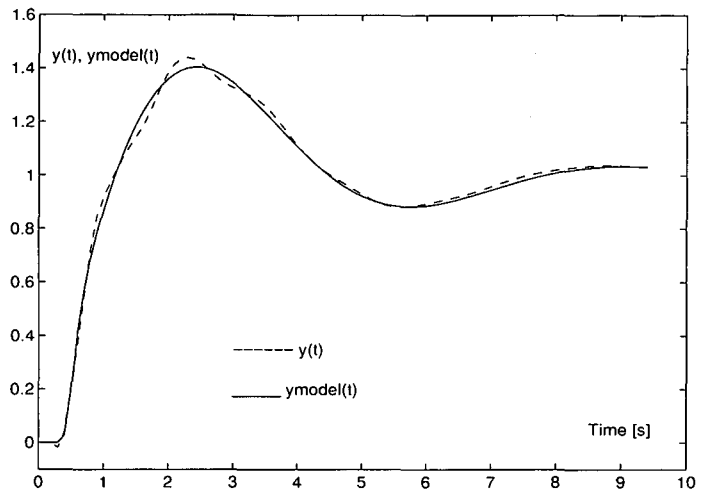


Figure 2-15: Response of the third-order model

of parameter vector. The difficulties in the choice of the initial values in parameter estimation are also noticeable. Therefore, even relatively small gains in model accuracy are paid for very dearly in terms of computational difficulties, both with respect to the number of floating point operations and number of choices for the initial values for parameters. But what is perhaps the strangest result is that models of order 5 and 6, which were very costly to obtain, almost fail to rate better on validation test than the model of order 4! To explain this, we recall that the model was obtained from the sequence of 9.4[s] in duration, and by the end of this period output was already in the steady state. Therefore, the model error will consist of model error in the transient part and model error in the steady-state part. Some models may minimize the steady-state error at the cost of transient error and the longer the I/O sequence, the more superior this model would appear to be. But, if we compare the model output for some broadband input signal, such as white noise, where steady-state error has little effect on the total model output error, then the model which best fits the transient will rate as the best model.

It is worth noting that a lower signal-to-noise ratio can mask much of the system

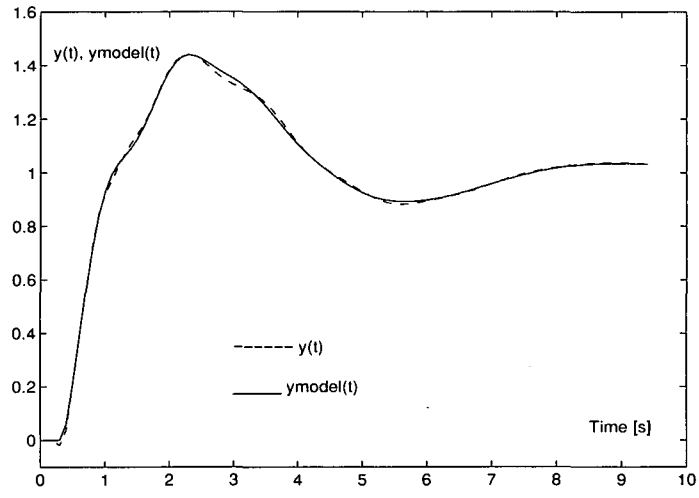


Figure 2-16: Response of the fourth-order model

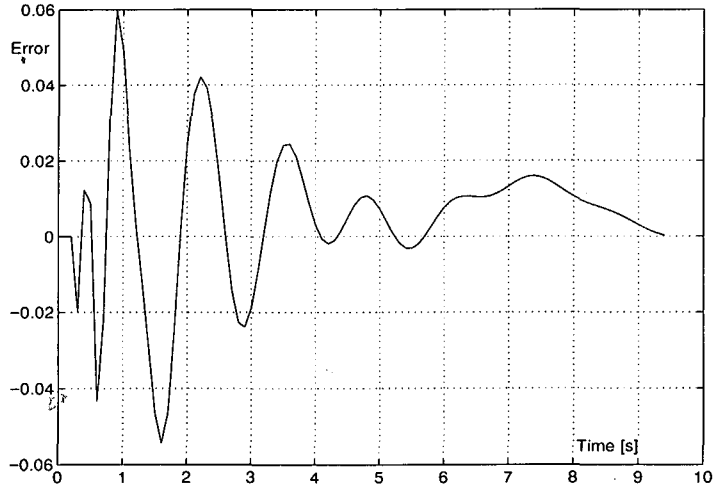


Figure 2-17: Output error for a third order model

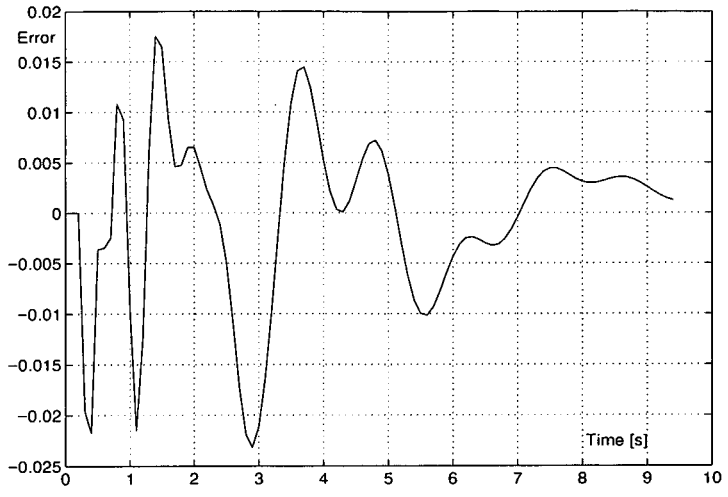


Figure 2-18: Output error for a fourth-order model

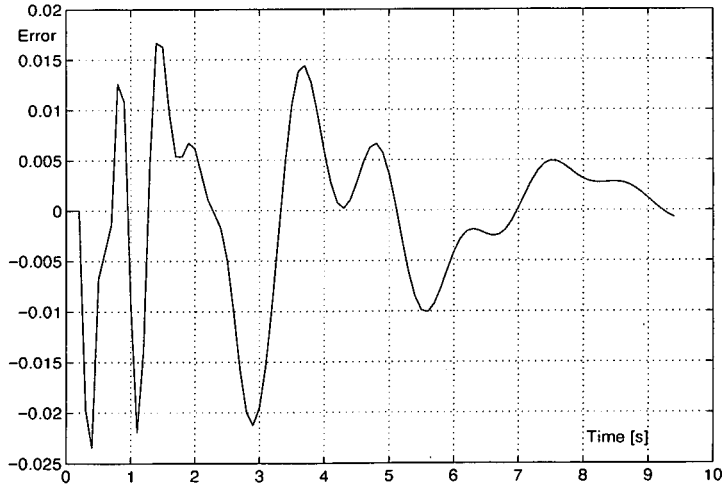


Figure 2-19: Output error for the model of order 5

functions. This representation describes only input and output characteristics of the model.

State-Space Model Representation

State-space (or state-variable) representation for linear, continuous, time-invariant models is of the form

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t). \end{cases} \tag{2.29}$$

In this representation, $\mathbf{x}(t)$ is the n-component state vector, $\mathbf{u}(t)$ is input vector and $\mathbf{y}(t)$ is the output vector. Matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ are of dimensions $n \times n$, $n \times p$, $q \times n$ and $q \times p$, respectively. From given time-invariant state-space equations, an equivalent (zero-state) transfer function matrix representation can be obtained as

$$\mathbf{Y}(s) = [\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}]\mathbf{U}(s). \tag{2.30}$$

The transfer function matrix itself is given by

$$\mathbf{G}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}. \tag{2.31}$$

The inverse problem, the one of finding state-space equations whose equivalent transfer function matrix is equal to the given transfer function matrix is called realization of transfer function matrix (Chen 1984, p. 155). It should be noted here that realizations of state-space equations may be done not only from transfer function matrices or, as already mentioned, polynomial matrix description, but also from input-output data (Buddin, 1971). Although every proper transfer function matrix is realizable (Chen 1984), the way of obtaining this realization is by no means as simple as in the inverse case, where the transfer function matrix is simply given by (2.31). An important point is that state-space descriptions obtained in this way will only be equivalent to the given transfer function matrix if the system is initially relaxed. Furthermore, these realizations are not

unique. This is due to the fact that there is an infinite number of state-space forms that describe the same model. Therefore, state-space models are not in general identifiable, as the parameterisation used to build the resulting model is not unique. This should not be regarded as an insurmountable obstacle, as we are more interested the existence of some parameterisation which fits the input-output vectors, rather than its uniqueness (Glover and Willems 1974). Because of such a wide choice of different parameterisation, the question that arises is whether some of them are better (in some way) than the others. A very important class of realizations are minimal realizations. The feature of these realizations is that they can be described by a minimal number of parameters in their model class, defined by the model order and the number of inputs and outputs. As such, they are useful for identification of multivariable systems.

Identification of Multivariable Systems

As already mentioned, identification of multivariable systems poses many more difficulties than the single-variable case. Essentially, identification of MIMO systems is analogous to the identification of SISO systems. However, the arsenal of difficulties that the identification problem has in the multivariable case is much larger than the one in the SISO case. This justifies separate study of the problem for at least two reasons. Firstly, the choice of parameterisation is greater than in the SISO case. Secondly, numerical difficulties are much larger. Order reduction techniques do not successfully lend themselves to the multivariable case as the model order is not the only characteristic of a multivariable model. Number of inputs and outputs is also a fundamental characteristic of the multivariable model. In Chapter 4, decomposition schemes allowing for the new parallel identification approach for the multivariable case will be developed. Before that, some common identification approaches for multivariable systems are presented briefly, with the aim of comparing their benefits with the hierarchical identification method that will be introduced. More on this can be found in abundant literature on the subject. A survey of discrete identification methods can be found in Young and Wang (1987).

Identification of a continuous multivariable model can be found in Boje (1991). Since most of the identification methods are developed for discrete-time models, conversion methods for obtaining continuous models from discrete models are developed, see for example, Bingulac and Cooper (1991), Sinha and Lastman (1991). In this text continuous multivariable models will be used. The industrial application examples of identification for multivariable systems can be found in, for example, Backx and Damen (1992) and Hallager *et al.* (1984).

2.3.2 Identification of Transfer Function Matrix

In this section, identification methods based on input-output descriptions of the system are described. It is assumed that input-output records exist and that this data is sufficiently good to yield a consistent estimate of model parameters. The system under observation is assumed to have p inputs and q outputs. Before identification of the transfer function matrix is described, determination of the impulse response matrix will be discussed.

Identification of Impulse Response Matrix

It is well known that a linear, initially relaxed system is completely described by its impulse response, in which case its input-output relation is given by

$$y(t) = \int_{-\infty}^{\infty} g(t,\tau)u(\tau)d\tau. \tag{2.32}$$

In this equation, u is input signal, $y(t)$ is the value of the output at t and $g(\cdot,\tau)$ is impulse response of the system (response of the system to Dirac's delta function). The expression of the form $g(t,\tau)$ simply denotes response of the system at the instant of time t due to the impulse signal applied at moment τ. Because all physical systems are causal, the response cannot occur before the system is excited, thus requiring that $\tau < t$. As any input signal can be considered in a limit process as superposition of impulse signals, the

integral (2.32) is called the superposition integral because it is obvious that output at any given moment t is the superposition of all impulse responses that occurred prior to that moment. Therefore, (2.32) is more often written in the form

$$y(t) = \int_{-\infty}^{t} g(t,\tau)u(\tau)d\tau \tag{2.33}$$

or

$$y(t) = \int_{t_0}^{t} g(t,\tau)u(\tau)d\tau \tag{2.34}$$

if the system is initially relaxed at the moment t_0. In the case of time-invariant systems, this becomes

$$y(t) = \int_{t_0}^{t} g(t-\tau)u(\tau)d\tau = \int_{t_0}^{t} g(\tau)u(t-\tau)d\tau \tag{2.35}$$

Integrals (2.32) - (2.35) are applicable for the SISO case. This concept can be extended trivially to the multivariable case if the impulse response matrix is formed:

$$\mathbf{G}(t,\tau) = \begin{pmatrix} g_{11}(t,\tau) & g_{12}(t,\tau) & . & . & g_{1p}(t,\tau) \\ g_{21}(t,\tau) & g_{22}(t,\tau) & . & . & g_{2p}(t,\tau) \\ . & . & . & . & . \\ . & . & . & . & . \\ g_{q1}(t,\tau) & g_{q2}(t,\tau) & . & . & g_{qp}(t,\tau) \end{pmatrix} \tag{2.36}$$

in which case input-output relation is

$$\mathbf{y}(t) = \int_{t_0}^{t} \mathbf{G}(t,\tau)\mathbf{u}(\tau)d\tau. \tag{2.37}$$

Terms $g_{ij}(t,\tau)$ from (2.36) denote impulse response of i-th output at the moment t due to the impulse applied to the j-th input at the moment τ when all other inputs are relaxed. Input vector has p components and output vector has q components. Matrix $\mathbf{G}(t,\tau)$ can be obtained, at least in principle, by applying an impulse to input terminal j, while other input terminals are all relaxed. By measuring the outputs, impulse responses between

j-th input and i-th output $g_{ij}(\cdot, \tau)$ for $i = 1, 2, \ldots, q$, are determined. The same is then repeated for all other inputs. This procedure is mainly of theoretical importance. There are reasons for this. Firstly, the impulse signal is impossible to generate. Secondly, even if we manage to approximate the impulse signal with very strong and short pulse signal, the equipment (system) may be damaged by this sudden influx of energy. Thirdly, the experiment assumes that the system is completely available for the identification procedure which often may not be the case, particularly in the case of large-scale systems.

Identification of Transfer Function Matrix

The transfer function matrix can be estimated directly by identifying the transfer functions from each input to each output. This means that there are $p \cdot q$ transfer functions to be estimated. This can be achieved in very much the same way as with identification of SISO models, by applying a test signal (for instance, unit step) to a particular input and measuring responses in each of the outputs. This approach requires p successive experiments to be performed. It will also require that the system be completely available for identification experiments.

A more realistic approach is to estimate transfer functions from the plant under operating conditions through a process known as random signal testing (Davies 1970). The procedure is based on correlation functions (Papoulis 1977, O'Flynn 1990). Input and output of a linear model are related in the time domain as in (2.35). On the other hand, cross-correlation between input signal $u(t)$ and output signal $y(t)$ is given as

$$r_{uy}(\tau) = \lim_{T \mapsto \infty} \frac{1}{2T} \int_{-T}^{T} u(t) y(t + \tau) dt. \tag{2.38}$$

The input (test) signal is assumed to be periodic with period T. In order to evaluate the response $y(t)$, we rewrite (2.35) in the form

$$y(t) = \int_{-\infty}^{\infty} g(s) u(t - s) ds. \tag{2.39}$$

By substituting y from the expression (2.39) into (2.38), we get

$$r_{uy}(\tau) = \lim_{T \mapsto \infty} \frac{1}{2T} \int_{-T}^{T} u(t) \int_{-\infty}^{\infty} g(s) u(t + \tau - s) ds dt \tag{2.40}$$

which becomes

$$r_{uy}(\tau) = \int_{-\infty}^{\infty} g(s) \left[\lim_{T \mapsto \infty} \frac{1}{2T} \int_{-T}^{T} u(t) u(t + \tau - s) dt \right] ds. \tag{2.41}$$

The expression in brackets is nothing else but an auto-correlation function of the input signal with time shift of $\tau - s$, so we finally get

$$r_{uy}(\tau) = \int_{-\infty}^{\infty} g(s) r_{xx}(\tau - s) ds. \tag{2.42}$$

By applying the deconvolution theorem we get

$$R_{uy}(j\omega) = G(j\omega) R_{uu}(j\omega) \tag{2.43}$$

from which the expression for transfer function is obtained as

$$G(j\omega) = \frac{R_{uy}(j\omega)}{R_{uu}(j\omega)}. \tag{2.44}$$

In this expression, $R_{xx}(j\omega)$ denotes power spectral density of input signal $u(t)$ and $R_{uy}(j\omega)$ denotes cross-spectral density.

This approach is very useful for estimating the transfer function of a system that is under operation. If a test signal of small amplitude $u_a(t)$ that is uncorrelated to the normal (operating) input signal $u_n(t)$ is added to the input, the resulting input signal will be

$$u(t) = u_n(t) + u_a(t). \tag{2.45}$$

Output of the system $y(t)$ will then consist of two components, $y_n(t)$ which is due to the

normal input signal and $y_a(t)$ which is due to the test signal:

$$y(t) = y_n(t) + y_a(t). \tag{2.46}$$

The cross-spectral density will be for this case

$$R_{u_a y}(j\omega) = R_{u_a y_n}(j\omega) + R_{u_a y_a}(j\omega). \tag{2.47}$$

The first term is zero, because of the assumption that the test signal is uncorrelated to the normal input, and therefore, to the normal output. On the other hand, from (2.44) we have

$$G(j\omega) = \frac{R_{u_a y}(j\omega)}{R_{uu}(j\omega)} = \frac{R_{u_a y_a}(j\omega)}{R_{u_a u_a}(j\omega)}. \tag{2.48}$$

In this way, the transfer function can be determined while the system is under operation, without interfering too much with normal operating conditions, providing that the test signal has low amplitude. That is also important as a low-amplitude signal is not likely to drive the system into a nonlinear mode of operation. The drawback is that measuring equipment must be precise enough to accurately measure the test signal and its effects on the output. The test signal should ideally be white noise, as this signal has flat power density spectrum over the entire frequency range. Instead of white noise, which is impossible to generate, a pseudo-random binary sequence, which does not suffer from such drawbacks, is popularly used. The pseudo-random binary sequence is periodic with period T. It is important that this period be comparable with or greater than the dominant time constant of the system. Pseudo-random signals are similar to white noise in that their auto-correlation function is defined only at instances 0, T, $2T$, . . ., and is zero for all other values of τ, just as in the case of the white noise, where the autocorrelation function has non-zero value only for $\tau = 0$. For detailed discussion about pseudo-random sequences and their applications, see Davies (1970) and Godfrey (1993).

This approach can be also used for the MIMO systems. In this case, the test signal

in each output has to be cross-correlated with each input, and all transfer functions can be identified. This procedure requires that the system be available for identification for a long period of time as many experiments have to be done, although with minimum interference with normal operation. A faster way of identification is suggested in Davies (1970) by a technique called simultaneous multi-input method. In this case, all inputs are excited by different pseudo-random signals. For cross-correlation between the i-th input and j-th output, by applying pseudo-random test signal of period T, we can write

$$r_{u_i y_j}(\tau) = \frac{1}{T}\int_0^T u_i(t) y_j(t+\tau) d\tau. \tag{2.49}$$

On the other hand, output y_j has components due to every input:

$$y_j(t) = \sum_{i=1}^{p} \int_{-\infty}^{\infty} g_{ij}(s) u_i(t-s) ds. \tag{2.50}$$

If y from the equation (2.50) is substituted for the output in the (2.49), and the resulting expression rearranged, we get

$$r_{u_i y_j}(\tau) = \int_{-\infty}^{\infty} g_{ij}(s) \left[\frac{1}{T} \sum_{k=1}^{p} \int_0^T u_i(t) u_k(t+\tau-s) ds \right] d\tau \tag{2.51}$$

.The expression in brackets is the cross-correlation between test signals, so we can write

$$r_{u_i y_j}(\tau) = \sum_{k=1}^{p} \int_{-\infty}^{\infty} g_{ij}(s) r_{u_i u_k}(\tau - s) ds. \tag{2.52}$$

Assuming that response will decay to zero before the length of the period of pseudo-random test signal, which is normally sufficiently long in duration and assuming that the response of the system is zero for $t < 0$, (2.52) can be written as

$$\begin{aligned} r_{u_i y_j}(\tau) &= \sum_{k=1}^{p} \int_0^T g_{ij}(s) r_{u_i u_k}(\tau - s) ds + \sum_{k=1}^{p} \int_T^{2T} g_{ij}(s) r_{u_i u_k}(\tau - s) ds + \ldots \\ &= \sum_{k=1}^{p} \int_0^T g_{ij}(s) r_{u_i u_k}(\tau - s) ds \end{aligned}$$

$$= \int_0^T g_{ij}(s) r_{u_i u_i}(\tau - s) ds + \sum_{k=1}^{p} \int_0^T g_{ij}(s) r_{u_i u_k}(\tau - s) ds, \; k \neq i. \qquad (2.53)$$

It is now argued from Davies (1970) that the second term of this sum can be ignored because the test signal can be chosen in such a way that cross-correlations between test inputs are negligible. In order for any pseudo-random signal to be good, its auto-correlation function should be negligible for all τ, except for $\tau = 0,\ T,\ 2T,\ \ldots$ where it should have spikes. If these spikes are represented by an impulse of the form $K\delta(\tau)$, then, from the "sifting" property of the impulse function, we can estimate the impulse response for each input-output pair from

$$r_{u_i y_j}(\tau) = K g_{ij}(\tau). \qquad (2.54)$$

In this way, all weighting sequences g_{ij} can be estimated in one single experiment.

All previously described methods for identification of the transfer function (or impulse response) matrix, require that the system be available to us for experiments, an assumption which is often not fulfilled. Sometimes we are dealing with systems that are very large and this approach is not practical or even impossible (for example, socio-economic systems). The advantage of this method is that, apart from the assumption of linearity, very little has to be known *a priori* about the system under observation, in order to estimate the transfer function matrix. However, identification from normal operating input-output data by this method is not feasible.

What is necessary is to have a method of identifying the system from the given input-output sequence when there are no possibilities of any further experimenting on the system. In addition to linearity, the only assumption that we will make here is that the input-output sequence is good enough to enable identification of most of the system modes. This can be achieved by identifying a model in its state-space representation.

2.3.3 Identification of State-Space Models

In this section identification of a system by the model of the form given in (2.29) will be analyzed. Because this constitutes the model class in which a suitable model will be chosen, the most important task is to establish proper parameterisation of such models. As in the case of SISO systems, the main choice that has to be made is the choice of the model order. It will be established in Chapter 3 that in the case of SISO models, a model of order n will require estimation of at most $2n$ parameters, if dead-time parameter is omitted. In the case of a multivariable model (2.29) with p inputs, q outputs and n states, the required number of parameters that have to be estimated is theoretically equal to $n^2 + np + nq + pq$. In the majority of cases the last term in this sum can be taken to be zero, as inputs are rarely directly fed to outputs, resulting in the matrix $\mathbf{D}$ from (2.29) being a zero-matrix. In the general case, the number of parameters of MIMO model that have to be determined is often prohibitively large, even in the case of relatively low model orders and the assumption that $\mathbf{D}$ is zero-matrix. In order to establish the identification procedure for multivariable systems, the concept of canonical form has to be discussed.

Canonical Forms

When state-space equations are formed on the basis of physical considerations, the parameters of state-space equations of the model should have a physical interpretation. In this situation the question of identifiability is automatically satisfied (Glover and Willems 1974), and the nature of the problem will dictate the number of parameters. Some of the parameters may be eliminated through the process of aggregation or other model reduction techniques.

In the case of black-box identification, when the model structure is not known beforehand, there is no clear guidance on the composition of matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ and on the number of free parameters (Correa and Glover 1986). For controllable and observable models (which should always be the result of modelling from input-output behaviour of the system), models used for identification should be in so-called canonical form, as they

can be described by the minimal number of parameters.

The rigorous treatment of terms related to definition of canonical forms can be found in Denham (1974). The concept of canonical form arises from the question of whether a suitable form can be found to represent a class of models within the given model set. For example, within the model class given by

$$M = \{(\mathbf{A}, \mathbf{B}, \mathbf{C}) \mid \mathbf{A} \in R^{n \times n},\ \mathbf{B} \in R^{n \times p}, \mathbf{C} \in R^{q \times n}\} \tag{2.55}$$

which represents all linear multivariable models of order n with p inputs and q outputs, two models $M_1 = (\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1)$ and $M_2 = (\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2)$ can be considered equivalent, in notation $M_1 \sim M_2$, if there exist the following relations between them

$$\begin{cases} \mathbf{A}_2 = \mathbf{P}\mathbf{A}_1\mathbf{P}^{-1} \\ \mathbf{B}_2 = \mathbf{P}\mathbf{B}_1 \\ \mathbf{C}_2 = \mathbf{C}_1\mathbf{P}^{-1} \end{cases} \tag{2.56}$$

where $\mathbf{P}$ is arbitrary, nonsingular, $n \times n$ matrix. This similarity transformation is an equivalence transformation. A particular model from an equivalence class that represents all of its members will be referred to as the canonical model, and its parameterisation form is called canonical form. The other important characteristics of a model in the canonical form is that the resulting parameterisation is unique, meaning that no model can be represented in the same canonical form with a different set of parameters. This is not the case for general state-space models as there is an infinite number of state-space representations that can be used to describe a particular model (Young and Wang 1987).

Of special interest are those forms that can be represented by a minimal number of parameters, relative to the other members of its equivalence class. Such forms have been widely studied in the literature because of their importance for realizations and identification (see Buddin 1971, Bingulac 1976, Gupta and Fairman 1974 and Liu and Suen 1977). As a prelude to the multivariable model canonical forms, we briefly state

(minimal) canonical forms for SISO case.

Canonical Forms for SISO Models A linear model of order n, given by its transfer function

$$G(s) = \frac{b_{n-1}s^{n-1} + b_{n-2}s^{n-2} + \ldots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0} \tag{2.57}$$

has $2n$ free parameters, which is the number of non-fixed coefficients of numerator and denominator. On the other hand, a SISO model of the same order in the state-space form

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}\mathbf{u} \\ \mathbf{y} = \mathbf{c}\mathbf{x} \end{cases} \tag{2.58}$$

in which $\mathbf{x}$ is an n-component state vector, and $\mathbf{b}$ and $\mathbf{c}$ are $n \times 1$ and $1 \times n$ input and output matrices, respectively, will have $n^2 + 2n$ parameters, which is the number of elements of matrix $\mathbf{A}$ and vectors $\mathbf{b}$ and $\mathbf{c}$. Since both of these representations describe the same linear system, the number of parameters in both transfer function and state-space representations should also be the same. That means that there must be a particular form of the matrix $\mathbf{A}$ and vectors $\mathbf{b}$ and $\mathbf{c}$ such that the number of parameters in resulting dynamic equations will be equal to the number of parameters for an equivalent transfer function representation. The canonical form state-space realization of the

transfer function in (2.57) will be (Rubio 1971, Raven 1979, De Carvalho 1993)

$$\begin{cases} \mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & . & 0 \\ 0 & 0 & 1 & . & 0 \\ . & . & . & . & . \\ 0 & 0 & 0 & . & 1 \\ -a_0 & -a_1 & -a_2 & . & -a_{n-1} \end{pmatrix} \\ \mathbf{b} = \begin{bmatrix} 0 & 0 & . & . & . & 0 & 1 \end{bmatrix}^T \\ \mathbf{c} = \begin{bmatrix} b_0 & b_1 & . & . & . & b_{n-2} & b_{n-1} \end{bmatrix}. \end{cases} \tag{2.59}$$

The dynamic equation corresponding to this form is called controllable canonical form realization of transfer function (2.57). This form is called controllable because it is controllable regardless of whether numerator and denominator of (2.57) are coprime or not (Chen 1984). Similarly, an observable canonical form realization can be defined as

$$\begin{cases} \mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & . & -a_0 \\ 1 & 0 & 0 & . & -a_1 \\ 0 & 1 & 0 & . & . \\ . & . & . & . & -a_{n-2} \\ 0 & 0 & 0 & 1 & -a_{n-1} \end{pmatrix} \\ \mathbf{b} = \begin{bmatrix} b_0 & b_1 & . & . & . & b_{n-2} & b_{n-1} \end{bmatrix}^T \\ \mathbf{c} = \begin{bmatrix} 0 & 0 & . & . & . & 0 & 1 \end{bmatrix}. \end{cases} \tag{2.60}$$

These dynamic equations have the same number of non-fixed parameters as the equivalent transfer function form.

Canonical Forms for Multivariable Models For multivariable models, similarly to the case of SISO models, we are looking for canonical form realizations of transfer function matrices. Extension of the concept of canonical form from SISO to the multivariable case is more difficult.

While it is easy to establish the order of the model given by its transfer function, which is the order of the polynomial in the denominator of the transfer function, for the transfer function matrix this is not so obvious. The order of transfer function matrix $\mathbf{G}$ is defined as the degree of its characteristic polynomial. The characteristic polynomial of a proper transfer function matrix is defined as the least common denominator of all of its minors. Methods of realization of transfer function matrix can be found in Chen (1984).

For SISO models, the controllable canonical form is uniquely determined by its model order. However, in the case of multivariable models, the canonical form in no longer unique. In addition to the model order and the rank of matrices $\mathbf{B}$ and $\mathbf{C}$, another set of parameters is necessary to uniquely define the canonical form. This set is called the set of Kronecker invariants. It can be obtained from the controllability matrix:

$$\begin{aligned} \mathbf{U} &= \left[\mathbf{B} \quad \mathbf{AB} \quad \mathbf{A}^2\mathbf{B} \quad \ldots \quad \mathbf{A}^{n-2}\mathbf{B} \quad \mathbf{A}^{n-1}\mathbf{B} \right] \\ &= \left[\mathbf{b}_1 \quad \ldots \quad \mathbf{b}_p \quad \mathbf{Ab}_1 \quad \mathbf{Ab}_2 \quad \ldots \quad \mathbf{Ab}_p \quad \ldots \quad \mathbf{A}^{n-1}\mathbf{b}_p \right]. \end{aligned} \tag{2.61}$$

The question is how to choose n linearly independent column vectors from (2.61). This can be done in several ways, depending on the search algorithm. Some of these methods can be found in Chen (1984) and Denham (1974). The one used in this text starts the search of the second expression in (2.61) for the first n linearly independent column vectors. After they are all found, they can be arranged and sequenced as in the following

$$\left[\mathbf{b}_1, \ldots, \mathbf{A}^{\nu_1-1}\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{A}^{\nu_2-1}\mathbf{b}_2, \ldots, \mathbf{b}_p, \ldots \mathbf{A}^{\nu_p-1}\mathbf{b}_p\right] \tag{2.62}$$

where $\nu_1+\nu_2+\ldots+\nu_p = n$. The set $\{\nu_1, \nu_2, \ldots, \nu_p\}$ is also called the set of controllability

indices. Given this set of controllability indices, the following controllable canonical form for a multivariable model can be used

$$A = \begin{bmatrix} \mathbf{A}_1 & \mathbf{K}_{12} & . & . & \mathbf{K}_{1p} \\ \mathbf{K}_{21} & \mathbf{A}_2 & . & . & \mathbf{K}_{2p} \\ . & . & . & . & . \\ . & . & . & . & . \\ \mathbf{K}_{p1} & \mathbf{K}_{p2} & . & . & \mathbf{A}_p \end{bmatrix} \tag{2.63}$$

such that submatrix $\mathbf{A}_i$ is $\nu_i \times \nu_i$ matrix given as

$$\mathbf{A}_i = \begin{bmatrix} 0 & 0 & . & . & . & x \\ 1 & 0 & . & . & . & x \\ 0 & 1 & . & . & . & x \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & . & . & 1 & x \end{bmatrix} \tag{2.64}$$

and $\mathbf{K}_{ij}$ is $\nu_i \times \nu_j$ matrix given by

$$K_i = \begin{bmatrix} 0 & . & . & 0 & x \\ . & . & . & . & x \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & . & . & 0 & x \end{bmatrix} \tag{2.65}$$

where x designates the positions of non-zero elements. Matrix $\mathbf{B}$ in this canonical form

is a fixed matrix of the form

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}^1 \\ \mathbf{B}^2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{B}^p \end{bmatrix} \tag{2.66}$$

where $\mathbf{B}^k$ is $\nu_i \times p$ matrix with all elements zero except for the one fixed at the value of 1 at the position i in the first row:

$$\mathbf{B}^k_{i,j} = \begin{cases} 1, \ i = 1, \ j = k \\ 0, \ \text{for all other matrix indices.} \end{cases} \tag{2.67}$$

Matrix $\mathbf{C}$ is completely free. Therefore, we can conclude that this canonical form has in total $n(p+q)$ free parameters that have to be determined, while other n^2 are fixed zeros or ones. For the similar analysis for observable canonical forms and discussion on minimal number of parameters, see Bingulac (1976), where it is argued that under certain conditions, the minimal number of parameters can be given as $n(p+q^2-q+1)$.

Identification in Canonical Form

Identification methods using the state-space models are known to be numerically more stable than others (like transfer function, weighting function, etc., see Moonen *et al.* 1991). In identification by state-space models, canonical forms are clearly very important. Not only are they described by a minimal number of parameters, but also the parameterisation associated with such canonical forms is unique. This means that for the same model there is only one set of parameters which uniquely describes the given model. As already mentioned, this is not the case if the state-space equations are not given in canonical form, in which case the same model can be described with infinitely many different parameterisations.

However, use of canonical forms in identification of multivariable systems is not always advisable. This is the case when identification is performed from noisy data (Glover and Willems 1974). However, when very little is known *a priori* about the system that we would like to model, identification in canonical form is very useful, and sometimes it is the only way for black-box type identification. Having decided on a particular model structure, i.e. model order and set of structural indices, the remaining problem is the one of parameter estimation. This can be done in the same manner as in the case of the SISO systems. There is a multivariable least-squares method (Hsia 1977, Goodwin and Payne 1978). Also, frequently used are maximum likelihood methods. In this study, the model parameters will be estimated by minimizing a suitably chosen model output error, very much in the same way as in SISO case. The output error based matching criterion is now marginally more complex because there is in general more than one output from the model. The number of parameters even in minimal realizations is still very large for models representing large-scale multivariable systems. The main problem with direct estimation of state-space equation matrices is that the time taken for such identification can be extremely long, making any real-time application of such identification difficult. The proposed new methods for reducing the time required by the identification process will be conceptually similar to the one for SISO systems, which follows. We will apply the hierarchical identification method to multivariable systems and deal with the identification problem as the LSS identification. These will be presented in detail in Chapter 4.

But first, a hierarchical identification procedure for identification of SISO systems is presented in the next chapter.

2.4 Conclusion

In this chapter, some classical identification methods were described. A few examples were given in order to illustrate some practical difficulties related to SI. Some of these

include the choice of model order, the choice of the input signal and its duration, validation strategies etc. However, most of the difficulties arise from the numerical complexity of minimization of the model error. The required number of instructions for solving an optimization problem is shown to grow substantially faster than the number of optimization problem parameters. In the following two chapters we will develop an identification methodology that will reduce the number of parameters that simultaneously enter optimization routines.

Chapter 3

Hierarchical Identification of SISO Systems

Some of the numerical difficulties associated with identification were illustrated in the previous chapter. It can be concluded from the examples given there that searching for more accurate models by increasing the model order is often very costly and results are by no means guaranteed. On the other hand, it would be very beneficial if the time required by SI procedures could be decreased. This is particularly important in real-time applications, where all simulations must be completed within a given time interval. One approach is to use parallel processing. As parallel processing is a broad area, we need to clarify the issue for application in SI.

There are several aspects of this parallelism. The hierarchically lowest level of parallelism is achieved at computer instruction level. Such parallelism is embedded in processor architecture (Malinowski *et al.* 1985, Pearse *et al.* 1986, Bertsekas and Tsitsiklis 1989). One example is instruction pipelining, where execution can be speeded up, at least in theory, as many times as the value of so the called depth of pipeline. Due to the branching in the program execution, this never exactly happens, but still pipelining brings significant increase in processing speed.

The next level of parallelism can be achieved by transforming sequential numerical

algorithms, where possible, into equivalent, parallel form. There are many examples of these, including parallelization of vector addition (Bertsekas and Tsitsiklis 1989), solving partial differential equations, solving systems of ODE, etc. (Lèi *et al.* 1985). The efficiency of all of these methods will largely depend on how a sequential algorithm (or portions of it) can be converted into one suitable for parallel processing. Often, specialized processors are specifically built for efficient execution of such algorithms. However, it is obvious that such algorithms and processor realizations cannot be used to solve more general problems, i.e. they are restricted to the applications for which they are specifically designed.

The highest level of parallelizations is achieved at the application level. In this case, the initial problem is broken into a set of smaller-size problems, which are similar or may even be essentially the same as the original one. These smaller-size problems can then be solved independently of each other. Application level decomposition of the original problem will crucially depend on the nature and structure of the problem itself. For example, the decomposition strategies developed for circuit analysis cannot be directly applied to solve problems in, for instance, ecology or economy (Šiljak 1978). However, once decomposition is achieved, the resulting problems are fairly autonomous. Parallelism at this level is virtually independent of the underlying processor architecture, and it may go up to the level of employing loosely connected or even completely separated computers for decomposed problem resolution.

It is exactly this type of parallelization that we have in mind for development of hierarchical identification procedures proposed in this research. Our primary goal will be to break the initial identification problem into the set of smaller size identification problems. In other words, we are looking for an identification procedure that would require estimation of a smaller parameter vector in each of the phases of such procedure, compared to a one-shot approach. Apart from the smaller number of floating point operations required, smaller parameter vectors should, in general, bring fewer problems regarding numerical stability and the choice of initial values for parameters. These techniques have been

utilized in Janković and Bajić (1996a, 1996b) and Bajić and Janković (1997).

Our main tool for breaking up the original identification problem into a set of smaller size problems is the decomposition principle. Decomposition was introduced in Section 1.3.3, but it will be necessary to treat the subject in more depth in order to develop a procedure for hierarchical identification.

3.1 Decomposition as Problem Size Reduction Technique

The decomposition principle is widely used in analysis of a large class of complex problems (Šiljak 1978, 1983). It is often introduced heuristically, or it is related to some specific area for which it is employed. However, this technique can be described in a more formal way, and that will be discussed now.

3.1.1 Formal Treatment of Decomposition Technique

In general, the decomposition principle is used in the analysis of the so called composite problems. A composite problem can be defined as one consisting of interacting subproblems. It can be seen from this definition that the analysis of LSS would inevitably lead to composite problems, because such systems consist of interacting subsystems. Decomposition is a natural approach to this class of problems. It is worth noting that the decomposition principle is not restricted to the analysis of dynamic systems, but has a much broader application for complex problems in general. In order to be able to formally treat the composite problem analysis, some notation is necessary.

Following mainly Guardabassi (1982), an *abstract problem* P can be defined as ordered triple

$$P = (D, Z, \pi) \tag{3.1}$$

where D is some data set, Z is a subset of the solution set S, and $\pi : D \rightarrow Z$ is

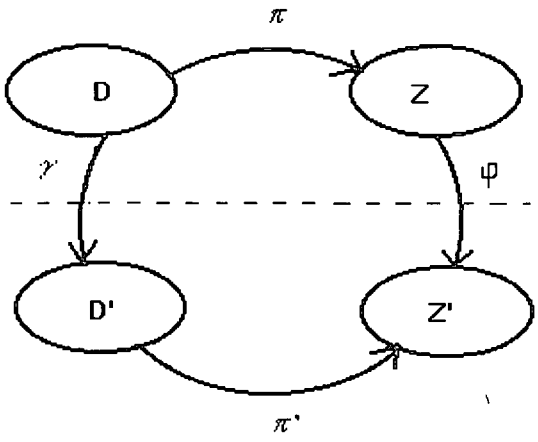


Figure 3-1: Decomposition of a problem

some function between the two sets. This function is called the *intrinsic function of the problem* P. To solve the problem P means to find an image of D under function π. If $\pi(D) = \{\emptyset\}$, then we say that the problem has no solution. If the following expression holds

$$(\forall d \in D) \qquad (\pi(d) = a,\ a \in Z,\ a \neq \emptyset) \tag{3.2}$$

image Z consists of a singleton only, for all $d \in D$, then we say that the problem has a unique solution.

If the problem is a difficult one, we can employ similar approach to the one used in aggregation: we will transform problem P into some other problem P' in a such a way that the relationship between the two will enable us to solve P by solving P'. The related problem P' should be easier to solve than original problem P.

If the following relation holds (see Figure 3-1)

$$\pi = \varphi^{-1}\pi'\gamma \tag{3.3}$$

in which function π' is the intrinsic function of the problem $P' = (D', Z', \pi')$, function π is the intrinsic function of the problem $P = (D, Z, \pi)$, and $\gamma : D \rightarrow D'$ and $\varphi : Z \rightarrow Z'$ are

some mappings between the data sets of the problems P and P' and their solution sets respectively, we say that perfect decomposition is achieved. Implicit to this definition is the assumption of invertibility of the function φ. This will obviously make sense only when the composite problem's intrinsic function π' is easier for analysis than the original problem's intrinsic function π. If we can find functions φ, π' and γ such that (3.3) holds, then problem P' is called proper reformulation of the problem P. The concept of proper reformulation is mainly of theoretical importance as finding the functions φ, π', and γ is possible only for specially created examples. In real life situations we are not able to find them exactly, and then we restrict ourselves to approximating the problem P by some other problem P', such that their solutions are close in some way. Thus, the decomposition yields a more or less accurate approximation of the original problem and we hope that the benefits of such decomposition will outweigh the significance of those inaccuracies.

When transforming the original problem P into a problem P', the question that arises is what sort of problem should P' be. If we want to employ the decomposition principle, then P' should belong to the class of composite problems.

A composite problem Ψ of order n is defined as a set of n interacting subproblems P_i, $i = 1, 2, ..., n$. The composite problem, just as any other problem, has its data set and its solution set, which are in this case called global data and global solution sets. On the other hand, each subproblem has its own data and its own (local) solutions. However, these data sets are not in general disjoint subsets of the global data set of the composite problem, but will depend on the solutions of other subproblems and, possibly, on the solution of the subproblem corresponding to it. To describe the dependence of each of the subproblem's data sets on the local solutions of others, we need a set of interacting functions. Finally, the global solution must be a function of the local solutions and global data, and this function is called a global solution function. This means that we

can describe a composite problem as

$$\begin{cases} \Psi = \{\Pi, D_g, S_g, I, \Gamma\} \\ I = \{I_i : S_j \rightarrow D_i,\ i = 1, 2, ..., n,\ j = 1, 2, ..., n\} \\ \Gamma : (S_i, D_g) \rightarrow S_g \end{cases} \tag{3.4}$$

where Π is a set of subproblems, each of them being of the form (3.1), D_g is the global data set, S_g is the global solution set, S_i is local solution set for subproblem P_i, I is the set of interacting functions and Γ is a global solution function.

The knowledge of the interacting function set I and the set of global solution functions Γ are clearly very important in composite problem analysis. A very convenient tool for subproblem interaction analysis is graph theory since graphs can be successfully used in describing interactions among the subproblems (Šiljak 1978). These graphs (digraphs, to be more precise) are called interaction graphs. Given an intrinsic function π_i, the solution of subproblem P_i will depend only on its local data set D_i. Because of this, we can justify representation in which each node of interaction graph represents both the subproblem P_i and the local data D_i set associated with it. The branches of the graph show how a particular local data set depends on the local solutions of other subproblems. The term extended graph will denote a graph in which branches representing a global solution function are included, as well as nodes denoting global data and solution set.

Relation Between Decomposition and Aggregation

Related to decomposition and composite models is the notion of equivalent overall (global) problem of the composite problem which can be described as

$$P_g = (D_g, Z_g, \pi_g) \tag{3.5}$$

In this case, the description given by (3.4) can be regarded as internal description of the problem, and the one given by (3.5) as its aggregate. In this way, we are able to formally

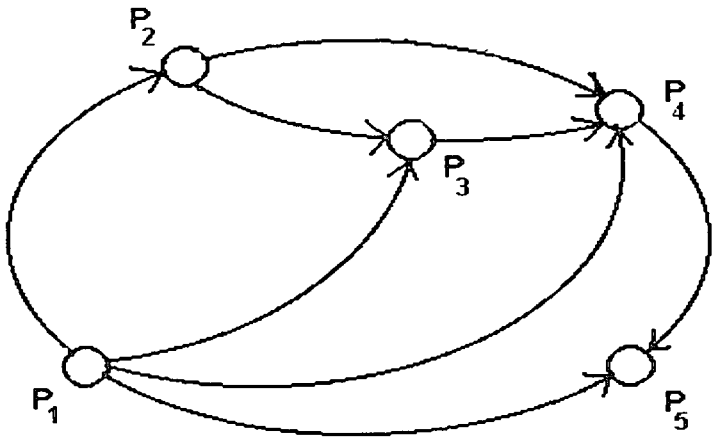


Figure 3-2: Interaction graph

establish the link between the aggregation and decomposition in problem analysis.

When graphs are used in a composite problem analysis, we can see that in the case of acyclic graphs (ones that contains no closed loops), there is a strict order in dependency of each node's local data set on the preceding nodes' local solutions. In that case the solution of the composite problem can be found as the sequence of solutions of subproblem set members. This is so because if the interaction graph is acyclic, then each subproblem P_k will depend only on its local data set D_k. In this case D_k depends only on the solutions of subproblems P_i, $i = 1, 2, ..., k-1$ and, possibly, on its own solution, but not on the local solutions of subproblems P_i for $i > k+1$. As can be seen from Figure 3-2, problem P_3 depends only on the solutions of the problems P_1 and P_2, but not on problems P_4 and P_5. Similar argument holds for all other nodes in this graph. In general, this will not be the case and some closed loops will occur in the graph. In order to make decomposition effective in the sense that subproblems can be solved in a sequence, a two phase approach is necessary. In the first one we must identify the strongly connected subgraphs. A strongly connected (di)graph is a graph in which every node can be reached from any other node of that graph. If we now apply the aggregation principle in such a way as to create new graph nodes from identified strongly connected subgraphs, we

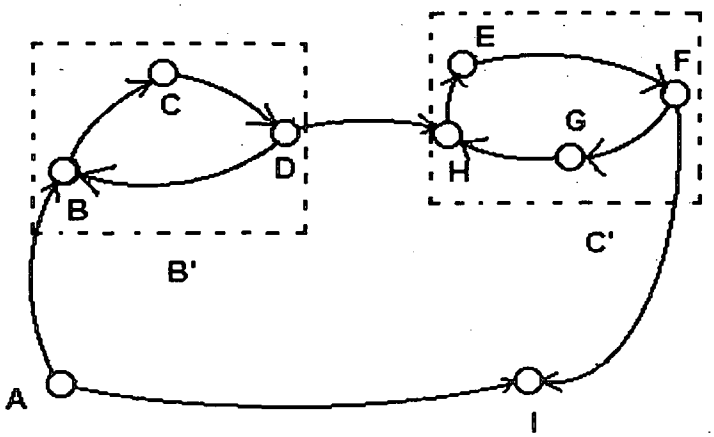


Figure 3-3: Aggregation of interaction graph

obtain a simplified, aggregated, acyclic graph. Then we can resolve this aggregated (but still composite) subproblem separately, and the solution is then used for the second phase sequence (Figure 3-3). These considerations are critical due to the way we approach the hierarchical identification of LSS in this and the following chapter.

It is important to note that if interactions among the subproblems are limited and well-defined, we may try to evaluate aggregated subproblems independently from one another. For this case, subproblem nodes are unconnected, and local solutions can be obtained in parallel. In the second phase, we may study the influence of other local solutions and the global data set on local data sets. This will translate into some form of subproblem solution feedback.

Although this formalism sheds some light on the decomposition principle, it does not provide us with any guidelines as to obtaining the subproblem set or establishing the interactions among them. This cannot be done for the general case, because some insight into internal structure of the problem is often necessary. For example, an application of the decomposition principle in complex optimization problems for static multilevel systems can be found in Schoeffler (1971), where subproblem interaction assumes hierarchical form. This approach is also known as the decomposition-aggregation method, as

coordination variables are introduced as variables of the global solution function.

3.1.2 Decomposition of Large-Scale Dynamic Systems

Due to the way identification will be conducted, we are mainly interested in the analysis of large-scale dynamic systems. When studying decomposition of LSS, we can apply the formalism of the previous chapter, but it must be borne in mind that the above formalism was intended for decomposition of a problem in general. For example, we may wish to study input-output relations of LSS, which can bring us to the complex set of differential equations, or a complex optimization problem, which can be then solved using decomposition-aggregation and decomposition-coordination principles (Pearson 1971, Crorkin *et al.* 1985, Malinowski *et al.* 1985). Decomposition-coordination methods are particularly interesting for hierarchical control, in which modules on higher hierarchical levels coordinate the actions of lower-level subsystems. As we are interested in structural and numerical issues associated with such systems, all of the above formalism will apply to them. This time, however, we can be more specific, as description of a dynamic system is readily available in the form of

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} = \mathbf{C}\mathbf{x}. \end{cases} \tag{3.6}$$

A complex, large-scale dynamic system is assumed to consist of interacting subsystems. The question is whether or not we can decompose the original dynamic system into the set of interacting dynamic subsystems. This set, together with the set of interactions among the subsystems defines the composite dynamic model, the members of which we would like to be able to study separately.

We have already seen that the mapping from the global problem description into a composite one will enable us to use decomposition methods in dealing with such a problem. If we decompose the dynamic LSS into a set of interacting dynamic subsystems, the problem is to establish the relation between them as well as the relations among

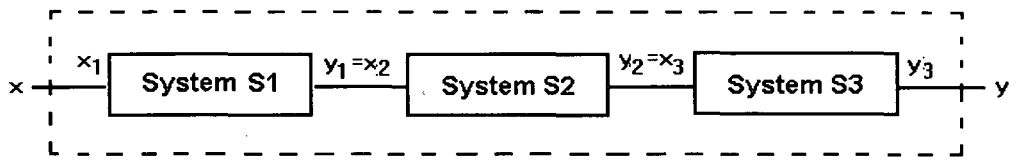


Figure 3-4: Serial connection of systems

subsystems. From the previous arguments, we can conclude that decomposition has potential benefits in terms of a reduced problem size, and offers a sequential or even parallel approach to analysis of dynamic subsystems. In the spirit of relation (3.5), a model will also have its global and composite representation. A formalism regarding the relation between a complex system, its global (aggregated) representation and decomposition can be found in Takahara (1982). To clarify the issue of global and composite model representation, we borrow an example from Takahara's paper. As we deal here with the input-output representations of the systems, we will assume that, for some system S all possible inputs belong to the set X, and outputs belong to the set Y. Suppose now that we have three systems, S_1, S_2 and S_3 such that $S_1 \subset X_1 \times Y_1$, $S_2 \subset X_2 \times Y_2$ and $S_3 \subset X_3 \times Y_3$, where X_i and Y_i represent input-output sets of systems S_i. We now connect these systems serially, as in Figure 3-4, and specify input-output relations for each system S_1, S_2 and S_3 as $y_1 = x_1$, $y_2 = 2x_2$ and $y_3 = 2x_3$, respectively. Systems S_1, S_2 and S_3 are now subsystems of the a composite system $\hat{S}$. This global representation has its own input-output sets, which will in this case be $\hat{X} = X_1$ and $\hat{Y} = Y_3$. The input-output relation here will be $\hat{y} = 4\hat{x}$, where $\hat{x} \in \hat{X}$ and $\hat{y} \in \hat{Y}$. We see now that in the global representation $\hat{S} \subset \hat{X} \times \hat{Y}$ a lot of information about subsystems S_1, S_2 and S_3 is lost. As opposed to global system representation, in composite representation our goal is to retain the information pertinent to the subsystems. For the above example, we can define composite representation of the system as $S \subset X \times Y$, but now S is defined as a set of ordered pairs of triples, $((x_1, x_2, x_3), (y_1, y_2, y_3))$, such that $(x_1, y_1) \in S_1$, $(x_2, y_2) \in S_2$, $(x_3, y_3) \in S_3$ and, to indicate serial connection, $x_2 = y_1$ and $x_3 = y_2$. Sets X and Y are

now $X = X_1 \times X_2 \times X_3$ and $Y = Y_1 \times Y_2 \times Y_3$.

The question is now how to relate these two representations. In order to establish this relation between the composite and global representations, we should find mappings $h_1 : X \rightarrow \hat{X}$ and $h_2 : Y \rightarrow \hat{Y}$ such that $h_1(x_1, x_2, x_3) = \hat{x}$ and $h_2(y_1, y_2, y_3) = \hat{y}$. In other words,

$$\begin{cases} (\forall (x, y) \subset S)(h_1(x),\ h_2(y)) \subset \hat{S} \\ (\forall (\hat{x}, \hat{y}) \subset \hat{S})(\exists (x, y) \subset S),\ h_1(x) = \hat{x},\ h_2(y) = \hat{y}. \end{cases} \tag{3.7}$$

We see that the pair (h_1, h_2) forms surjection from S to $\hat{S}$, as in this case mappings are uniquely determined by the input-output mappings of subsystems.

In a similar consideration for the relation between the global and composite representation, we can find two mappings $k_1 : \hat{X} \rightarrow X$ and $k_2 : \hat{Y} \rightarrow Y$, such that $k_1(\hat{x}) = (\hat{x}, \hat{x}, 2\hat{x})$ and $k_2 = (\hat{y}/4, \hat{y}/2, \hat{y})$ (because of the input-output relation of each of the subsystems S_1, S_2 and S_3 and the serial connection among them). We can then write

$$\begin{cases} (\forall (\hat{x}, \hat{y}) \subset \hat{S}) \implies (k_1(\hat{x}), k_2(\hat{y})) \subset S \\ ((k_1(\hat{x}), k_2(\hat{y})) = (k_1(\hat{x}_1), k_2(\hat{y}_1))) \implies (\hat{x}, \hat{y}) = (\hat{x}_1, \hat{y}_1). \end{cases} \tag{3.8}$$

This time, the pair (k_1, k_2) forms injection from $\hat{S}$ to S.

In this simple case, mappings h_1 and h_2 can be shown to be bijective, making S and $\hat{S}$ isomorphic. In the general case we will not have enough insight into the internal composition of a system and therefore the two representations need not be isomorphic. Takahara's paper investigates general properties of these functions.

Pearson (1971) introduced the notion of an integrated system. An integrated linear systems can be described by

$$\dot{\mathbf{x}}_i = \mathbf{F}_i(\mathbf{x}_i, \mathbf{m}_i, \mathbf{u}_i, t) \tag{3.9}$$

where $\mathbf{x}_i$ is a state vector, $\mathbf{m}_i$ is a control vector, $\mathbf{u}_i$ is an input vector, $\mathbf{x}_i(t_0) = \mathbf{x}_{i0}$ and $i = 1, 2, ..., N$. The coupling between the subsystems in terms of the output feedback can

be described by

$$\mathbf{z}_i = \sum_{j=1}^{N} \mathbf{L}_{ij}\mathbf{y}_i. \tag{3.10}$$

Therefore an integrated problem is given in a composite representation, as subsystem descriptions are clearly present in the overall integrated problem description.

In another example, it is well known that through similarity transformation the dynamic system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{3.11}$$

can be brought, by change of the state variables $\mathbf{z} = \mathbf{Q}^{-1}\mathbf{x}$, where $\mathbf{Q}$ is a non-singular matrix, to

$$\dot{\mathbf{z}} = \mathbf{H}\mathbf{z} + \mathbf{Q}^{-1}\mathbf{B}\mathbf{u},$$

with $\mathbf{H} = \mathbf{Q}^{-1}\mathbf{A}\mathbf{Q}$. The matrix $\mathbf{Q}$ can be selected in such a way that the matrix $\mathbf{H}$ will be diagonal if all eigenvalues are distinct, and it may be in Jordan canonical form if they are not. In the first case we have obtained representation of the model that has an equivalent composite representation in the form of parallel connection of low-order models. In the second case, this decomposition will only be partial, related to the structure of the Jordan blocks.

Methods of decoupling multivariable models by state or output feedback also have an aim of obtaining parallel decomposition of the original model.

In conclusion, analysis of a large-scale dynamic model poses serious difficulties. Various methods of attacking these problems are suggested (Šiljak 1978, Brown and Norton 1980, Malinowski 1985). They are mostly based on decomposition of the original large-scale problem into series of smaller-order problems. Where the nature of interactions between system components is known, matrices of interaction may be formed and if it leads to sparse matrices, optimization methods specially suited for them may be used. Techniques based on indexing of non-zero elements are popular in this context (Pooch and Nieder 1973). Input and output decentralisation is described in (Šiljak 1978), to

facilitate estimation of large-scale system stability.

Before proposing a method for hierarchical identification of SISO systems, we look briefly at some of the methods derived from LSE theory that have as an aim the reduction of the size of the problem of estimation.

3.2 Two Methods for Reducing the Size of Identification Problem

The first method for reduction of identification complexity that we mention here is based on the decomposition of LSE problem as described in Section 2.2.3. The algorithm was proposed by Sultan *et al.* (1985). The main idea is to split matrices $(\mathbf{X}^T\mathbf{X})^{-1}$ and $\mathbf{X}^T$ occurring in the equation (2.15) into the sum of diagonal and off-diagonal matrices. In this way we have to deal with independent, smaller size problems which can be solved separately (in parallel).

Another approach deals with "reusability" of the obtained models. As it was stated on many occasions, the model order (and the number of parameters) is not known in advance. This means that lower order models that do not pass the validation tests are to be discarded, and the procedure is then repeated for a new model of higher order. That is a costly exercise, as it was asserted in the Example 3 of the previous section. The question that now arises is whether models already obtained during the identification procedure can somehow be reused in order to obtain new, more accurate models. In this way the effort of obtaining the initial, less accurate models would not be completely wasted. This is a very important question in this study and before we propose a method for achieving this, we look at the method derived from LSE for dealing with this (details of the method can be found in Hsia 1977).

From (2.15) follows

$$(\mathbf{X}^T\mathbf{X})\hat{\theta} = \mathbf{X}^T y. \tag{3.12}$$

Suppose that the parameter vector $\hat{\theta}_p$ has been computed for the p parameter case. If now q $(q > p)$ parameter case is to be considered, with a one-shot approach, the size of the matrix $\mathbf{X}^T\mathbf{X}$ that has to be inverted is $q \times q$. Instead, a new parameter vector θ_q can be introduced and determined in the following way: if a parameter vector θ_q is partitioned as

$$\theta_q = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}, \ \theta_1 \in R^p, \ \theta_2 \in R^q \tag{3.13}$$

then $(m \times q)$ matrix $\mathbf{X}$ can be partitioned as

$$\mathbf{X} = [\mathbf{X}_1 \ \mathbf{X}_2] \tag{3.14}$$

where $\mathbf{X}_1$ is a an $(m \times p)$ matrix used to estimate parameter vector θ_p from m measurements $(\hat{\theta}_p = (\mathbf{X}_1^T\mathbf{X}_1)^{-1}\mathbf{X}_1^T\mathbf{y})$ and $\mathbf{X}_2$ is an $(m \times (q-p))$ matrix. We note that

$$[\mathbf{X}_1 \ \mathbf{X}_2]^T = \begin{bmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \end{bmatrix}$$

which, if substituted in (3.12) yields

$$\begin{cases} \theta_1 = \hat{\theta}_1 - (\mathbf{X}_1^T\mathbf{X}_1)^{-1}\mathbf{X}_1^T\mathbf{X}_2 \left[\mathbf{X}_2^T\mathbf{X}_2 - \mathbf{X}_2^T\mathbf{X}_1(\mathbf{X}_1^T\mathbf{X}_1)^{-1}\mathbf{X}_1^T\mathbf{X}_2\right]^{-1} \cdot \\ \qquad \cdot \mathbf{X}_2^T(\mathbf{y} - \mathbf{X}_1\hat{\theta}_1) \\ \theta_2 = \hat{\theta}_1 - \left[\mathbf{X}_2^T\mathbf{X}_2 - \mathbf{X}_2^T\mathbf{X}_1(\mathbf{X}_1^T\mathbf{X}_1)^{-1}\mathbf{X}_1^T\mathbf{X}_2\right]^{-1} \mathbf{X}_2^T(\mathbf{y} - \mathbf{X}_1\hat{\theta}_1). \end{cases} \tag{3.15}$$

We can see that both components of the parameter vector θ_q depend on the p-parameter vector θ_1 and the associated error of estimation (the term $\mathbf{y} - \mathbf{X}_1\hat{\theta}_1$). Matrices involved in (3.15) are of lower dimension than the matrix $\mathbf{X}$ from the equation (2.15) which would be have to be determined for the case of non-recursive identification.

This shows that previously obtained models can be re-used to create new ones, thus speeding up the calculation of the new parameter vector.

So far, two methods for reducing the problem size were briefly mentioned. The first

one suffers from the "order uncertainty" problem, as parameter vector size (or model order in the identification case) has to be incrementally adjusted during the estimation. In other words, if the so obtained model is unsatisfactory, we have to start again, with the new model order. The second one creates reusable models, where each subproblem benefits from the solution of the previous one. In light of our discussion in Section 1.3.1, for this approach the subproblem interaction graph will be cyclic. This means that it is suitable for a sequential estimation procedure, but not for a parallel one. A parallel procedure would, of course, be preferable. The initial proposal for the search for one such procedure based on the philosophy of LSS analysis was suggested to this author by Bajić (1994) and in what follows, several approaches to this problem are developed.

3.3 Hierarchical Identification

In this section we introduce the basics of a new methodology in SI that we name hierarchical identification, for reasons to be clear later. It would be obviously beneficial if parallelism and reusability of previously obtained models (in the way described in the previous section) could be combined for a general identification procedure, in which parameter estimation techniques are not necessarily based on the LSE. This is particularly important when, in order to properly capture the model dynamics, we have to perform the identification by high-order models, i.e. if we have to use high-order models to represent such systems in order to properly describe their dynamics. Although many systems can be adequately described by low-order models, in the case when signal-to-noise ratio is large, identification by low-order models may not be sufficient to describe most of the system modes. In order to use the high-order models for which identification would not be numerically very costly, we propose the following procedure following an initial idea of Bajić (1994) and those developed by the author.

Before presenting it, some explanations are necessary. The procedure is based on the solution of the so called exact model matching problem. This problem can be broadly

stated as follows: given two linear, possibly multivariable models, find the control law for the second model such that its input-output characteristics will match those of the first model. The control law is usually taken in the form of either state-variable feedback and/or dynamic compensation. For the formal treatment and conditions under which this matching can be done see, for example, Moore and Silverman (1972) and Tzafestas and Paraskevoupolos (1976). These solutions are given on the basis of some principal control laws for linear systems. The one that is mostly studied in the literature states that all system eigenvalues can be arbitrarily assigned by the use of constant matrix gain in the state feedback loop, providing that any complex, preassigned eigenvalues come in complex conjugate pairs (Wonham 1967). Therefore, by manipulating the feedback gain matrix, we can achieve the desired positioning of eigenvalues in the complex plane. The solution for our objective of employing parallelism and model re-usability now follows naturally in the form of the following two phase algorithm: firstly, we obtain the models of various orders, and secondly, using those models as building blocks, create a composite model which can be tuned using the feedback with static and/or dynamic compensation laws. In this way, we attempt to match such composite model to the "true" model (notion of the "true" model being one already described in Chapter 1).

Our primary goal is to minimize the number of parameters necessary for parameter estimation. In order to achieve this we propose employing output feedback, instead of state feedback (Levine and Athans 1970). In our case, we would prefer the term incomplete state feedback (used in Davison 1970) to the term output feedback, for reasons that will be clear shortly. In this way the number of eigenvalues that can be assigned (i.e. freely adjusted independently of other eigenvalues) is smaller, but at the same time the number of parameters of the feedback matrix that has to be estimated is also smaller. The benefits and drawbacks of utilization of output feedback for the purpose of hierarchical identification will be analyzed later.

The hierarchical identification procedure can be now described as follows. In the first phase, for each of the adopted subsystems of the overall system the transfer function

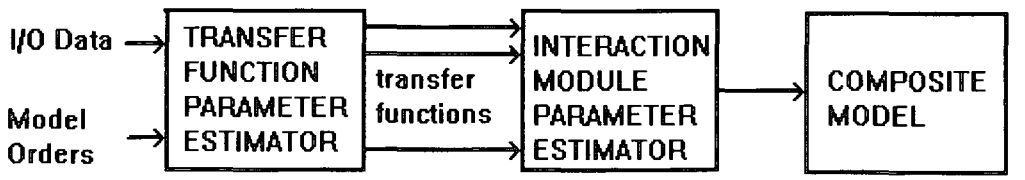


Figure 3-5: Hierarchical identification procedure

models (which can be of various orders) are determined. Each of these identification subproblems have for their local data the same I/O sequence, making the solution of each of subproblems independent of the solutions of others. This means that parallel processing for the identification of these transfer functions can be fully employed in this phase.

Because the so obtained models are in general of low order, they may not be able to fit the given I/O sequence adequately. To improve the dynamics, we connect these models via an interaction module, thus creating a composite model whose order is equal to the sum of orders of the previously identified models of subsystems. Parameters of the interaction module are then chosen in such a way that this composite model fits the I/O sequence better than the models resulting from the first phase. This constitutes the second phase. It is important to limit interactions among the subsystems so that the number of parameters for the second phase remains relatively small. In this way, the time required for estimation of a composite model should be shorter than the time required for estimation of an equivalent high-order transfer function by a one-shot identification approach. This procedure is presented in the Figure 3-5.

The first step consist of identification by models of orders n_s, $n_{s+1}, ..., n_{s+N-1}$ of the given I/O sequence, where n_s is the lowest model order and N is the number of subsystems. Each of these models (denoted as M_i) can be presented by the operator equations

$$M_i : y_i = G_i(s)e^{-L_i s}u \tag{3.16}$$

where y_i is the output of the model M_i, G_i is the i-th transfer function and L_i is the corresponding apparent dead-time. In our approach, every subsequent model should be of the order greater by one than the model immediately preceding it, as we do not know in advance what the proper model order is. However, other possibilities also exist regarding the choice of the orders of M_i, $i = 1, 2, ..., N$ (see Janković and Bajić 1997b).

These outputs are then used in the next phase when we identify the interactions among the subsystems. The choice of the interaction module is very important as it has to improve the composite model's dynamics with a reasonably small number of parameters. There are various ways to improve the composite systems dynamics. The one that we will consider here, in the spirit of the model matching approach, employs an output feedback scheme. It can be now seen why the term incomplete state feedback is more appropriate: the outputs of the subsystems are in fact no longer outputs of the composite model, but can now be regarded as a subset of the transformed state vector of the composite model. The composite model's output now becomes the true output, because it must also have only one output, as does the system we model by it.

The first composite model is given by the operator equation

$$\begin{cases} y = e^{-Ls}\mathbf{y}^* \\ \mathbf{y}^* = \mathbf{c}_{cm}\mathbf{y} \\ \mathbf{y} = [y_1 \ ... \ y_N]^T \\ y_i = G_i(s)e^{-L_i s}u_i \\ u_i = u + m_i \\ \mathbf{m} = \mathbf{A}_{cm}\mathbf{y} \\ m = [m_1 \ ... \ m_N]^T \end{cases} \tag{3.17}$$

for $i = 1, 2, \ldots, N$ where N is the number of subsystems, $\mathbf{y}$ is the vector whose components are the outputs of subsystems, $\mathbf{m}$ is the vector of feedback signals, $\mathbf{A}_{cm}$ is the $N \times N$ feedback matrix, L_i represents the apparent dead-time in every subsystem and $\mathbf{c}_{cm}$ is the $1 \times N$ vector combining the subsystems' outputs. The model given by

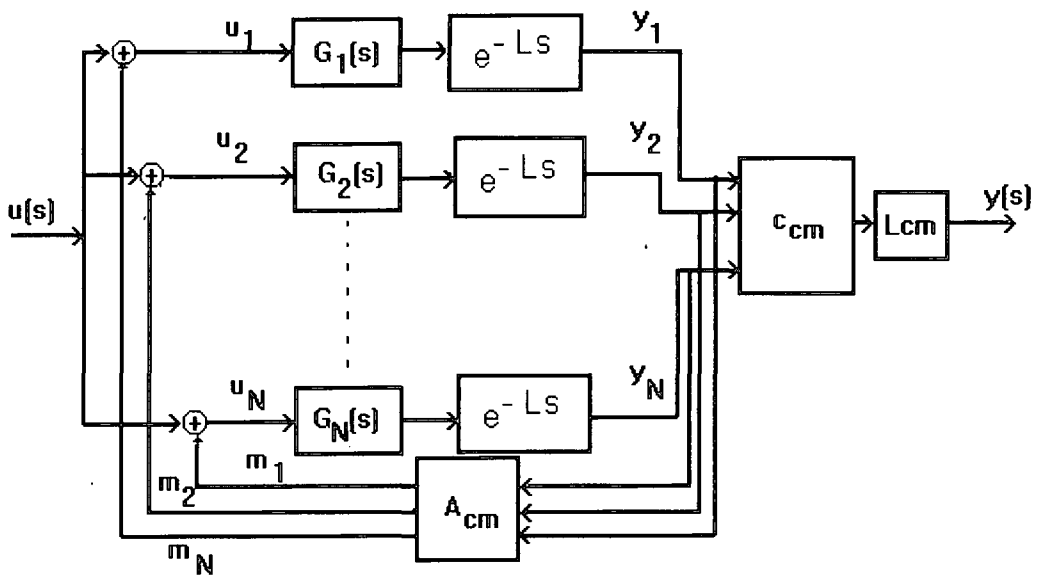


Figure 3-6: Interaction module and subsystems for SISO systems

(3.17) is shown in Figure 3-6. We refer to models given by (3.16) as subsystems of the composite model (3.17).

A word regarding the dead-times in the model. It can be seen from Figure 3-6 that the point for taking the signal for feedback can be either before or after the dead-time element. If we look at the composite model given by (3.17), the feedback loop starts at the output of each of the subsystems. Because dead-time elements L_i are part of the subsystem M_i, the feedback loop starts after the dead-time element.

However, there is a good reason for looking at this issue more closely. Dead-time is defined as time taken by the input signal to produce an output signal due to the propagation through the system. If we apply the test signal at moment 0 and the response signal (the one due to this particular input) appears after 1[s] at the output, we say that system dead-time is 1[s]. Very often, when signals at the input and particularly at the output are noisy, this dead-time cannot be precisely estimated. In this case, the dead-time parameter as used in the model is just one of the components of the parameter vector that is adjusted in such a way that the resulting model output error is minimized.

As a result of this, it may not truly reflect the dead-time in the sense of its definition. It can be seen from Examples 2.2 and 2.3 that dead-time is used for minimizing the model error particularly in the transient period. In the case of Example 2.3, where dead-time of the original system is zero, even good models have some nonzero dead-time, for example 0.1[s]. In this way, the obtained dead-times become an integral part of the model dynamics, which means that dead-time obtained in one model cannot in general be re-used in some other model of the same or different order. This is despite the fact that the system dead-time is an intrinsic characteristic of a system as a whole and as such should be invariant to any particular model representation of the same order (Tuch *et al.* 1994). In short, the obtained dead-times are adjusted to this particular set of model parameters. In the light of this, the interpretation of the dead-time will have an impact on the form of the interaction elements. Depending on the interpretation given to the parameters corresponding to the dead-times, various strategies can be devised regarding the roles of dead-times in the second phase of the hierarchical identification procedure. These strategies are now discussed.

Firstly, dead-times can be considered in Phase 2 of identification as free parameters that will be adjusted to the change in the subsystem dynamics that will occur due to the output feedback. This approach will enable greater flexibility of the optimization procedures, but at the cost of a higher number of parameters.

In the second case, if we accept that dead-times do not precisely describe the true system dead-times, but are just some parameters of the composite model, we will then have some freedom to choose the role of dead-times in such a way to make the composite model simpler. However, it must be borne in mind that insertion of a pure dead-time in a feedback loop may cause oscillatory behaviour and even instability of the closed loop systems. This approach was investigated in Janković and Bajić (1996a). As a starting point, subsystem dead-times were all fixed to the same value. The choice of the value to which all subsystem dead-times are fixed is very important. On one hand, by fixing these values we increase the composite model order as dead-times can be approximated

by an infinite number of terms. On the other hand, when fixed, these values must not be too large because such dead-times cannot be compensated during the optimization in a direction towards the left on the time axis. Therefore, if the values for dead-times estimated in the Phase 1 are significant we should replace them with some smaller value which will be the dead-times of subsystems in Phase 2. For this reason it is necessary to introduce a new parameter, dead-time of the overall composite model which is included in the adjustable parameter set in Phase 2. This parameter takes the role of the SISO model's dead-time. As a conclusion, in this approach subsystem dead-times serve to increase the composite model order, while the composite model's dead-time L (estimated in Phase 2) has to capture the SISO systems dead-time. As there is no clear guidance on how to select the value for the dead-time of the subsystems in the second phase, extensive simulations were performed. The best results were achieved by fixing the subsystem dead-times to the value L_I such that $L_I < \bar{L}_i,\ i = 1, 2, ..., N$. The usual value chosen is of the form

$$L_I \leq \frac{\bar{L}_i}{p} \tag{3.18}$$

for some integer p. Unfortunately, there is no clear rule as to which value parameter p should be fixed. However, simulations suggest that p should be chosen in such a way that the value of L_I is small. For example, a good choice for p could be 10.

Another possibility is to retain dead-times obtained in the first phase of identification, in which parameters of the subsystems are estimated. As already mentioned, these parameters are specially adjusted together with the other model parameters so that the resulting model error is minimized. However, when the model dynamics change in the second phase, these original values for dead-times are more than likely to negatively affect the I/O characteristics of the composite model, as they are adjusted specifically to the subsystems' dynamics.

Finally, subsystem dead-times can all be fixed to zero in Phase 2. There are two possible reasons for this. One problem with the fixed non-zero dead-times is that the optimization procedures cannot compensate for the values of dead-time in the negative

direction of time axis. This means that if dead-times are fixed to some level that may be too high, compensation of this will not be possible. Secondly, dead-times can be approximated by high-order linear models with no dead-time. This follows from the fact that dead-times can be approximated by rational expressions, for example by Padé approximations (De Carvalho 1993). Principally, a similar effect obtained by insertion of small dead-times in the feedback loop can be now achieved by using higher order subsystems with no dead-time. The composite model dead-time L will have the same role of capturing the SISO system's dead-time. This approach yields the following model: if by $\mathbf{y}_I = \mathbf{y} = [y_1 \ ... \ y_N]^T$ we denote the vector of subsystems outputs and by $\mathbf{m} = [m_1 \ ... \ m_1]^T$ the vector of feedback signals, then we can write

$$\begin{cases} y = e^{-Ls}\mathbf{y}^* \\ \mathbf{y}^* = \mathbf{c}_{cm}\mathbf{y} \\ y_i = G_i(s)u_i \\ u_i = u + m_i \\ \mathbf{m} = \mathbf{y}_I\mathbf{A}_{cm} \end{cases} \tag{3.19}$$

where $\mathbf{c}_{cm}$ is a $1 \times N$ vector. This model is presented in Figure 3-7.

Examples 3.4 and 3.5 deal with selection of the dead-times for the feedback loop.

A few words regarding the matrix $\mathbf{A}_{cm}$ and vector $\mathbf{c}_{cm}$. Since subsystem output feedback influences the internal couplings of the subsystems, matrix $\mathbf{A}_{cm}$ is mainly responsible for positioning poles of the equivalent transfer function of the composite model in Phase 2. On the other hand, vector $\mathbf{c}_{cm}$ couples subsystem outputs into the composite model output and can thus be considered as being mainly responsible in Phase 2 for placing zeros of the equivalent transfer function representation of the composite model (see Aström *et al.* 1984).

We can now state the hierarchical identification procedure for SISO systems in the form of the following algorithm:

Phase 1.

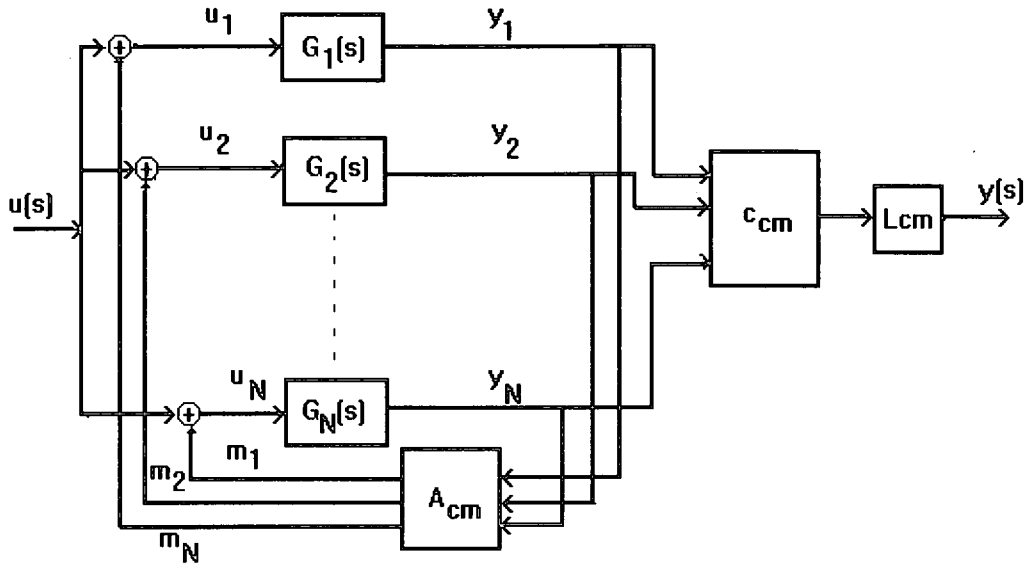


Figure 3-7: Composite model with no dead-times on the feedback loop

1a Obtain the input-output sequences;

1b Prepare I/O sequences to correspond to initially relaxed system by bringing the input and output to level zero before input is applied;

1c Identify the parameters of N transfer functions for models of order i, $i+1$, $i+2$, . . ., $i+N-1$, according to Equation (3.16). As all of the subsystems share the same local (I/O) data this step can be performed in parallel.

Phase 2.

2a Use the transfer functions obtained in Phase 1 to form a composite model given by (3.17) or (3.19).

2b In this step we estimate parameters of subsystem interactions (matrix $\mathbf{A}_{cm}$, vector $\mathbf{c}_{cm}$ and the composite model delay L_{cm}) in such a way that the composite model's output error criterion is minimized.

3.4 Benefits of Hierarchical Identification

The main goal of hierarchical identification is to reduce the number of parameters that simultaneously enter optimization routines by utilizing the decomposition principle. It would now be important to study the potential benefits of such decomposition and of the associated hierarchical identification procedure presented in the previous section. The composite model is relatively complex and will not bring numerical advantages over the ordinary one-shot identification methods in all cases. The aim of this section is to compare these two approaches in order to establish if and when the hierarchical identification, as introduced in the previous section, will bring some numerical benefits and how significant those benefits may be. The analysis consist of two parts: in the first we study the possible numerical benefits of hierarchical identification approach, and in the second we study the ability of the output feedback applied in models (3.17) and (3.19) to improve the model's dynamics.

3.4.1 Numerical Benefits

The numerical complexity of any problem normally increases much faster than the problem size itself. There are many examples of that. For instance, if the parameters are to be estimated using the LSE, this would require the inversion of an $n \times n$ matrix, where n is the number of parameters to be determined. The number of required instructions for matrix inversion increases faster than the matrix size. It was already mentioned that the number of instructions per iteration for quasi-Newton optimization methods is proportional to the square of the number of parameters. Example 2.3 from Section 2.2.4 is particularly illustrative in this respect and shows that numerical difficulties lead to a much higher number of instructions than suggested by theory. The choice of initial values for parameters also becomes more difficult as the dimensionality of the problem increases. It is therefore clear that it is important to resolve problems associated with LSS in a sequence of smaller size problems rather than in a single, large size problem.

If a model is given in the operator form by the transfer function description

$$y = G(s)e^{-Ls}u = \frac{B(s)}{A(s)}e^{-Ls}u = \frac{b_m s^m + b_{m-1}s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0}e^{-Ls}u. \tag{3.20}$$

then the corresponding parameter vector θ has $2n+1$ components:

$$\theta = [a_{n-1}\ a_{n-2}\ \ldots\ a_1\ a_0\ b_{n-1}\ b_{n-2}\ \ldots\ b_1\ b_0\ L] \tag{3.21}$$

where n is the order of the model and L the apparent dead-time of the system. For the transfer function models given by (3.20) we can write that the number of parameters and the model order satisfy the relation

$$\Theta(G, n) = 2n + 1. \tag{3.22}$$

The relation is linear. For the composite model (CM) described in the previous section we see that the total number of parameters is given by

$$\Theta(CM, n) = \sum_{i=1}^{N} \Theta(G_i, n_i) + \Theta(IM, N). \tag{3.23}$$

In this notation $\Theta(CM, n)$ is the number of parameters of the n-th order composite model; $\Theta(G_i, n_i)$ is the number of parameters for the i-th subsystem's transfer function, where n_i is the order of i-th subsystem, and $\Theta(IM, N)$ is the number of parameters in interaction module (IM). It is assumed that there are N subsystems in the composite model. In all the cases, the effect of dead-times in feedback loop on the composite model order will not be taken into account. Once fixed, these dead-times do not change the number of parameters that optimization procedures will tune. Since $\Theta(G_i, n_i)$ is given by (3.22), what is left is to estimate the last term of (3.23). For the interaction module, if the subsystem's dead-times are fixed, we can write

$$\Theta(IM, N) = N^2 + N + 1 \tag{3.24}$$

where the N^2 term comes from the matrix $\mathbf{A}_{cm}$ and the term N comes from vector $\mathbf{c}_{cm}$; the last parameter is the composite model dead-time. If parameters corresponding to subsystem dead-times are not fixed, then the total number of parameters is

$$\Theta(IM, N) = N^2 + 2N + 1. \tag{3.25}$$

To get the expression for the number of parameters of the composite model we note that for the number of parameters for each of the subsystems, without loss of generality, the following ordering relation holds

$$\Theta(G_1, n_1) \leq \Theta(G_2, n_2) \leq \ldots \leq \Theta(G_N, n_N) \tag{3.26}$$

as subsystems can always be indexed in an appropriate way. At this point it should be noticed that all terms of the form $\Theta(G_i, n_i)$ can be estimated in parallel. Because of this, the numerical complexity will be determined by the term $\Theta(G_N, n_N)$ only, which is the consequence of the relation (3.26). In other words, if all of transfer functions G_1, G_2, ..., G_N are determined in parallel, it is reasonable to assume that the identification of the highest-order subsystem's transfer function, G will require the longest period of time. Therefore, in our analysis of the estimated time required for identification by hierarchical identification method, we are only interested in the time taken for identification of G_N, as identification of G_1, G_2, ..., G_{N-1} is assumed to be completed before the completion of identification of G_N. For this reason, we only take into account the number of instructions required for identification of G_N. In light of this, the maximal number of parameters that will, roughly speaking, determine the computation complexity in hierarchical identification, where parallel processing is taken into account, is given by

$$\Theta(CM, N) = \Theta(G_N, n_N) + \Theta(IM, N). \tag{3.27}$$

Once again, this is a result of our concern only with the highest order model in Phase

1, as others are estimated in parallel. In this expression, $\Theta(IM, N)$ is the number of parameters that are required to be estimated during Phase 2. It is worth noting that the composite model is of the order $\sum_{i=1}^{N} n_i$. Again, the effect of dead-times on the model order is ignored. This will be discussed in the next section.

A particular identification procedure would be more efficient than some other identification procedure if it requires estimation of a smaller number of parameters for identification of a system by a model of the same order and similar precision. We can now compare the estimated time required for identification of a linear model's transfer function to the hierarchical identification by composite models. The important consideration is related to the number of subsystems. From (3.24) it follows that the required number of parameters in Phase 2, $\Theta(IM, N)$, depends only on the number of subsystems and not on the order of any particular subsystem model and that this number is proportional to the square of the number of subsystems. On the other hand, the order of the composite model also increases with the number of subsystems. It would be interesting to find out the increase of the composite model order as a result of an increase of the number of the subsystems. This increase cannot be estimated easily for the general case, as it depends crucially on the form of the subsystem models. In order to simplify the case, we will assume that models are given in the form of transfer functions in such a way that the order of each of the subsystems G_i is greater by one then the order of the subsystem G_{i-1} :

$$\text{order}(G_i)=\text{order}(G_{i-1})+1. \tag{3.28}$$

As in the relation (3.26), n_1 will denote the order of the lowest order subsystem. In this case the maximal number of parameters, as well as the order of the composite model, are well defined and can easily be calculated.

In order to compare the relative computational benefit of such decomposition, we must somehow relate the computational effort to the problem size. In our case, computational effort would be translated into the number of processor instructions required for identification of the given model and the problem size will be defined in terms of the

number of model parameters. Alternatively,problem size can be expressed in terms of the model order, as there is a relation between the maximal number of model parameters and its order. For the purpose of this analysis, we will (very) conservatively assume that the number of instructions required to solve a typical optimization problem will grow proportionally to the square of the number of parameters. The time necessary to complete identification will be taken to be directly proportional to the required number of instructions, i.e. no operating system overhead will be taken into account. We here analyze only the optimization part of the problem, i.e. the numerical effort associated with calculating the output error function of the model. Computational effort associated with simulation which is necessary to evaluate the model error criterion has not been taken into account as this criterion depends very much on the criterion itself, simulation methodology and the length of input-output data. It would be, however, necessary to keep in mind that such overhead exists. For the above mentioned reasons the analysis that follows will be qualitative in nature.

Under these assumptions, the estimate of the number of instructions required to identify the composite model by the proposed two step identification procedure can be calculated as follows. The order of a composite model (CM) consisting of N subsystems is given as

$$\text{order}(CM) = \sum_{i=1}^{N} n_i \tag{3.29}$$

where n_i is the order of the $i-th$ subsystem. Under the assumption (3.28) and $n_1 = 1$, the composite model order is given as

$$\text{order} = \sum_{i=1}^{N} n_i = \frac{N(N+1)}{2}. \tag{3.30}$$

For the first step, the number of instructions will be limited by the number of parameters of the highest order subsystem N, and that number of parameters is $2N + 1$, including the apparent dead-time of the overall system. For the second phase this number of parameters is given by (3.24) and is $N^2 + N + 1$, so the total number of instructions for

both steps is

$$\text{ins} = (2N+1)^2 + (N^2+N+1)^2. \tag{3.31}$$

Again, in relation (3.31) we have not taken into account the number of instructions required for identification of transfer functions G_1, G_2, ..., G_{N-1}, as already explained. For one-shot estimation of a transfer function the number of instructions is easy to estimate and is given by

$$\text{ins} = (2n+1)^2 \tag{3.32}$$

where n is the model order.

In the case where $n_1 = 2$ we have

$$\text{order} = \sum_{i=1}^{N} n_i = \frac{N(N+3)}{2} \tag{3.33}$$

and the corresponding number of instructions for the highest order subsystem and the interaction module will be

$$\text{ins} = (2N+3)^2 + (N^2+N+1)^2. \tag{3.34}$$

For the general case, when $n_1 = k$, the model order is

$$\text{order} = \sum_{i=1}^{N} n_i = \frac{N(N+2k-1)}{2}. \tag{3.35}$$

and the number of instructions for the highest order subsystem and the interaction module will be

$$\text{ins} = (2N+2k-1)^2 + (N^2+N+1)^2. \tag{3.36}$$

The number of parameters for the second phase is independent of the value for n_1. The efficiencies of the two identification methods (one-shot and hierarchical approach) can be compared by creating a ratio R between the number of instructions required to identify a model of the same order by hierarchical approach and one-shot approach. These ratios

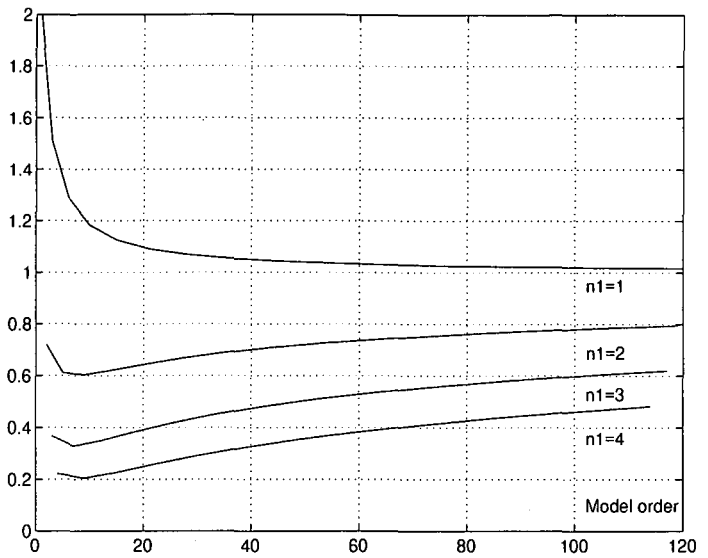


Figure 3-8: R vs. model order

are plotted in Fig. 3-8. In Figure 3-9, this ratio is plotted as a function of the number of subsystems.

Several conclusions can be drawn from Figure 3-8. Firstly, if $n_1 = 1$, hierarchical identification procedure is less efficient than the classic, one-shot approach. Starting from $n_1 \geq 2$, hierarchical identification becomes more efficient. This is not a particularly restrictive condition since the *raison d'être* for composite models is that they are intended for modelling of a process by high-order models. On the other hand, the assumption of quadratic increase in computational effort due to the increase in the number of parameters is too conservative and in reality this condition should not be taken too strictly, as we shall illustrate later. Secondly, there is a particular model order for which the benefit over the one-shot approach is maximal, after which this benefit (slowly) deteriorates. In fact, it can be shown that if $n_1 = k$, then the order of the highest order subsystem will be $N + k - 1$, resulting in the number of instructions required by hierarchical identification becoming

$$\text{ins}_{HI}(k, N) = (2N + 2k - 1)^2 + (N^2 + N + 1)^2. \tag{3.37}$$

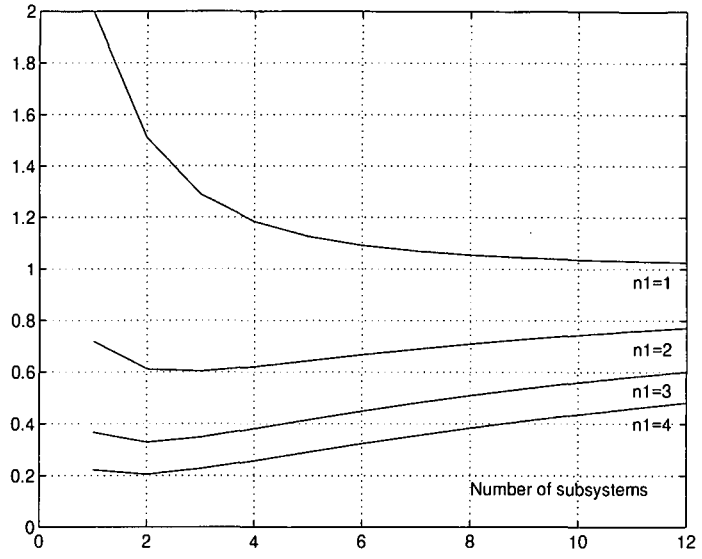


Figure 3-9: R vs. number of subsystems

For a transfer function model of the same order, the number of instructions required in the one-shot approach will be

$$\text{ins}_{OS}(k,N) = [2(N(N+2k-1)/2+1]^2 = (N(N+2k-1)+1)^2. \tag{3.38}$$

From these two equation follows that

$$\lim_{N\to\infty} \frac{\text{ins}_{HI}(k,N)}{\text{ins}_{OS}(k,N)} = 1 \tag{3.39}$$

This means that the numerical advantages of the hierarchical identification procedure over one-shot identification will disappear for very high model orders. However, in reality, we never go so far in the choice of the model order. Thirdly, we can conclude that there is an optimal range of number of subsystems for any choice of n_1.

In Figure 3-9 the efficiency of hierarchical (parallel) identification over the one-shot identification is shown as a function of the number of subsystems i.e. $\frac{\text{ins}_{HI}(k,N)}{\text{ins}_{OS}(k,N)}$ is plotted for $k = 1$, 2, 3 and 4 and $N = 12$. As already mentioned, from Fig. 3-8 it can be seen

that the relative advantage of parallel identification over one-shot identification drops as the number of subsystems increases. On the other hand, the order of composite model can be increased by using subsystems of higher orders, instead of using larger numbers of low-order ones. It would therefore appear that the number of subsystems should be kept at a minimum, while subsystems should be of higher orders. There is, however, a problem with such an approach. The benefit of interactions by an output feedback is reduced if the number of subsystems is reduced. The ability of subsystem interactions to optimize the model dynamics depends crucially on the number of model eigenvalues that it can modify independently from each other. This number is equal to the number of output feedbacks which is then equal to the number of subsystems. Therefore, the higher the number of subsystems, the greater the possibility that the interaction module will optimize the composite model. Unfortunately, the portion of an interaction module that is mainly responsible for the changes of the composite model dynamics (the matrix **A** from 3.17) is the main contributor to the number of parameters in hierarchical identification. It can be concluded that the optimal decomposition will require the number of subsystems to be chosen somewhat to the right of the minimum point in Fig. 3-8. It can be seen from this graph that the number of subsystems should be somewhere between 3 and, approximately, 8. This does not decrease the relative computational benefit and still enables the algorithm to change a significant number of system eigenvalues.

Finally, should the composite model with subsystem dead-times that are not fixed be used instead of the above analyzed one, its parameter estimation efficiency will be worse. The analysis of benefits for this case is analogous to the one above.

In conclusion, hierarchical identification procedure my be beneficial when compared to the one-shot identification approach even under very mild assumptions regarding the relation between the size of a problem and the required computational effort. The procedure is illustrated with a few examples in the next section, after which further analysis of the hierarchical identification method will be done.

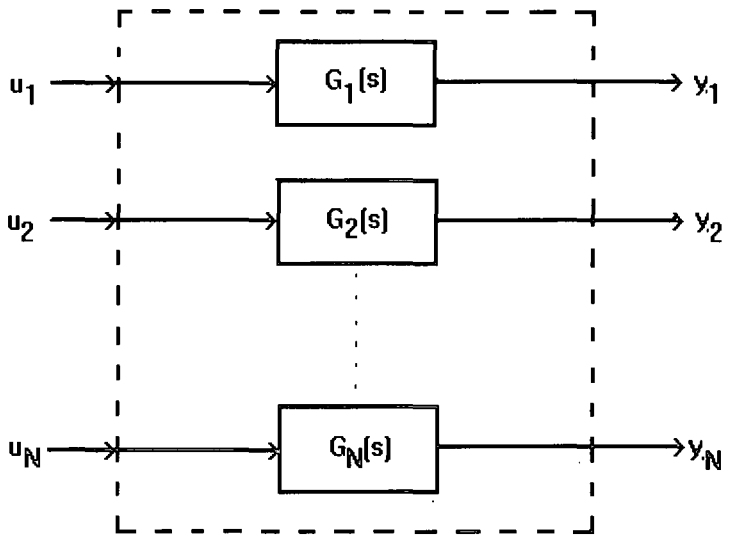


Figure 3-10: Uncoupled multivariable system

3.4.2 Analysis of Composite Model Dynamics

In addition to the analysis of benefits in reducing the optimization complexity brought by hierarchical identification, it is also necessary to assess the ability of such approach to modify the composite model dynamics. This ability must be sufficient in order for the algorithm to be able to adequately match the input-output set by procedures that constitute Phase 2. It is clear that the larger the number of eigenvalues that can be modified freely, the better the chance that the composite model will be good. The objective of the analysis that follows is to establish the relation between the number of subsystems in model (3.19) and the number of model eigenvalues that can be assigned to some given set of values. In order to do this, we observe the composite model consisting of N subsystems, shown in Figure 3-10. Here the signal $\mathbf{u} = [u_1, \ ..., \ u_N]$ is an input signal and $\mathbf{y} = [y_1 \ ... \ y_N]$ is the output vector. For this parallel connection we can derive the state-space equation in the following way. For each of the transfer functions, there exists a state-space equation of the form (2.13), where the matrix $\mathbf{A}_i$ is to be taken in

the controllable canonical form as in (2.59):

$$\begin{cases} \dot{\mathbf{x}}_i = \mathbf{A}_i\mathbf{x}_i(t) + \mathbf{b}_i u_i(t) \\ y_i = \mathbf{c}_i\mathbf{x}_i(\mathbf{t}) \end{cases} \tag{3.40}$$

where index i denotes $i-$th subsystems and vectors $\mathbf{x}$, $\mathbf{b}$ and $\mathbf{c}$ are $n_i \times 1$, $n_i \times 1$ and $1 \times n_i$, respectively and matrix $\mathbf{A}_i$ is $n_i \times n_i$. Vector $\mathbf{b}_i$ is, according to (2.59) of the form $[0,\ 0,\ ...,\ 0,\ 1]^T$. For the composite model from Fig. 3-10 consisting of parallel connections of uncoupled subsystems we can write

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y} = \mathbf{C}\mathbf{x}(\mathbf{t}) \end{cases} \tag{3.41}$$

In state-space equations (3.41) we have

$$\mathbf{x}(t) = \begin{pmatrix} \mathbf{x}_1(t) \\ . \\ . \\ . \\ \mathbf{x}_N(t) \end{pmatrix}, \qquad \mathbf{y}(t) = \begin{pmatrix} y_1 \\ . \\ . \\ . \\ y_N \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} \mathbf{b}_1 & & & \\ & \mathbf{b}_2 & & \\ & & . & \\ & & & \mathbf{b}_N \end{pmatrix}$$

where vectors $\mathbf{x}_i(t)$ and $\mathbf{b}_i$ belong to the i-th subsystem, and scalar y_i is the output of the i-th subsystem. State vector $\mathbf{x}(t)$ is clearly an aggregation of the subsystem state vectors. As subsystems are uncoupled, we can write for the matrices $\mathbf{A}$ and $\mathbf{C}$:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & & & \\ & \mathbf{A}_2 & & \\ & & . & \\ & & & \mathbf{A}_N \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} \mathbf{c}_1 & & & \\ & \mathbf{c}_2 & & \\ & & . & \\ & & & \mathbf{c}_N \end{bmatrix} \tag{3.42}$$

Empty spaces in these matrices denote zero submatrices. The matrix $\mathbf{A}$ is $n \times n$, where n is the order of the composite system, $n = \sum_{i=1}^{N} n_i$. The matrix $\mathbf{C}$ is a $N \times n$ and matrix $\mathbf{B}$ is $n \times N$. Matrices $\mathbf{A}_i$, $\mathbf{b}_i$ and $\mathbf{c}_i$ correspond to those in (3.41).

In order to study the effect of the output feedback on positioning of the system eigenvalues, we will assume that for some given model order there exists an optimal model in the sense that it represents the true model of a system to be identified from its I/O data. The task of the output feedback in the model (3.41) is to match the optimal model's eigenvalues as closely as possible. It is therefore necessary to assess how closely we can assign the composite model eigenvalues to those belonging to the optimal model. Although output feedback will change the positions of all eigenvalues of the composite model, the question remains how closely they can approach any given set of values. This is very difficult to assess for the general case. Some of the eigenvalues can be placed exactly to certain desired positions, while the others cannot be placed exactly, but within some distance from the desired positions. In order to assess the ability to assign eigenvalues by output feedback of any composite model to some given set of values (we call this flexibility), we introduce the following definition.

Definition 1 Two composite models of the same order are called equally flexible if the number of their system eigenvalues that can be assigned to some predetermined value is the same.

We now consider the case of uncoupled system given by (3.40) - (3.42). In order to find the number of eigenvalues that we can assign by the incomplete state feedback (the vector $\mathbf{y}$) in this case, we need the following lemma.

Lemma 1 The composite model given by (3.41) and (3.42), created in the way described above is always controllable and observable.

Proof We first note that dynamic equations of each of the subsystems in the (3.40) can be brought into the controllable canonical form (2.59) or into observable canonical form (2.60) as it is directly derived from the transfer function. The transfer function description is known to describe only the controllable and observable part of the system

(Chen 1984, p.205). The lemma is obvious because the system (3.41) consists of uncoupled (independent) subsystems each of which is controllable and observable. We first prove that the system (3.41) is controllable and we form the controllability matrix

$$\mathbf{U} = \left[\mathbf{B}\ \mathbf{AB}\ \mathbf{A}^2\mathbf{B}\ ...\ \mathbf{A}^{n-1}\mathbf{B}\right] \tag{3.43}$$

where $n = \sum_{i=1}^{N} n_i$. To show that system (3.41) is controllable, it is necessary that $rank(\mathbf{U}) = n$ holds. The proof is lengthy but straightforward, and it will be sketched as follows. Because of the structure of matrices given in (2.59), matrix B can be formed as

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & . & . & . & 0 \\ . & . & & & & . \\ 0 & 0 & & & & . \\ 1 & 0 & & & & . \\ 0 & 0 & & & & . \\ . & . & & & & . \\ 0 & 0 & & & & 0 \\ . & 1 & & & & . \\ . & . & & & & . \\ . & . & & & & . \\ 0 & 0 & . & . & . & 1 \end{bmatrix} \tag{3.44}$$

It is immediately seen that this matrix consist of N linearly independent columns. In the next step, we consider the following product

$$\mathbf{AB} = \begin{bmatrix} \mathbf{A}_1\mathbf{b}_1 & & & & \\ & \mathbf{A}_2\mathbf{b}_2 & & & \\ & & . & & \\ & & & . & \\ & & & & \mathbf{A}_N\mathbf{b}_N \end{bmatrix} \tag{3.45}$$

Each of the products $\mathbf{A}_i\mathbf{b}_i$ is a vector of the form [0 0 ... 0 1 $*$], where $*$ denotes any non-zero element, as $\mathbf{A}_i$ and $\mathbf{b}_i$ are given by (2.59). The product (3.45) becomes

$$\mathbf{AB} = \begin{bmatrix} 0 & 0 & . & . & . & 0 \\ . & . & & & & . \\ 1 & 0 & & & & . \\ * & 0 & & & & . \\ 0 & 0 & & & & . \\ . & . & & & & . \\ 0 & 1 & & & & 0 \\ . & * & & & & . \\ 0 & 0 & & & & 0 \\ . & . & & & & . \\ . & . & & & & 1 \\ 0 & 0 & . & . & . & * \end{bmatrix}. \tag{3.46}$$

These columns are again linearly independent. Moreover, these columns are linearly independent from the columns of matrix $\mathbf{B}$ given by (3.44). In order to evaluate products $\mathbf{A}^p\mathbf{B}$, we note that for $n_i \times n_i$ matrix $\mathbf{A}_i$ as in the (2.59) we have

$$\mathbf{A}_i^2 = \begin{pmatrix} 0 & 0 & 1 & . & 0 \\ 0 & 0 & 0 & . & 0 \\ . & . & . & . & . \\ 0 & 0 & 0 & . & 1 \\ -a_0 & -a_1 & -a_2 & . & -a_{n-1} \\ * & * & * & . & * \end{pmatrix} \tag{3.47}$$

where, as usual, x denotes position of a non-zero element. Each subsequent power of $\mathbf{A}$ will have an additional row of non-zero elements, until $\mathbf{A}_i^i$ which will have no fixed zeros

or ones. On the other hand we have

$$\mathbf{A}^p = \begin{bmatrix} \mathbf{A}_1^p & & & \\ & \mathbf{A}_2^p & & \\ & & . & \\ & & & \mathbf{A}_N^p \end{bmatrix}, \qquad \mathbf{A}^p\mathbf{B} = \begin{bmatrix} \mathbf{A}_1^p\mathbf{b}_1 & & & \\ & \mathbf{A}_2^p\mathbf{b}_2 & & \\ & & . & \\ & & & \mathbf{A}_N^p\mathbf{b}_N \end{bmatrix}$$

Bearing in mind (3.47), we can conclude that $\mathbf{A}_i^p\mathbf{b}_i$ will be $[0 \ldots 1 \ * \ *]$ for $p = 2$, $[0 \ldots 1 \ * \ * \ *]$ for $p = 3$, etc. Repeating the same analysis as for the case $p = 1$, we get that columns of $\mathbf{A}^p\mathbf{B}$ are linearly independent and also linearly independent from the columns of $\mathbf{B}$, $\mathbf{AB}$, ... $\mathbf{A}^{p-1}\mathbf{B}$. Eventually, for $p = N - 1$ we will span R^n which means that $rank(\mathbf{U}) = n$. The same analysis can be repeated to prove that the rank of the observability matrix for the system (3.41) in also n. The analysis is simplified if the subsystem dynamic equations are now written in observable canonical form (2.60). This can always be done when dynamic equations are realized from the proper transfer function.

Alternatively, the lemma can be proved by applying Kalman's canonical decomposition theorem (Kalman 1963, Chen 1984, De Carvalho 1993) and the principle of mathematical induction.

Under the full controllability and observability assumption we now apply the well-known fact, independently reported by Kimura (1975) and Davison and Wang (1975) (see also Rajagoplan 1984 and Lee *et al.* 1994) that the output feedback can asymptotically assign $\max(\min(rank(\mathbf{B}) + rank(\mathbf{C}) - 1, n))$ eigenvalues of the model, where n is the model order, providing that any complex eigenvalues come in complex conjugate pairs (Davison 1970, Davison and Chaterjee 1971). In our case this means that the procedure proposed in this chapter can assign at most $2N - 1$ model eigenvalues to some arbitrary values. Nothing can be said about the positions of the remaining $n - 2N + 1$ eigenvalues, although in some cases sufficient conditions have been found under which they can be

confined to a certain area of the complex plane (see Vardulakis 1975).

3.5 Implementation and Examples

We will now attempt to apply the proposed procedures for the identification of the autopilot model described in Example 2.3 in Section 2.2.4. We will use the same I/O sequence to identify different composite models. Various approaches with regard to subsystem dead-times will be used. Common to all examples is that the order of the lowest order subsystem is one. This is done in order to illustrate that in practice, the scope of applicability of hierarchical identification will often be much broader than suggested by the analysis from the previous section. This is because the assumption of quadratic increase in computational effort as the number of free parameters in the optimization routines increase is often too conservative. Once again, as it is impossible to accurately link the computational effort with the problem size, achieved savings in computational effort differs from the results achieved in our analysis in Section 3.4.1, although the prediction that the number of instructions necessary for identification will decrease did materialize.

It should be noted that no suitable parallel processing computer was available for testing of the method. These tasks were performed on the same computer in sequence. However, due to the high level of independence of identification tasks for each of subsystems, the author can see no conceptual difficulties in extrapolating achieved results to the case of computations on a parallel processing computer.

Example 3.1. The 3-channel autopilot model presented in Example 2.3 will now be identified by the hierarchical identification approach. The building blocks for the composite model (the subsystems) will be transfer functions of orders 1, 2 and 3 obtained in Example 2.3 and given in Appendix B. The input test signal is the unit step. Subsystem dead-times are taken to be free parameters in the optimization procedure. The obtained model has an IAE of only 0.37, which is better than the transfer function model of sixth order, which had IAE of 0.45. The measured number of floating point operations in

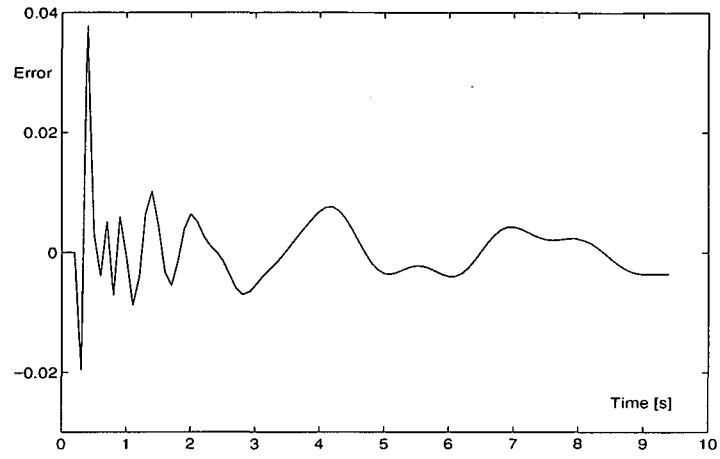


Figure 3-11: Output error for the model in Example 1

hierarchical approach for both Phase 1 and Phase 2 was 39334000 compared to 188106000 in the case of the one-shot approach. We can also see the way in which the optimization procedure has modified the initial set of subsystem dead-times, which are 4.45 [s], 0.71 [s] and 0.23 [s]. These values obviously do not correspond to the true dead-time of the original model. Model error is plotted in Figure 3-11 and the frequency response in Fig. 3-12. Just as in the case of the sixth-order model from Example 2.3, the frequency characteristic is very good in the lower frequency range, and deteriorates in the higher frequency range.

Example 3.2. The system to be identified in this Example is the same as in the previous one and the input signal is again the unit step. In this case the composite model that was used is the one given by (3.19) in which subsystem dead-times are fixed to zero. The IAE obtained for this case is 0.45 and the measured number of floating point operations for both phases of the hierarchical identification procedure was 32078000. The model error is shown in Figure 3-13 and frequency response in Figure 3-14. When compared to the composite model from Example 3.1 this one is slightly less accurate, but requires less computational effort as the number of parameters that had to be determined

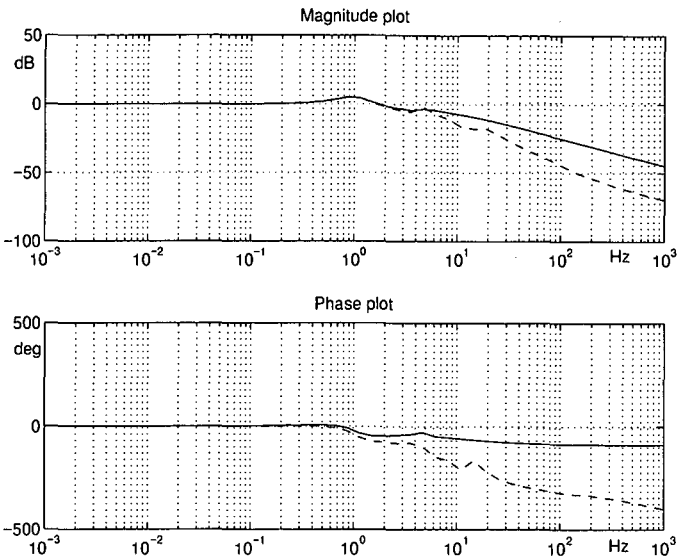


Figure 3-12: Frequency response for the model in Example 1

is smaller (subsystem dead-times are fixed in this example). Compared to the sixth-order model from the Example 2.3, this model has identical IAE, but somewhat worse frequency response in middle range of frequencies.

Example 3.3 As building blocks for our composite model we use models obtained from the "white noise" signal in Example 2.4. The composite model used for hierarchical identification is given by (3.19) and the input signal is the same white noise sequence as in Example 2.4. The frequency response for this model is presented in Fig. 3-15. The $\sqrt{ISE}$ of this model (for the same input as in Example 2.4) is 0.0265, slightly better than in the case of fourth and fifth order models from Example 2.4. However, when we compare the frequancy response of this model and to those obtained in Example 2.4 and whose frequency responses are shown in Figs. 2.29 and 2.30, it can be seen that now both low and high frequency responses are good. In the middle range of frequencies, the composite model has slightly worse magnitude and slightly better phase characteristics. Hierarchical identification of the composite model (3.17) with dead-times fixed to the

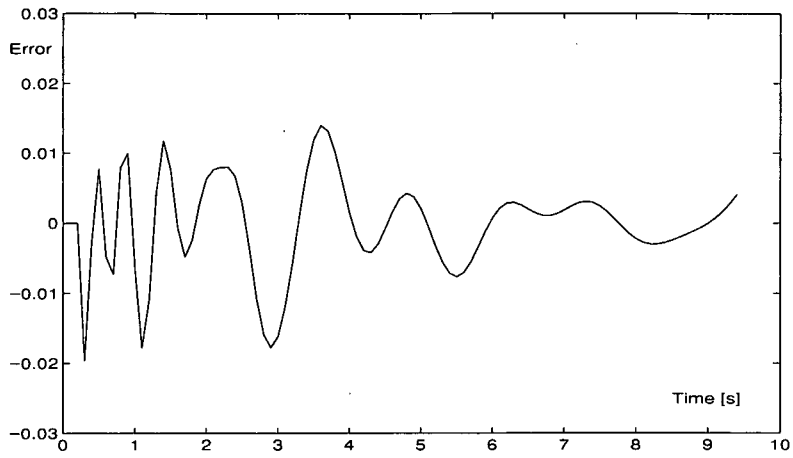


Figure 3-13: Output error for the model with no dead-times in feedback loop

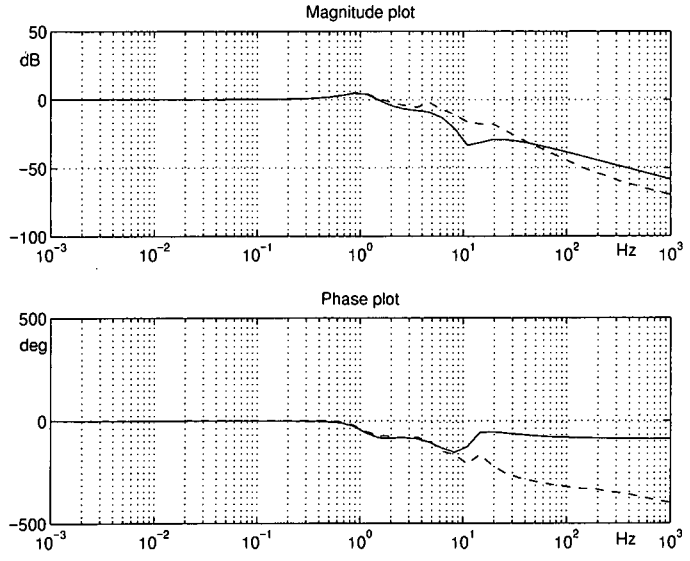


Figure 3-14: Frequency response for the model in Example 3.2

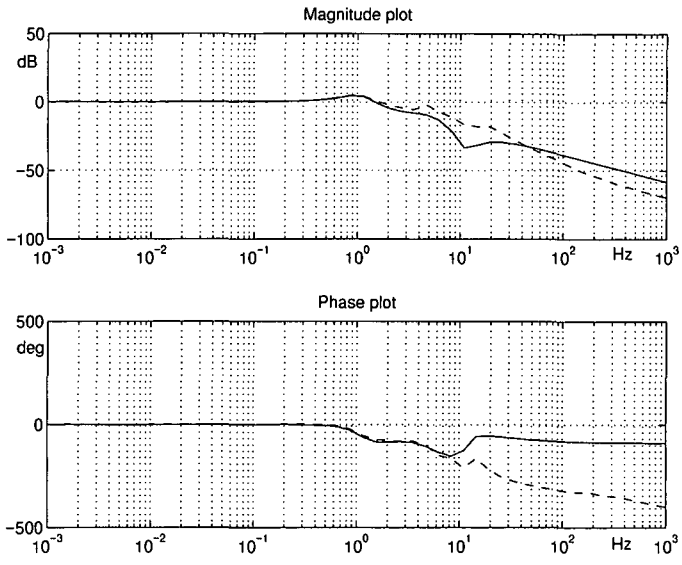


Figure 3-15: Frequency response for the model from Example 3.3

minimal dead-time of all subsystems has not given good results.

Example 3.4 In this example we investigate the effect of presence of fixed dead-times in the output feedback loop in models (3.17) and (3.19). Although we have found in Example 3.3 that small dead-times in the feedback loop did not give good results, we suspect that part of the reason for this is that since the original system dead-time is small (in fact zero), the inability to compensate for larger fixed-dead-times in the composite model will result in model inaccuracies. We hope that when system dead-time is significant, the presence of fixed dead-times in the output feedback loop may improve the model accuracy. The presence of dead-time can be easily established in Phase 1. In order to illustrate the case with significant dead-time and presence of fixed dead-times in the feedback loop, we use the same model as in Example 3.1 but with a dead-time of 1[s]. Various dead-times were inserted in the feedback loop and the results are summarized in Table 3.1.

fixed dead-time	$\sqrt{ISE}$	L_{proc}	total dead-time (L_{proc}+fixed dead-time)
0	0.0483	1.0939	1.0939
0.1	0.0485	1.1144	1.2144
0.25	0.0505	0.9133	1.1633
0.5	0.0502	0.6720	1.1720

Table 3.1:

fixed dead-time	$\sqrt{ISE}$	L_{proc}	total dead-time (L_{proc}+fixed dead-time)
0	0.0318	1.0977	1.0977
0.1	0.0333	1.0559	1.1559
0.25	0.0350	0.9272	1.1772
0.5	0.0484	0.6720	1.1720

Table 3.2:

In this table, L_{proc} is the composite model dead-time. As expected, the sum of dead-times is close to the true dead-time of the original model. The best result, both in terms of the output error and model dead-time has been achieved by the utilization of the composite with no dead-time in the output feedback loop. A small dead-time can be, however, be inserted in the feedback loop in which case the results are still good.

Example 3.5 As the results from Example 3.4 could not be optimal (in the sense that the global minimum has not been found), we take another example to investigate the effect of dead-times in the feedback loop. The model used to generate I/O sequences is taken from *Simulink User's Guide* (The Math Works, Inc 1992) and is of order 42. Dead-time of the model is set to 1[s]. The unit step response of the model is shown in Figure 3-16.

Models of order 1, 2 and 3 are identified from the I/O sequence. They are used in Phase 2 as subsystems, and dead-times of various durations were inserted in the output feedback loop. The results are summarized in Table 3.2.

Output errors for these composite models are shown in Figs. 3-17, 3-18, 3-19 and 3-20. Again, zero dead-time has given the best result, both in terms of output error as

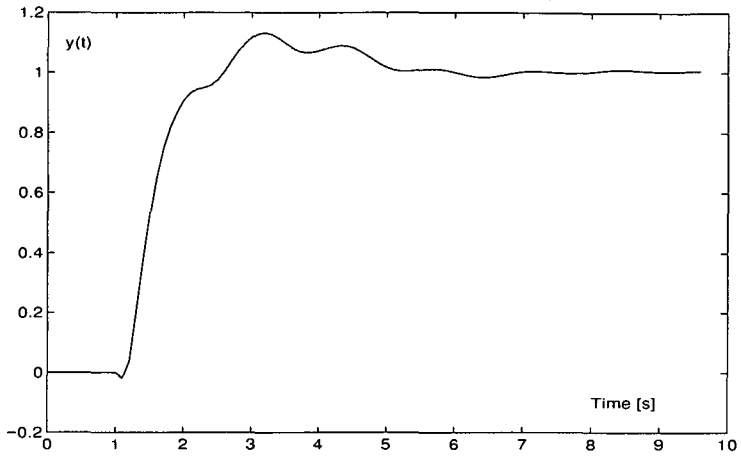


Figure 3-16: Unit step response for the model from Example 3.5

well as estimate of system dead-time. It is noticeable that the output error increases with the value of inserted dead-time. Again, we can conclude that if there is any dead-time in the feedback loop, it should be kept small.

3.6 Further Reduction in the Number of Parameters

It can be seen from (3.24) and (3.25) that the number of parameters in the interaction module is proportional to N^2. This seriously decreases the usability of the algorithm for a larger number of subsystems. In this section we address the problem by using local feedbacks for each of the subsystems and then study the effect of such compensation. The composite model with local output feedbacks is shown in Figure 3-21. From the discussion in section 3.4 it is now easy to conclude that local feedback can assign one eigenvalue (or in this case pole) in each of the subsystems to some predetermined value. Again, nothing can be said about the positions of the remaining $n_i - 1$ poles. So, although output feedback will change the values of all system eigenvalues, assessment of how close they are to some given set of values is again very difficult.

We observe that in the case of the first order model, output feedback can place a pole

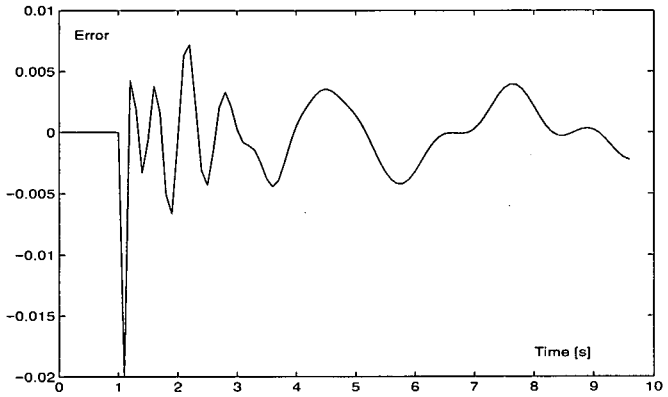


Figure 3-17: Output error for model with dead-time fixed to 0[s]

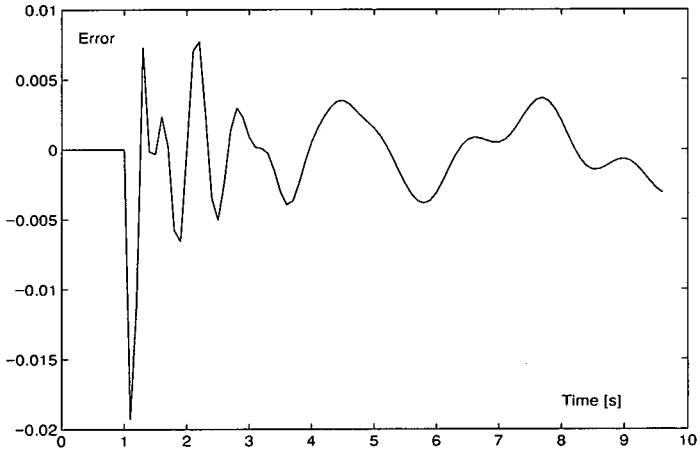


Figure 3-18: Output error for model with fixed dead-time of 0.1[s]

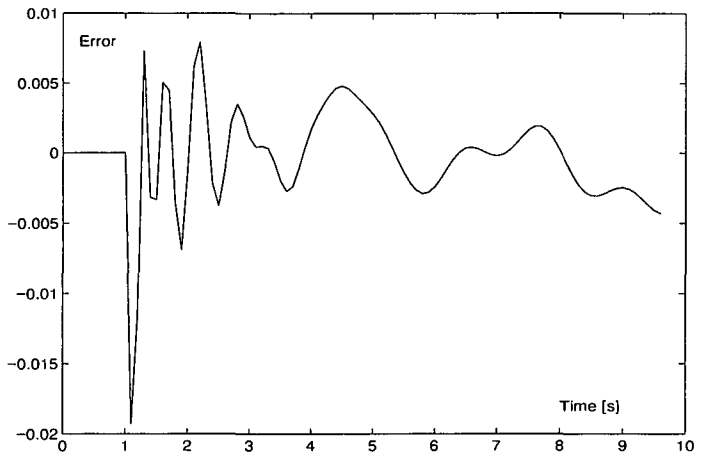


Figure 3-19: Output error for model with fixed dead-time of 0.25[s]

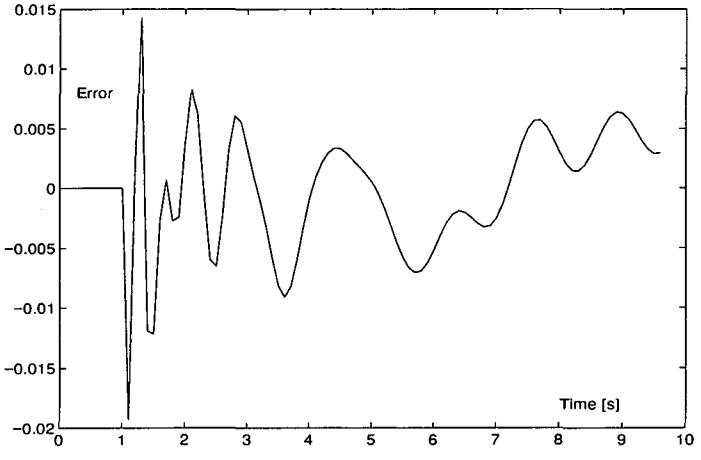


Figure 3-20: Output error for model with fixed dead-time of 0.5[s]

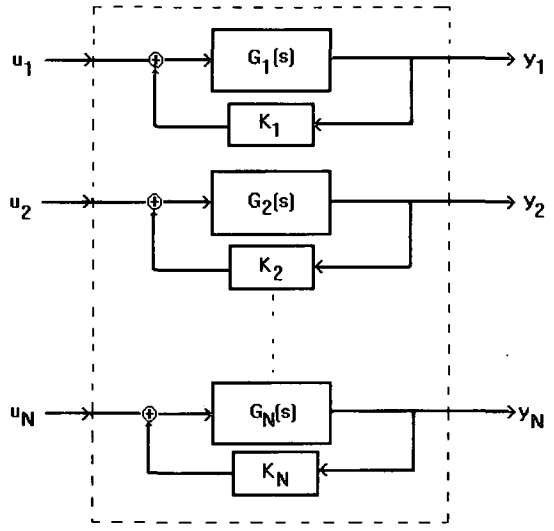


Figure 3-21: Uncoupled composite model with local feedback

at any position on the real axis. In the case where all poles are complex (for even model orders), a single proportional output feedback will not be able to place any of the poles into a predetermined position in the complex plane. This can be compensated for by creating subsystems of odd orders only, as they will always contain at least one pole on the real axis. These poles on the real axis we can then place at an arbitrary position on the real axis.

Therefore, we have established the following theorem for the reduced composite model from Fig. (3-21):

Theorem 1 The reduced composite model (3.41) is less flexible in the sense of Definition 1 than the model (3.17) and has flexibility N if all of the subsystems are represented by transfer functions of odd number order.

The question that arises now is whether the composite model with local feedbacks can be successfully used instead of (3.17), despite the fact it is less flexible than the composite model (3.17). To answer this question, we recall our idea of matching the composite model to the "true" model. Obviously, the smaller the number of eigenvalues that we can assign, the smaller the possibility of matching a model. However, we have already

concluded that the notion of the "true" model is somewhat misleading, which means that although we may not be able to match the "true" model, we may be able to match some other model, which still adequately describes the system dynamics. Therefore, we may be able to find a combination of system poles and zeros that will work well, thus overcoming the problem of inability to place all poles to specific locations. Despite this, the fact remains that only pure real eigenvalues can be arbitrarily placed. There is no way that this scheme will allow placement of complex eigenvalues at desired positions in the complex plane. The most that can be said about these eigenvalues is that they will be manipulated by local feedback together with real eigenvalues, so some acceptable positioning may be achieved. It is worth noting that the composite model given by (3.17) in the previous section suffers from the same problem, albeit to a smaller degree.

We can now repeat the analysis of the benefits of a reduced composite model as was done in the previous section. We will assume that the lowest model order is n_1 and, for the same reasons as in the case in Section 3.4.1, the computations related to the simulation will not be taken into account. We note that the number of instructions in Phase 1 is the same as in the case of the composite model (3.17), which is for $n_1 = 1$ equal to $2N+1$, where N is the number of subsystems. Because matrix $\mathbf{A}$ is diagonal, the number of parameters in the second phase is significantly lower than in the case described in the previous section. Therefore, to improve the model dynamics, subsystem delays no longer have to be of fixed values and will be considered here as free parameters. Thus, the number of parameters for the second phase is $3N + 1$. In this case N parameters come from $\mathbf{A}_{cm}$, another N from $\mathbf{c}_{cm}$ and there are also N dead-times. The required number of processor instructions, under the same assumptions as in the previous section is, for the case when $n_1 = 1$

$$\text{ins} = (2N + 1)^2 + (3N + 1)^2. \tag{3.48}$$

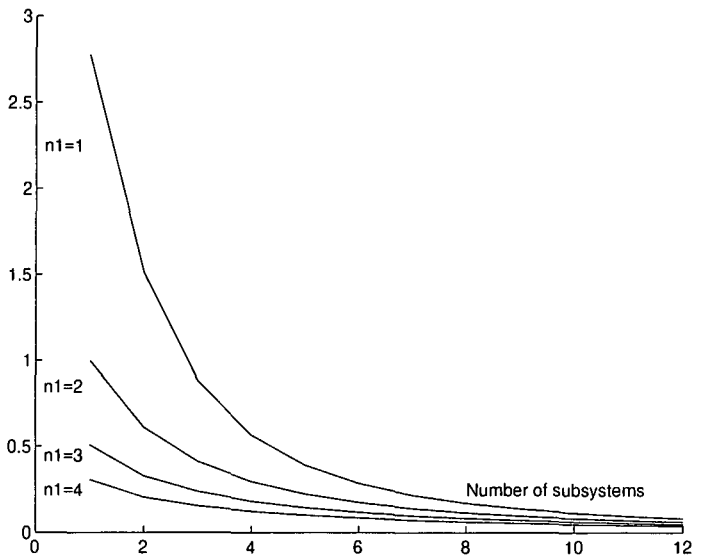


Figure 3-22: R vs. number of subsystems - reduced model

If $n_1 = 2$, then this number will be

$$\text{ins} = (2N + 3)^2 + (3N + 1)^2. \tag{3.49}$$

For the general case, when $n_1 = k$ this is

$$\text{ins} = (2N + 2k - 1)^2 + (3N + 1)^2. \tag{3.50}$$

At the same time, identification by one-shot transfer function model will require the number of instructions as already calculated in the previous section. The ratio R between the required number of instructions for the reduced parameter composite model and the transfer function model can be formed and it is presented in Fig. 3-22. It can be seen that in this case, the ratio is much more favourable for the composite model. We have ended up in the usual engineering trade-off: numerical benefits vs. model accuracy.

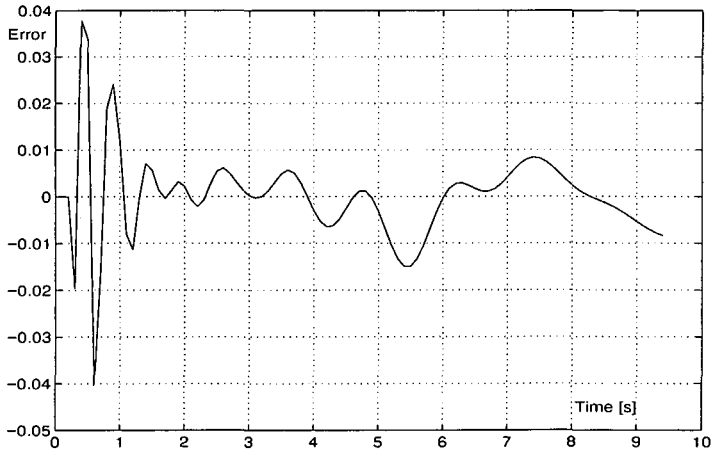


Figure 3-23: Output error for the reduced composite model

Example 3.6 The 3-channel autopilot model from the previous examples is used. The subsystems consist of transfer functions of first, second and third order that were already obtained in Example 2.3, and local output feedback is employed. The obtained IAE is 0.5583 after 33050000 floating point operations. The computational effort is, as expected, smaller than in the case of the identification in Example 3.3. It is interesting to note that although one of the subsystems (of order 2) has a pair of complex-conjugate poles, this composite model is still able to fit the given I/O sequence well. The frequency response of this model is presented in Fig. 3-24. Output model error is shown in Fig. 3-23. It can be concluded that the high-frequency characteristics are not as good as in the case of the model with full output feedback.

3.7 Dynamic System As Interaction Module

In the previous section an interaction module based on the static output feedback was introduced. We recall now that the exact model matching problem can sometimes be solved by introducing dynamic compensation, ether in open or closed loop configuration. In this section we analyze a type of dynamic compensation as an alternative to the output

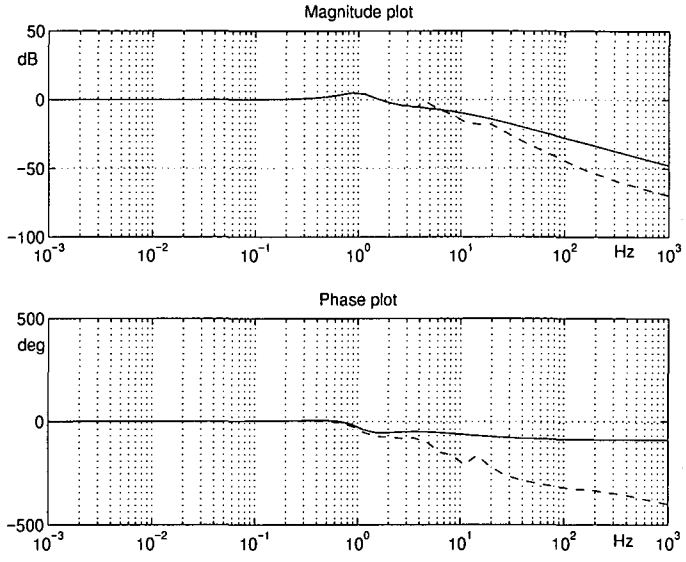


Figure 3-24: Frequency response for the reduced composite model

feedback one, which was presented in Section 3.3. The main difference between static output feedback and dynamic compensation is that the latter results in a composite model of higher order than given by the sum of the subsystem orders. Dynamic compensation and the presence of dynamic components in the feedback loop greatly increases our ability to obtain the desired model. The price to be paid is, of course, increased model complexity and a greater number of parameters. One possible dynamic interaction module will now be presented.

The composite model which will be studied can be described as follows:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ y = \mathbf{c}\mathbf{x}(\mathbf{t}) \\ \mathbf{u} = \begin{bmatrix} u_1 & . & . & . & u_N \end{bmatrix} \\ u_i = G_i(s)e^{-L_i s}u \end{cases} \tag{3.51}$$

where u is the input signal to the SISO system and y is its output. The matrix $\mathbf{A}$ is $n \times n$ matrix ($n \geq N$), the matrix $\mathbf{B}$ is $n \times N$ and the vector $\mathbf{c}$ is $1 \times n$. As can immediately be

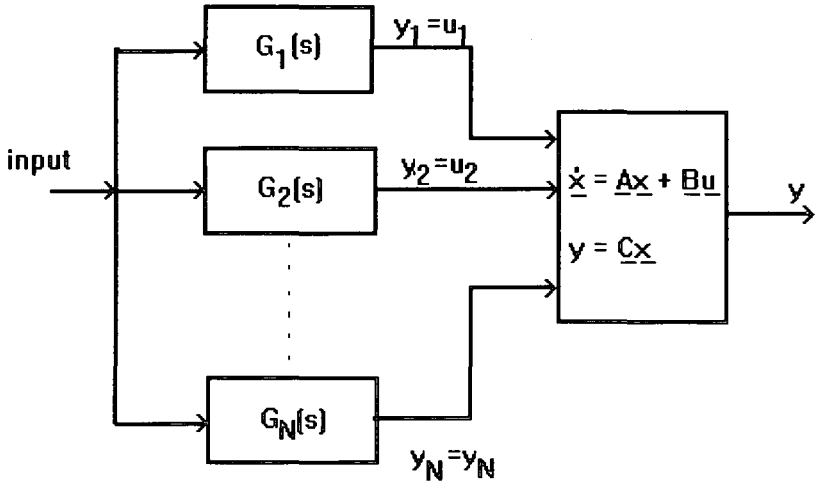


Figure 3-25: Composite model with dynamic interaction

seen, there is no feedback in this composite model, and the interaction module is purely dynamic. Subsystem outputs are fed directly into the interaction module. This scenario is depicted in Figure 3-25. In order to see how this compensation will influence the model eigenvalues, it is necessary to write the state-space equations of the composite model from (3.51). To do this, we will consider transfer functions G_1, G_2, ..., G_N as decoupled multivariable system $S1$, depicted in Fig. 3-10. State-space equations for $S1$ are

$$\begin{cases} \dot{\mathbf{x}}_{s1} = \mathbf{A}_{s1}\mathbf{x} + \mathbf{b}_{s1}u \\ \mathbf{y}_{s1} = \mathbf{C}_{s1}\mathbf{x}_{s1}(t). \end{cases} \tag{3.52}$$

Matrices $\mathbf{A}_{s1}$ and $\mathbf{C}_{s1}$ are formed as in (3.42). Serially to $S1$ is connected system $S2$

$$\begin{cases} \dot{\mathbf{x}}_{s2} = \mathbf{A}_{s1}\mathbf{x}_{s2} + \mathbf{B}_{s2}\mathbf{u} \\ y = \mathbf{c}_{s2}\mathbf{x}_{s2}(\mathbf{t}) \end{cases} \tag{3.53}$$

whose inputs $\mathbf{u}_{s2}$ are the outputs $\mathbf{y}_{s1}$ of $S1$. To form the state-space equation of (3.51),

we introduce the new state vector as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{s1} \\ \mathbf{x}_{s2} \end{bmatrix}. \tag{3.54}$$

Because $\mathbf{u}_{s2} = \mathbf{y}_{s1} = \mathbf{C}_{s1}\mathbf{x}_{s1}$, (3.52) becomes

$$\begin{cases} \dot{\mathbf{x}}_{s2} = \mathbf{A}_{s2}\mathbf{x}_{s2} + \mathbf{B}_{s2}\mathbf{C}_{s1}\mathbf{x}_{s1} \\ y = \mathbf{c}_{s2}\mathbf{x}_{s2}. \end{cases} \tag{3.55}$$

Therefore, the state-space equations for the composite model become

$$\begin{cases} \dot{\mathbf{x}} = \begin{bmatrix} \mathbf{A}_{s1} & \mathbf{0} \\ \mathbf{B}_{s2}\mathbf{C}_{s1} & \mathbf{A}_{s2} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_{s1} \\ \mathbf{0} \end{bmatrix} u \\ y = \begin{bmatrix} \mathbf{0} & \mathbf{c}_{s2} \end{bmatrix} \mathbf{x}. \end{cases} \tag{3.56}$$

It is now obvious that the composite model order is equal to the sum of all subsystem orders (i.e. the order of $S1$) and the order of the dynamic compensator. Because the system matrix of (3.56) is a lower-triangular block matrix, its eigenvalues are simply the union of eigenvalues of $S1$ and those of $S2$. Therefore, this type of compensation cannot modify eigenvalues of the models obtained in Phase 1 and only the eigenvalues obtained in Phase 2 can be modified. We would therefore expect this type of interaction module not to be very beneficial.

The next step is to compare the numerical benefits of hierarchical identification by this approach to one-shot identification. As in the previous analysis, we will assume that the model order for $G_1(s)$ is n_1, and then we will study the benefits for $n_1 = 1$, 2, 3, etc. For the general case, when $n_1 = k$, the resulting model order will be $N(N+2k-1)/2+n_{s2}$, where N is the number of subsystems and n_{s2} is the order of subsystem S_2. With the same assumptions as in the feedback case, namely that the number of instructions will

depend on the square of the number of parameters, we can write

$$\text{ins} = (2N+1)^2 + (n_{s2}^2 + Nn_{s2} + n_{s2} + 1)^2. \tag{3.57}$$

where the first term of the sum corresponds to the number of instructions needed in the first phase (i.e. the number of instruction required by the highest order subsystem), while the second term corresponds to the number of instructions for the second phase in which parameters of dynamic module S_2 are estimated. This term includes the dead-time element, while subsystem dead-times can be fixed to their values obtained in the subsystem identification procedure. This can be justified by the fact that there is no feedback in this composite model, so we should leave the dead-times as they are, because the dead-time elements help to minimize the $\sqrt{ISE}$ of subsystems. Another approach is to fix subsystem dead-times to zero, for the reasons already discussed in Section 3.4. Analysis of benefits for the dynamic compensation case is slightly more complex than for the output feedback based schemes, as the composite model order depends not only on the form of the subsystems, but also on the order of S_2. In order to simplify the analysis, we take it that S_2 has order equal to the number of subsystems in S_1. In this case, (3.57) becomes

$$\text{ins} = (2N + 2k - 1)^2 + (2N^2 + N + 1)^2.$$

if $n_1 = k$. We can now plot the ratio R between the required number of instructions for hierarchical identification by this approach and the one-shot approach. This is plotted in Figure 3-26.

As can be seen from Fig. 3-26, the numerical efficiency of the composite model decreases as a function of the number of subsystems. Not only does numerical efficiency decrease, but also the relative ability of the dynamic module to influence the composite model's eigenvalues. To show this, observe that the system matrix of this composite

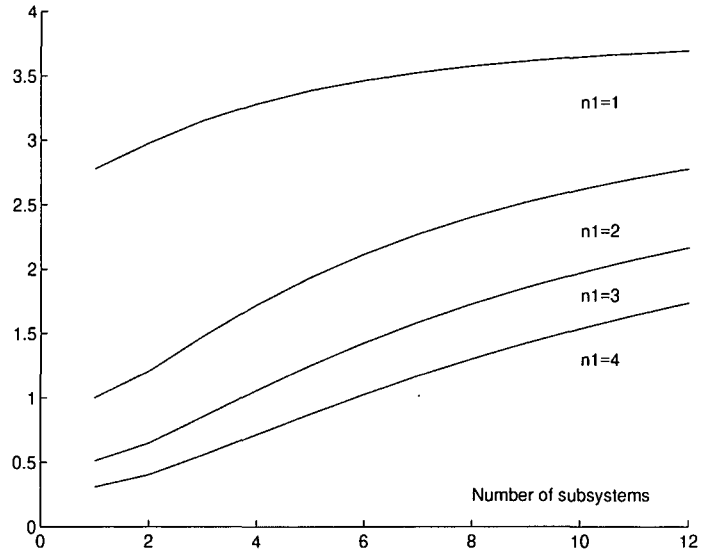


Figure 3-26: R vs. number of subsystems - dynamic compensation

model is, as found in (3.56)

$$\mathbf{A}_{cm} = \begin{bmatrix} \mathbf{A}_{s1} & \mathbf{0} \\ \mathbf{B}_{s2}\mathbf{C}_{s1} & \mathbf{A}_{s2} \end{bmatrix}$$

where A_{s1} and A_{s2} are system matrices of the system of parallel transfer functions (as in Fig. 3-10) and of the dynamic interaction module, respectively. From this it is obvious that this type of interaction cannot modify any of the eigenvalues of S_1. The only eigenvalues that it can change are those that belong to the interaction module itself. The number of composite model eigenvalues that can be placed to some preassigned values is in this case equal (in the light of Definition 1) to the order of the interaction module, which is, on the other hand, equal to or greater than the number of subsystems identified in Phase 1. The main difference between the feedback interaction module and the dynamic one is that static feedback modifies all eigenvalues (despite the fact that it can assign only N of them to some given values) and dynamic feedback modifies only those that belong to the interaction module. On the other hand, it is much easier to enforce the

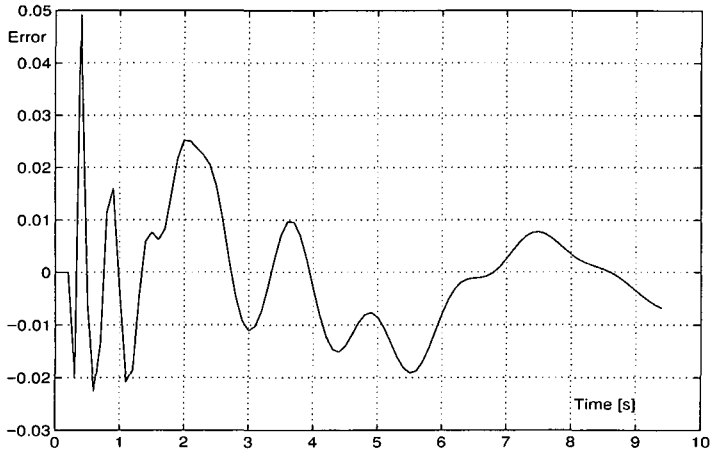


Figure 3-27: Output error for dynamic composite model

stability of such dynamic composite model, as all interaction module eigenvalues can be controlled and placed to arbitrary positions in the left half of the complex plane. As a conclusion, we can say that this approach will require higher number of parameters than the output feedback one. Also, the number of the eigenvalues that are manipulated by is smaller than in the output feedback case.

Example 3.7 Again, our favourite autopilot model is used to illustrate the method. In this case, the composite model consists of two subsystems, of orders 2 and 3, and the dynamic module is of second order. The original subsystem dead-times were fixed to zero. The reason for this is purely practical, due to the inability of the simulation software to compensate for dead-times in a negative direction on the time axis. Ideally, these dead-times should be free parameters, but in order to minimize their number this is the only viable option. The result is that IAE of this model is 0.854 and the frequency characteristics of this model are plotted in Figure 3-28. Model output error is shown in Fig. 3-27. The error of this model is bigger than the error of the model from previous examples. However, it has surprisingly good frequency characteristics relative to the output error.

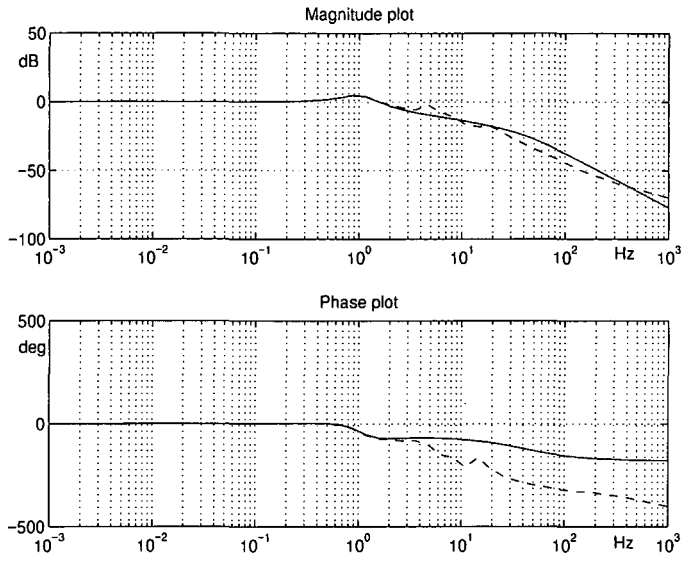


Figure 3-28: Frequency characteristics for dynamic composite model

3.8 Conclusion

Several approaches were discussed with regard to hierarchical identification of SISO LSSs. A composite model consists of a set of low-order models, identified in Phase 1 of hierarchical identification. These subsystems are coupled together and then the dynamic of the resulting composite model is modified. Two main types of composite models were investigated, static output feedback and feedforward dynamic. In the case of static output feedback schemes, various approaches to subsystem dead-times were investigated.

There are two main criteria on which various hierarchical identification approaches were evaluated. They are the maximum number of parameters required for identification in both phases and the ability of the identification procedure to match the given input-output sequences. It is not possible to assess the second criterion accurately; however, certain conclusions can be drawn with regard to the possibilities of various interaction schemes to modify the composite models dynamics. It was shown that even under mild assumptions, the hierarchical identification will bring numerical benefits over one-shot identification of models of equal order.

The first type of composite models consist of low-order models given by their transfer functions, determined in Phase 1. In Phase 2, these two subsystems are connected in parallel, and the subsystem outputs are fed (via feedback matrix) to the subsystem inputs. The order of the composite model is equal to the sum of orders of subsystems. In this way, under the assumption of parallel identification of subsystems, the numerical effort in Phase 1 is determined by the identification effort of the highest-order subsystem. The numerical effort of Phase 2 is determined only by the number of subsystems of the composite model, but not by the structure of subsystems themselves. The variations of this approach are derived from different possibilities regarding the subsystem dead-times. Subsystem dead-times can be regarded as free parameters to be tuned in Phase 2, but as this approach will increase the number of parameters that simultaneously enter optimization routines, it is preferred to have these dead-times fixed. There are two possibilities in this regard: in the first one, subsystem dead-times are taken to be zero, and the composite model's dead-time is solely modelled by the composite dead-time element. In the second approach, some non-zero dead-time can be inserted in the feedback loop. The result is that the effective composite model's order is theoretically infinite, as dead-time can be represented by the infinite-order linear model. In this case, a portion of system dead-time is placed in feedback loop and a portion in composite dead-time. Identification with various subsystem dead-times was performed in order to assess the approaches. It can be concluded that when the apparent dead-time from Phase 1 is small (relative to either dominant time constants or period of observation), the subsystem dead-times should be fixed to zero. In the case when dead-time obtained in Phase 1 is significant, fixed dead-time (equal for all subsystems) may be inserted in the feedback loop. There is no clear answer to what this dead-time should be, but simulations suggest that it should be small, if necessary at all.

Another static output feedback approach investigated, which reduces the number of parameters even further, employs local feedback for each of the subsystems. In this way $N \times N$ (N =number of subsystems) feedback matrix is reduced to an N-component

vector. This reduces the number of parameters in Phase 2 significantly. The price paid is that the model matching capabilities of such approach is smaller than in the case of full (global) output feedback. However, the capability of modifying the model dynamics (flexibility) decreases more slowly than the number of parameters, and this approach may be used in the case of large number of subsystems. In addition, due to the much smaller number of parameters in this case, subsystem dead-times can be regarded as free parameters. This will increase the ability of this approach to match input-output sequences. Identification results obtained by this method are acceptable.

One type of dynamic feedforward interaction was investigated. The outputs of subsystems are fed to a dynamic compensator. This type of compensation cannot modify the subsystem dynamics. Therefore, the ability of such approach to modify the composite model's behaviour is limited to the dynamic module only. The number of parameters in Phase 2 is relatively high, implying the use of smaller number of higher-order subsystems. The results obtained by hierarchical identification using dynamic compensation have shown higher output error, but the frequency characteristics compares well to the frequency characteristic of the autopilot system.

Chapter 4

Hierarchical Identification of MIMO Models

In this chapter, we propose methodology for hierarchical identification of large-scale MIMO systems. It will be shown that such hierarchical identification differs slightly from the SISO case. As in the case of SISO models, in the identification of MIMO models we can employ parallelism in the identification procedure by choosing a suitable composite model whose subsystems can be identified independently and in parallel. Therefore, the whole discussion about model matching procedure applies equally to the multivariable case. However, the decomposition strategies for multivariable models are different from those we are using for SISO models. In what follows, we first look at the possibilities for multivariable model decomposition and creation of interactions among the subsystems obtained by such decomposition.

4.1 Multivariable Model Decomposition

The main problem when decomposing SISO models is that there is only one measurable input and one measurable output. Very often, any decomposition of such models, even when there is enough knowledge of the system that is to be modelled, has to be done

purely in terms of an artificial, mathematical decomposition. A suitable composite model structure is chosen so that some of its subsystems can be identified in parallel and then parameters of interactions between these subsystems are determined.

In the multivariable case, this decomposition is, to a certain degree, more natural, because there is a greater freedom of choice of decomposition. Firstly, the study of correlation functions between input and output signals may reveal to a certain degree a decoupled system, one in which one input controls only one output and whose transfer function matrix is diagonal. It is then clear that each of these transfer functions can be evaluated independently of others, resulting in a parallel identification procedure (Godfrey 1993). Alternatively, a group of inputs can be found to be strongly correlated to some group of outputs, enabling separate identification of subsystems from such groups of I/O sets. The problem is, of course, what to do when such correlation does not exist and there is no other information on the system structure. In this case such parallelization cannot be applied. In this black-box case, the composite model must be built on mathematical decomposition.

Before we propose a mathematical decomposition procedure for generating the appropriate multivariable composite model structures, a hierarchical, two phase identification procedure proposed by Hasiewicz and Stankiewicz (1986) will be briefly described. This is the earliest work that the author has found dealing with two-phase identification of multivariable systems. Their results are developed only for the steady-state models of the LSS and as such they are not applicable to identification of dynamic models. Furthermore, they assume complete knowledge of the interaction module which is completely opposite to our approach. The system considered by Hasiewicz and Stankiewicz is given as

$$\mathbf{y} = \mathbf{F}(\mathbf{c}, \mathbf{u}), \quad \mathbf{u} = \mathbf{H}\mathbf{y} \tag{4.1}$$

in which $\mathbf{y}$ is the output vector, $\mathbf{u}$ is the input vector and $\mathbf{c}$ is a control vector. The operator $\mathbf{F}$ is unknown, but the interconnection matrix $\mathbf{H}$ is assumed to be exactly known. The algorithm is hierarchical and enables parallel identification of submodels,

but requires a knowledge of system structure, which is retained in the global model.

In our new hierarchical approach, we remove the assumption of the knowledge of system structure, and confine ourselves to the black-box identification. Once again, the only assumption taken here is that the input-output data is sufficiently good to identify important system modes.

It is well-known that a given set of state-space equations can be brought into more desirable form by application of some suitable transformations and control laws. One example would be the widely studied subject of decoupling by state feedback. The result of such decoupling is a model in the state-space form whose equivalent transfer function matrix is diagonal, in which case one input signal controls only one output. Systems controlled in a such a way have a definite advantage over uncoupled systems in certain applications (for instance aircraft dynamics - see Owens 1978 and Fortmann and Hiltz 1977).

A particular mathematical decomposition procedure that will be used here for development of hierarchical identification procedure is similar to the one proposed by Šiljak (1978) in which decomposition was developed in order to obtain input and output decentralized models. The main reason for development of such decentralized model structure is to ease the study of system stability in the presence of changes in structure of subsystem interactions.

Let us consider a multivariable model given as

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} = \mathbf{C}\mathbf{x} \end{cases} \tag{4.2}$$

where $\mathbf{x} \in R^n$ is a state vector, and $\mathbf{y} \in R^q$ and $\mathbf{u} \in R^p$ are output and input vectors. Matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are of dimensions $n \times n$, $n \times p$ and $q \times n$. What we would like to do is to find a particular model whose state vector $\mathbf{x}$ can be written as the direct sum of some vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$, where k is the number of partitions of the original state vector. To avoid complications in notation, in order to illustrate the idea, we consider

here decomposition of the system 4.2 into two subsystems.

Suppose that the state vector $\mathbf{x}$ is written as a direct sum of two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ such that $\dim x_1 = n_1$ and $\dim x_2 = n_2$, $n_1 + n_2 = n$. System 4.2 then becomes

$$\begin{cases} \begin{pmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} + \begin{pmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} \\ \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix}, \end{cases} \tag{4.3}$$

where matrices $\mathbf{A}_{11}$, $\mathbf{A}_{12}$, $\mathbf{A}_{21}$, $\mathbf{A}_{22}$ are of dimensions $n_1 \times n_1$, $n_1 \times n_2$, $n_2 \times n_1$ and $n_2 \times n_2$, respectively. The input vector is decomposed into two vectors, $\mathbf{u}_1$ and $\mathbf{u}_2$ of dimensions p_1 and p_2, such that $p_1 + p_2 = p$. Matrices $\mathbf{B}_{11}$, $\mathbf{B}_{12}$, $\mathbf{B}_{21}$ and $\mathbf{B}_{22}$ are of dimensions $n_1 \times p_1$, $n_1 \times p_2$, $n_2 \times p_1$ and $n_2 \times p_2$ respectively. On the output side, vector $\mathbf{y}$ is decomposed as $[\ \mathbf{y}_1 \quad \mathbf{y}_2\ ]^T$, where $\dim \mathbf{y}_1 = q_1$, $\dim \mathbf{y}_2 = q_2$ such that $q_1 + q_2 = q$. Matrices $\mathbf{C}_{11}$, $\mathbf{C}_{12}$, $\mathbf{C}_{21}$ and $\mathbf{C}_{22}$ are matrices whose dimensions are $q_1 \times n_1$, $q_1 \times n_2$, $q_2 \times n_1$ and $q_2 \times n_2$, respectively. From this consideration, it can be seen that the input and output vectors can be partitioned independently of each other. Decomposition of matrix $\mathbf{B}$ will depend on the way we decomposed the state and input vectors, while decomposition of the matrix $\mathbf{C}$ will depend on the method of decomposing the state vector and the output vector.

Vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ can be thought of as state vectors of two new subsystems, Ω_1 and Ω_2, which we will form. Matrices $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$ from 4.5 represent the effects of interactions between the two subsystems. Following Šiljak (1978), matrices $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ can be further written, although it is not immediately apparent, as

$$\begin{cases} \mathbf{A}_{11} = \mathbf{A}_1 + \mathbf{A}_{1s} \\ \mathbf{A}_{22} = \mathbf{A}_2 + \mathbf{A}_{2s} \end{cases} \tag{4.4}$$

where $\mathbf{A}_1$ and $\mathbf{A}_2$ are now new system matrices of the subsystems Ω_1 and Ω_2. Matrices $\mathbf{A}_{1s}$ and $\mathbf{A}_{2s}$ are matrices representing self-interaction for each of the subsystems. This

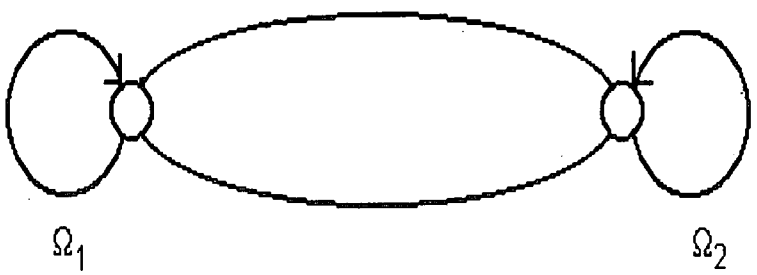


Figure 4-1: Subsystem interactions

can be seen more clearly if subsystems are represented in a graph form, as in Figure 4-1. The reason for this decomposition will be clear shortly.

We can rearrange Equation (4.3) in the following way:

$$\begin{cases} \dot{\mathbf{x}}_1 = \mathbf{A}_1\mathbf{x}_1 + \mathbf{B}_{11}\mathbf{u}_1 + \mathbf{A}_{1s}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 + \mathbf{B}_{12}\mathbf{u}_2 \\ \dot{\mathbf{x}}_2 = \mathbf{A}_2\mathbf{x}_2 + \mathbf{B}_{22}\mathbf{u}_2 + \mathbf{A}_{2s}\mathbf{x}_2 + \mathbf{A}_{21}\mathbf{x}_1 + \mathbf{B}_{21}\mathbf{u}_1 \\ \mathbf{y}_1 = \mathbf{C}_{11}\mathbf{x}_1 + \mathbf{C}_{12}\mathbf{x}_2 \\ \mathbf{y}_2 = \mathbf{C}_{21}\mathbf{x}_1 + \mathbf{C}_{22}\mathbf{x}_2 \end{cases} \tag{4.5}$$

The following dynamical equations can be considered separately:

$$\Omega_1 : \dot{\mathbf{x}}_1 = \mathbf{A}_1\mathbf{x}_1 + \mathbf{B}_{11}\mathbf{u}_1, \quad \mathbf{y}_1 = \mathbf{C}_{11}\mathbf{x}_1 \tag{4.6}$$

and

$$\Omega_2 : \dot{\mathbf{x}}_2 = \mathbf{A}_2\mathbf{x}_2 + \mathbf{B}_{22}\mathbf{u}_2, \quad \mathbf{y}_2 = \mathbf{C}_{22}\mathbf{x}_2 \tag{4.7}$$

These two equations will be used for the creation of the models of free subsystems Ω_1 and Ω_2. Input and output data for subsystems Ω_1 and Ω_2 are given by vectors $\mathbf{u}_1$ and $\mathbf{y}_1$ and $\mathbf{u}_2$ and $\mathbf{y}_2$. Because each of the subsystems has its own local data that is not dependent on the other subsystem's solution, they can be estimated in parallel. This estimation will constitute the first phase of the hierarchical identification procedure.

When comparing the set of equations (4.6) and (4.7) to those in (4.5), we see that what are missing from them are various terms representing the interactions among the subsystems. The missing components are the following:

1) interaction of subsystems Ω_2 and Ω_1, through the state vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively (matrices $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$);

2) influence of input signal $\mathbf{u}_1$ on subsystem Ω_2 and of signal $\mathbf{u}_2$ on subsystem Ω_1 (matrices $\mathbf{B}_{12}$ and $\mathbf{B}_{21}$);

3) components of global solution function that comprise interactions of subsystem's local solutions (e.g. matrices $\mathbf{C}_{12}$ and $\mathbf{C}_{21}$), and

4) Self interaction terms $\mathbf{A}_{1s}$ and $\mathbf{A}_{2s}$.

Although, in principle, we can put self interaction terms $\mathbf{A}_{1s}$ and $\mathbf{A}_{2s}$ to be part of $\mathbf{A}_1$ and $\mathbf{A}_2$, we will find suitable interpretation for $\mathbf{A}_{1s}$ and $\mathbf{A}_{2s}$ in the context of hierarchical identification and for this reason we will consider them separately. In order to match the composite model that consists of subsystems Ω_1 and Ω_2 to the original model from 4.2 it is necessary to introduce additional model components that will compensate for interactions between these two subsystems. Estimation of these model components will be the aim of Phase 2 of hierarchical identification procedure. This can be done as in the SISO case, by model matching through output feedback.

There are several possibilities regarding the approximation of the global model with some composite model. What must be kept in mind is that the objective of creating a composite model is minimization of the number of parameters that simultaneously have to be tuned by optimization procedures employed for the identification. This means that our choice of composite model structure is mainly guided by these practical issues. For example, state-feedback is capable of compensating for all of the subsystem inaccuracies because all of the model's eigenvalues can be assigned to any predetermined values. That means that such composite model can match any given model of the same order and

number of inputs and outputs even if parameters of all subsystems are chosen arbitrarily, as long as the subsystems are chosen in such a way that the composite model will be of appropriate order. However, because of the large number of parameters that would have to be estimated, in the previously described SISO case a less costly output feedback approach was chosen, thus necessitating determination of subsystem models with highest possible accuracy. This approach was shown to work well. In the multivariable case, the number of parameters is substantially higher, and the trade-off between the model accuracy vs. numerical complexity has to be carefully evaluated.

The main problem with the identification of subsystems Ω_1 and Ω_2 is that the subsystems have only partial information about the input and output signals of the original, global MIMO system. In other words, output of Ω_1 does not depend on vector $\mathbf{u}_2$ and output of Ω_2 does not depend on $\mathbf{u}_1$ when these subsystems are considered free i.e. when they do not form a composite model. This problem does not arise in the SISO case where subsystems have all of this information, because there are only one input and one output which are common to all of the subsystems. As in the single variable case, the composite model dynamics can be improved by introduction of output feedback. In SISO composite models, the number of outputs to be fed to inputs is equal to the number of subsystems. In the multivariable case, this number is higher, as every subsystem may have more than one output. For that reason, instead of taking all of the outputs to all subsystems, we propose that feedback loops be strictly local, resulting in a smaller number of parameters. This trade-off will be evaluated in the next section. We can now propose the following steps (similar to the approach proposed in Bajić and Janković 1997) to compensate for the model inaccuracies:

1) Firstly, information about all the system inputs has to brought to all of the subsystems. In order to achieve this, an input distribution matrix $\mathbf{M}_I$ is introduced. This matrix has the aim of optimally distributing the inputs to the subsystems;

2) On the output side, a matrix $\mathbf{M}_O$ is introduced which combines subsystem outputs into composite model output. This vector has the role of global solution function

described in Section 3.1;

3) Matrices $\mathbf{A}_{1s}$ and $\mathbf{A}_{2s}$ represent the influence of the subsystems on themselves. This interaction can be conveniently represented in the form of local output feedback of subsystems. This local output feedback is the main factor influencing the composite model's dynamics as it modifies the values of the model's eigenvalues. This approach is similar to the notion of connective stability (Šiljak 1978), where global system stability must be achieved through local feedback loops and limited subsystem interactions.

From these considerations, a desired composite model can be obtained in the following way. Firstly, input and output vectors are partitioned as in (4.3). Secondly, subsystems Ω_1 and Ω_2 are identified. Using Ω_1 and Ω_2 as building blocks, the following composite model is suggested:

$$\begin{cases} \mathbf{y} = \mathbf{M}_O \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \\ \mathbf{u}_1 = \mathbf{Z}_1 + \mathbf{A}_{1s}\mathbf{y}_1 \\ \mathbf{u}_2 = \mathbf{Z}_2 + \mathbf{A}_{2s}\mathbf{y}_2 \\ \dot{\mathbf{x}}_1 = \mathbf{A}_1\mathbf{x}_1 + \mathbf{B}_{11}\mathbf{u}_1, \quad \mathbf{y}_1 = \mathbf{C}_{11}\mathbf{x}_1 \\ \dot{\mathbf{x}}_2 = \mathbf{A}_2\mathbf{x}_2 + \mathbf{B}_{22}\mathbf{u}_2, \quad \mathbf{y}_2 = \mathbf{C}_{22}\mathbf{x}_2 \end{cases} \tag{4.8}$$

where with $\mathbf{Z}_1$and $\mathbf{Z}_2$, $\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_1 \\ \mathbf{Z}_2 \end{bmatrix} = \mathbf{M}_I\mathbf{u}$ we denote partition of $\mathbf{Z}$ to correspond to the partition of the input vector $\mathbf{u}$ for Phase 1 of identification. This is schematically shown in Fig. 4-2. This model does not cater for dead-times. In order to estimate dead-times, the only modification of the model given by (4.8) is to replace input vector $\mathbf{u}_i = [\ u_1 \quad u_2 \quad . \quad . \quad . \quad u_{n_i}\ ]$ with $\mathbf{u}_i = [\ u_1(t-l_1) \quad u_2(t-l_2) \quad . \quad . \quad . \quad u_{n_i}(t-l_{n_i})\ ]$

Therefore, we can describe the algorithm for hierarchical identification by the following steps:

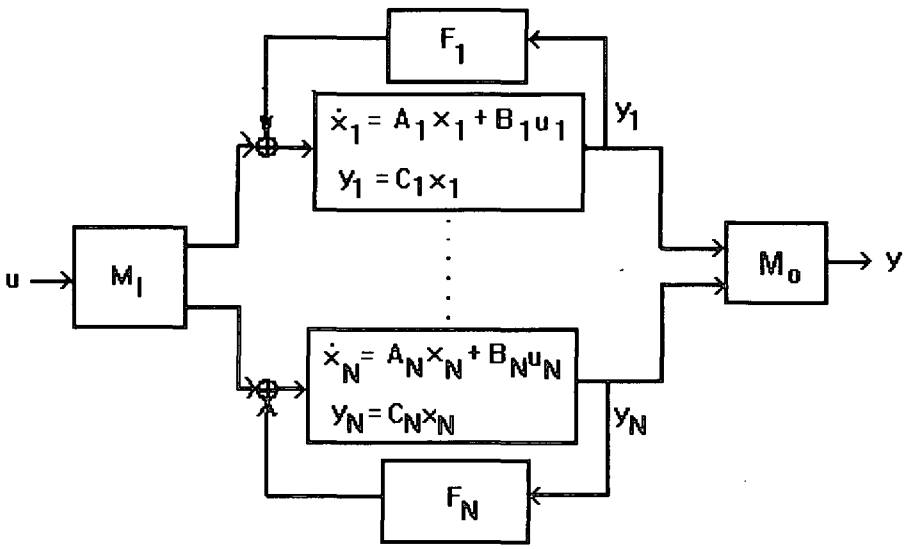


Figure 4-2: Composite multivariable model

Step 1

This is a preparatory step in which no identification is done. It has an aim of establishing the optimal numeration of the systems inputs and outputs. To get the maximum benefit from decomposition, correlation between each input and output should be calculated. Those inputs and outputs with better correlation should be grouped as inputs and outputs to a particular subsystem of order n_i and with p_i inputs and q_i outputs;

Phase 1

The identification of each of the subsystems from input-output sets established in Step 1 can now be performed independently and in parallel by multivariable models in canonical form. The result of this phase is a set of multivariable models.

Phase 2

Matrices $\mathbf{M}_I$, $\mathbf{A}_{si}$ and $\mathbf{M}_O$ should be estimated so that the resulting output model error is minimized.

4.2 Benefits of the Approach

Just as in the SISO case, we will investigate what numerical benefits the hierarchical identification procedure described in the previous section will have and what the shortcomings of the approach are.

Let us consider one-shot identification by multivariable models in canonical form. Suppose that the model has p inputs, q outputs and is of order n. It was mentioned in Section 2.3.3 that there are models which can be realized by a minimal number of free parameters, which was found to be $np + nq$, the rest being fixed ones or zeros. In the case where dead-times are taken into account this number becomes $np + (n+1)q$. Therefore, the required number of instructions for identification of such model will be $[np + (n+1)q]^2$. This estimate is made under our usual assumption that the numerical complexity will be related quadratically to the number of free parameters.

For the first phase in hierarchical identification, we identify N subsystems, each one of order n_i with p_i inputs, and q_i outputs, with $\sum_{i=1}^{N} p_i = p$ and $\sum_{i=1}^{N} q_i = q$, where N is the number of subsystems. The number of parameters for each of the subsystems will be $n_i p_i + (n_i + 1)q_i$. Just as in the SISO case, the numerical complexity in the Phase 1 of identification will be determined by the subsystem having the highest number of parameters which simultaneously enter optimization routines used for model identification. This is most likely to be the subsystem of the highest order, but it does not always have to be so, as number of inputs and outputs play an equally significant role in determination of how many parameters are needed to describe the model. In the second phase, this number will be the sum of elements of matrices $\mathbf{M}_I$, $\mathbf{A}_{si}$, $i = 1, 2, ..., N$, $\mathbf{M}_O$ and dead-times. These are, respectively, p^2, $\sum_{i=1}^{N} p_i q_i$, q^2 and q, where p_i is the number of inputs to the subsystem Ω_i, q_i is the number of outputs of the subsystem Ω_i and q is the number of dead-times.

Since there are too many variables in this analysis, we will take an example. Identification of a large-scale system with 10 inputs and 10 outputs by a model of the order 100 will require estimation of 2010 parameters simultaneously. The number of instructions is

proportional to 2010^2. Suppose now that hierarchical identification procedure has been employed and that the 3 subsystems are used for Phase 1: two of them with 3 inputs and outputs of order 30 and the third one with 4 inputs, 4 outputs and order 40. The first phase will be dominated by identification of the largest subsystem which requires estimation of 324 parameters. In the second phase the required number of parameters is 100 (for matrix $\mathbf{M}_I$) + 9 (for $\mathbf{A}_{1s}$) + 9 (for $\mathbf{A}_{2s}$) + 16 (for $\mathbf{A}_{3s}$) + 100 ($\mathbf{M}_O$) + 10 (for dead-times) = 244. Therefore, for both phases, the number of instructions will be proportional to $324^2 + 244^2$, which is about 25 times less then the number required for one-shot identification.

This decrease in estimated computational effort is significant. Still, the question remains how effective such hierarchical identification can be. Although the procedure in its second phase will modify all of the composite model eigenvalues, the question is how many eigenvalues it can asymptotically assign. The following lemma gives an answer to this question.

Lemma 2 The interaction module proposed in (4.8) can asymptotically assign a maximum of

$$\sum_{i=1}^{N} \min_i(n_i,\ p_i + q_i - 1) = p + q - N$$

eigenvalues, where n_i, p_i, and q_i are the order, number of inputs and number of outputs, respectively, for the subsystem i.

Proof Proof follows directly from the discussion about output feedback, after the proof of Lemma 1.

We should note that if the output feedback is not local to subsystems, but is global, i.e. output of all subsystems are fed to all inputs via $p \times q$ feedback matrix, the number of eigenvalues that can be assigned will be $p + q - 1$. However, because of the significant increase in the number of parameters in this case, we only study the local feedback schemes. As a conclusion, it can be said that the procedure will work well for the cases in which the number of system outputs is not too small compared to the order of the model used to identify the system.

In our analysis we can go one step further and analyze the effect of matrix $\mathbf{M}_O$ in the form described previously. We note that it contributed to the optimization problem with large number of parameters (q^2). In the next section we will attempt to identify a MIMO system by taking matrix $\mathbf{M}_O$ as diagonal matrix to show that even in this case when its effect is mainly a weighting of the output signals it may contribute to the quality of the overall model. This reduces the number of parameters and we would like to see what effect will such simplification have on the accuracy of hierarchical identification.

Finally, it should be noted that all remarks about excluding computations related to simulations from Section 3.4.1 are equally applicable to this case as well.

4.3 Examples

A few examples will be given here to illustrate the procedure. The required number of parameters can be very significant for a one-shot approach so that it is often very impractical to attempt identification with such methods. These examples serve the purpose of illustrating how well hierarchical identification will replace one-shot identification methods. Transfer function matrices are generated randomly, but is such a way that resulting matrices are strictly proper. The choice of subsystems was done in a fairly arbitrary fashion, as we expect the method to work well for large number of possible decompositions. Same is true for the choice of input signals. All of the theory related to the choice of input signals is, of course, still valid and applies equaly to any of identification methods. For that reason, no special effort has been made to use persistenly exciting signals for all inputs, as we are mainly interested to explore the ability of hierarchical identification schemes to minimize the output error of composite models as well as to reduce the output error in Phase 2 of such procedure. However, the quality of so obtained models will depend on input signals.

Example 4.1 The model used to generate I/O sequences in this first example will have 4 inputs and 4 outputs and is given with the following (randomly generated) transfer

Output #	Phase 1 error ($\sqrt{ISE}$)	Phase 2 error ($\sqrt{ISE}$)
1	0.4068	0.1688
2	0.0060	0.0198
total for subsystem 1	0.4128	0.1886
3	0.5115	0.3431
4	0.5474	0.3264
total for subsystem 2	1.0589	0.6695
total	1.4717	0.8581

Table 4.1:

function matrix:

$$\mathbf{G}(s) = \begin{bmatrix} \frac{1}{s+1} & \frac{0.25}{3s+1} & 0 & \frac{0.1}{s+1} \\ \frac{0.1}{2s+1} & \frac{1}{4s^2+s+1} & 0 & 0 \\ 0 & \frac{0.1}{s+1} & \frac{1}{s+1} & \frac{1}{2s^2+s+1} \\ \frac{0.1}{4s+1} & \frac{0.1}{2s^2+s+1} & \frac{0.4}{4s+1} & \frac{1}{s+1} \end{bmatrix} \tag{4.9}$$

We now attempt to identify the MIMO model that would match these I/O sequences by hierarchical identification methodology. The subsystems are formed by grouping the first two inputs and the first two outputs into subsystem 1 and the last two inputs and the last two outputs into subsystem 2. These subsystems are then modelled separately in Phase 1. In Phase 2, parameters of interaction module are determined. Subsystem matrices as well as interaction matrices are given in Appendix B. The identification results obtained in each of the phases of identification are summarized in Table 4.1.

It can be seen from the table that in the Phase 1, the output error from output #2 is very small. If we look at the transfer function matrix (4.9), we can see that the output 2 does not depend on inputs 3 and 4. Therefore, in Phase 1 this output has all the information about the relevant inputs contained within the input-output sequences for subsystem 1. We are therefore not surprised that the output error associated with output 2 is very small (see also the graphical representation). However, all other outputs depend on some of the inputs that belong to the other subsystem and we expect that

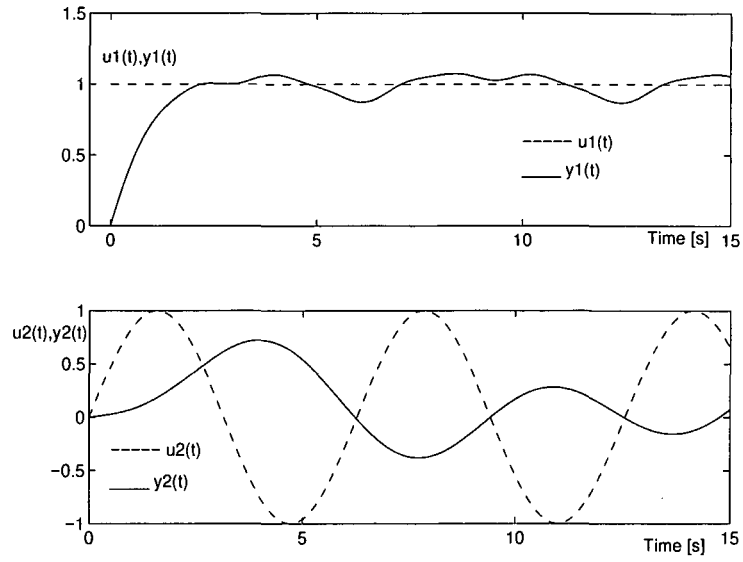


Figure 4-3: Inputs and outputs #1, 2

we may not be able to model them accurately. In Phase 2, parameters of interactions are determined and the associated output model error dropped significantly (it is about half of the total error of the subsystems obtained in Phase 1). It is interesting to note that this time the output error for output #2 has somewhat worsened. This is a result of the input distribution matrix which has primarily modified the input distribution to minimize the output error for other outputs.

Input and output signals are shown in Figure 4-3 and 4-4 while the results of identification in Phase 1 and Phase 2 are presented in Figures (4-5), (4-6), (4-7) and (4-8), respectively.

Example 4.2 In this example, we illustrate the application of hierarchical identification for a system that would be very difficult to identify by a one-shot identification approach. Secondly, we would like to see how well the identification approach will perform if no input-output correlation analysis is done before the actual identification of subsystems. This will illustrate the robustness of the procedure in the case when inputs and outputs

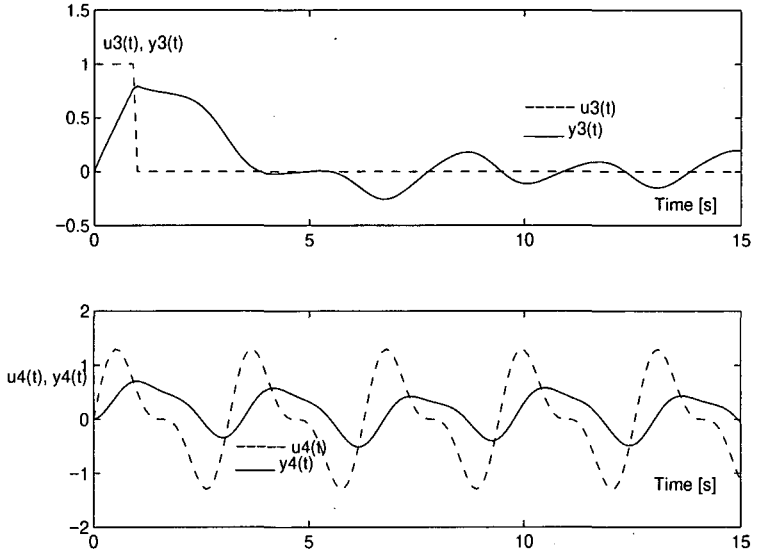


Figure 4-4: Inputs and outputs #3, 4

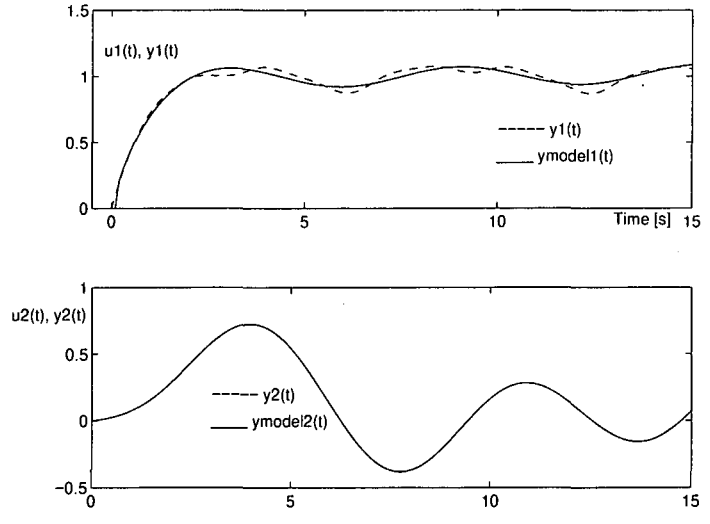


Figure 4-5: Results of Phase 1, outputs #1, 2

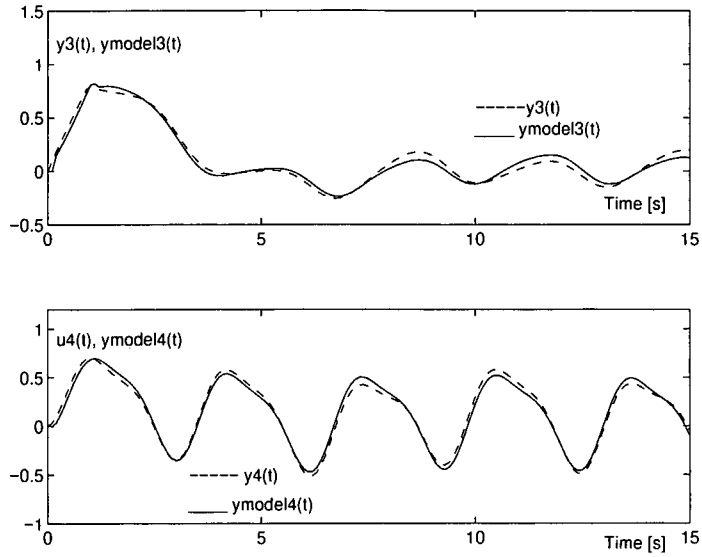


Figure 4-6: Results of Phase 1, outputs #3, 4

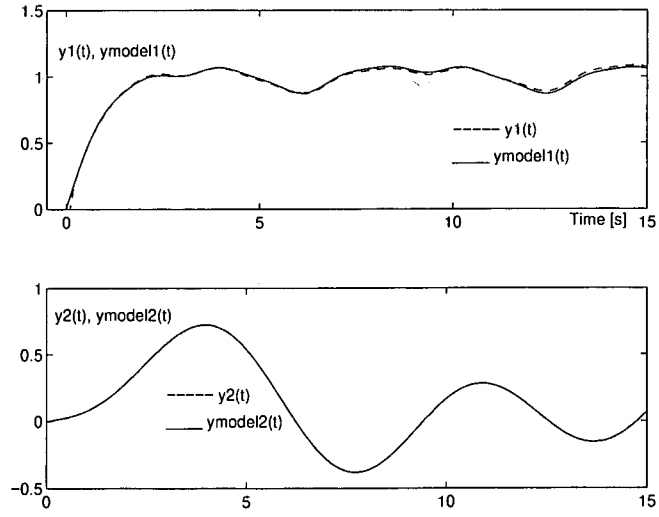


Figure 4-7: Results of Phase 2, outputs #1, 2

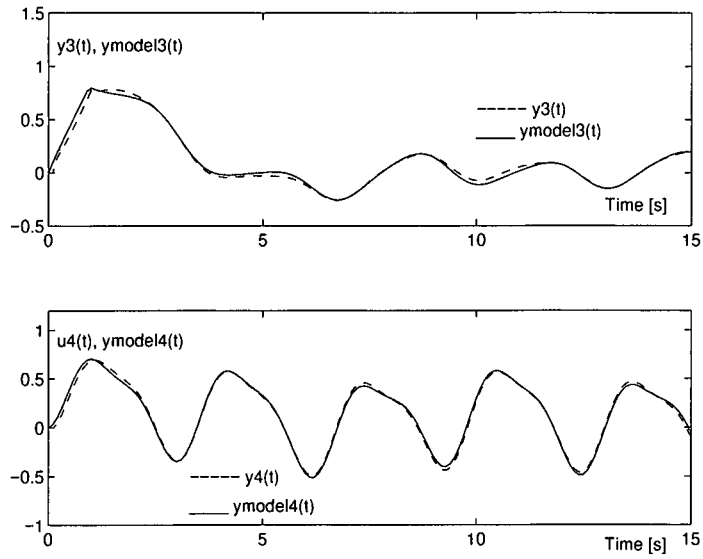


Figure 4-8: Results of Phase 2, outputs #3, 4

are strongly coupled. The transfer function matrix $\mathbf{G}(s)$ given by (4.10), used to generate I/O sequences for the MIMO system in this case, is 10×10 and there are consequently 10 input and 10 output sequences.

$$\mathbf{G}(s) = \begin{bmatrix}
\frac{1}{s+1} & \frac{0.25}{3s+1} & 0 & \frac{0.1}{2s^2+s+1} & \frac{1}{s+1} \\
\frac{0.3}{s+1} & \frac{1}{3s+1} & 0 & \frac{0.1}{s^2+s+1} & \frac{1}{s+2} \\
\frac{0.25}{s^2+s+1} & \frac{0.25}{s+1} & \frac{1}{2s^2+s+1} & 0 & \frac{1}{s+1} \\
\frac{1}{s+1} & \frac{0.25}{3s+1} & 0 & 0 & \frac{1}{s+1} \\
\frac{0.2}{s+1} & \frac{0.25}{3s+1} & \frac{0.3}{3s+1} & \frac{0.1}{s+1} & \frac{1}{2s^2+s+1} \\
0 & 0 & \frac{1}{s+1} & \frac{0.1}{s+1} & 0 \\
0 & 0 & 0 & \frac{1}{s+1} & \frac{0.25}{s+1} \\
0 & \frac{0.25}{3s+1} & \frac{0.25}{s+1} & \frac{0.1}{s+1} & \frac{1}{s+1} \\
\frac{0.25}{s+1} & \frac{0.25}{3s+1} & \frac{0.1}{s+2} & \frac{0.1}{s+1} & \frac{0.6}{s^2+s+1} \\
0 & \frac{0.25}{3s+1} & \frac{0.1}{s+1} & \frac{0.1}{s^2+s+1} & \frac{0.4}{s^2+s+1}
\end{bmatrix}$$

$$\begin{bmatrix}
\frac{0.25}{3s+1} & 0 & \frac{0.1}{s+1} & 0 & \frac{0.1}{2s+1} \\
0 & \frac{0.2}{s+1} & \frac{0.1}{s^2+s+1} & \frac{0.1}{s+1} & \frac{0.1}{s^2+s+1} \\
\frac{0.25}{3s+1} & 0 & \frac{0.1}{s+1} & 0 & \frac{0.1}{s+1} \\
\frac{0.25}{3s+1} & \frac{0.1}{s+1} & 0 & \frac{0.5}{2s^2+s+1} & \frac{0.1}{s+1} \\
0 & \frac{0.1}{s+1} & \frac{0.1}{3s+1} & \frac{0.2}{s+1} & 0 \\
\frac{2}{3s+1} & 0 & \frac{0.1}{s+1} & \frac{0.15}{s+1} & 0 \\
0 & \frac{2}{s+2} & \frac{0.1}{s+1} & 0 & \frac{0.1}{s+1} \\
\frac{0.25}{3s+1} & 0 & \frac{1}{4s^2+s+1} & \frac{0.2}{s+1} & 0 \\
\frac{0.25}{3s+1} & 0 & \frac{0.1}{s+1} & \frac{1}{2s^2+s+1} & \frac{0.1}{s+1} \\
\frac{0.25}{3s+1} & \frac{0.1}{s+1} & \frac{0.1}{s+1} & \frac{0.1}{s+1} & \frac{1}{4s^2+s+1}
\end{bmatrix} \tag{4.10}$$

In order to apply hierarchical identification procedure, three subsystems are formed. The I/O sequences for the first subsystem are formed by choosing the first three inputs and first three outputs from the original I/O list. For the second subsystem, we choose inputs 4, 5 and 6, as well as the corresponding outputs 4, 5 and 6. Finally, the last four inputs and the last four outputs form I/O sets for the subsystem 3.

It can be seen by observing the transfer function matrix (4.10) that the outputs in each of the subsystems depend on the inputs of other subsystems. This means that identification in Phase 1 may not be very accurate and indeed some of the outputs have large output error. In Phase 2, parameters of the interaction module are determined. The subsystems and interaction matrices are given in Appendix B. The identification results are summarized in Table 4.2.

By studying the results from Table 4.2, we see that the output error has been significantly reduced for subsystems 1 and 3 and only marginally reduced for subsystem 2. The possible reason is that the minimum achieved is not a global one. This is indicated by the structure of input distribution and feedback matrices (see Appendix B). The elements of these matrices responsible for changes in dynamics of subsystem 2 are left virtually

Output #	Phase 1 error ($\sqrt{ISE}$)	Phase 2 ($\sqrt{ISE}$)
1	0.3200	0.1932
2	0.7208	0.1640
3	0.1457	0.1097
total for subsystem 1	1.1865	0.4669
4	1.4078	1.4078
5	0.7944	0.7938
6	0.5557	0.5560
total for subsystem 2	2.7579	2.7576
7	0.9021	0.2638
8	0.6560	0.4303
9	0.7669	0.6380
10	0.4812	0.4033
total for subsystem 3	2.8062	1.7354
total	6.7506	4.9599

Table 4.2:

unchanged from their initial values by the optimization procedure. It is very likely that some other set of initial values would lead to improvement in output error for outputs that belong to subsystem 2. All dead-times, with the exception of one, are close to their true values.

4.4 Conclusion

In this chapter we have investigated hierarchical identification methods for identification of MIMO systems. Decomposition of MIMO models was done by partitioning the state vector. In this way independent subsystems were formed which were then identified in parallel. In the second phase, we employed the output feedback locally in each subsystem. In this way the number of parameters entering optimization procedures is smaller. The price to be paid is a somewhat smaller number of eigenvalues that can be placed in given positions in the complex plane. The number of parameters required by any of the identification phases is significantly smaller than the number required in the one-shot identification approach, particularly for high-order models with a substantial number of

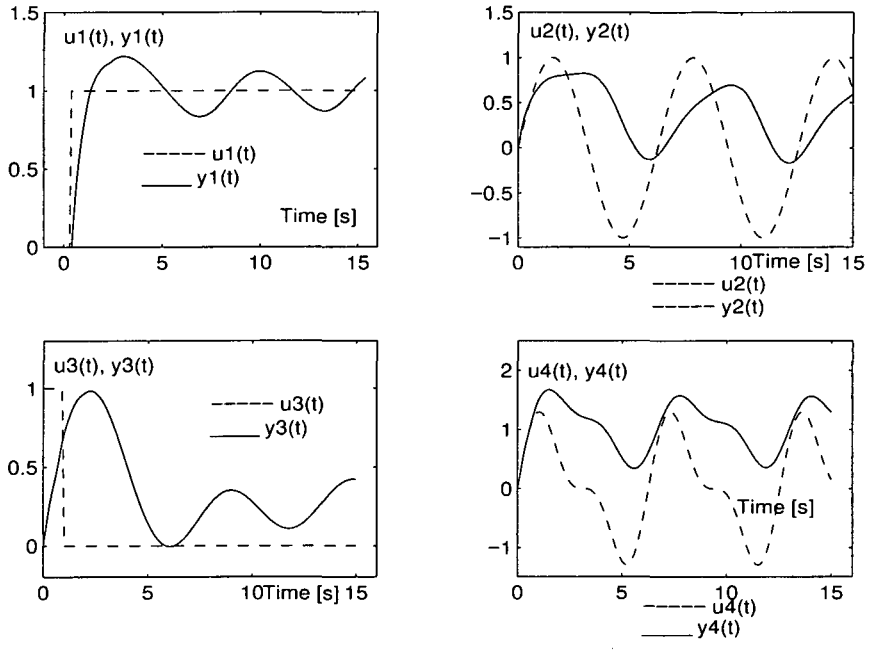


Figure 4-9: Inputs and outputs, #1, 2, 3, 4

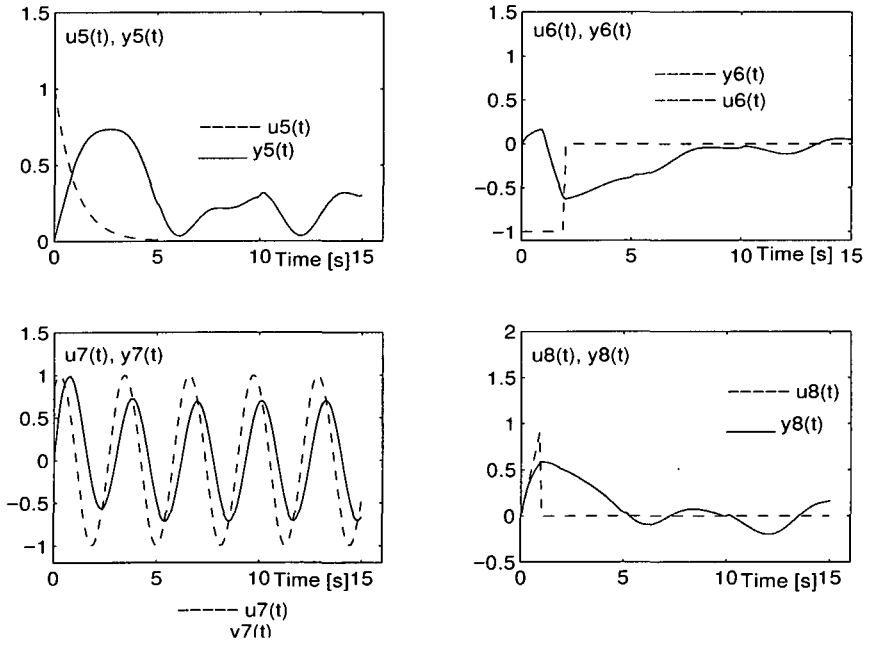


Figure 4-10: Inputs and outputs, #5, 6, 7, 8

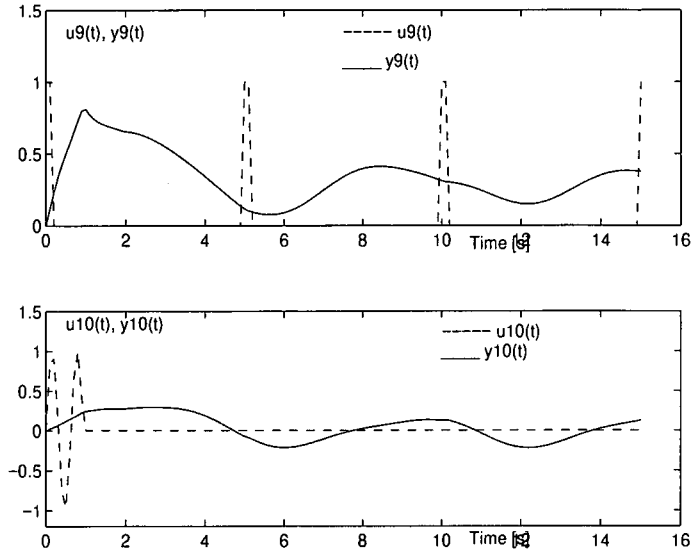


Figure 4-11: Inputs and outputs #9, 10

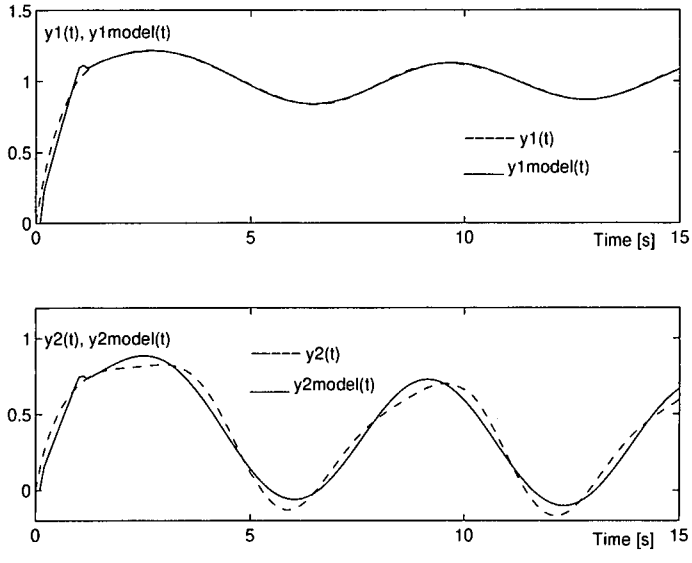


Figure 4-12: Results of Phase 1, outputs #1, 2

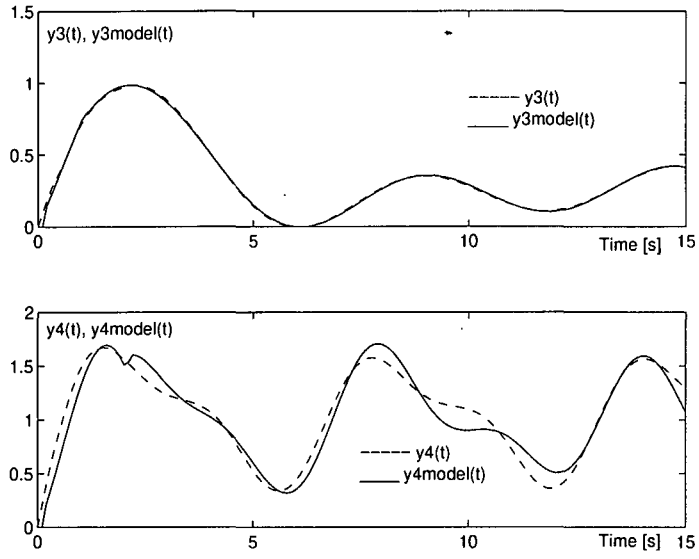


Figure 4-13: Results of Phase 1, outputs #3, 4

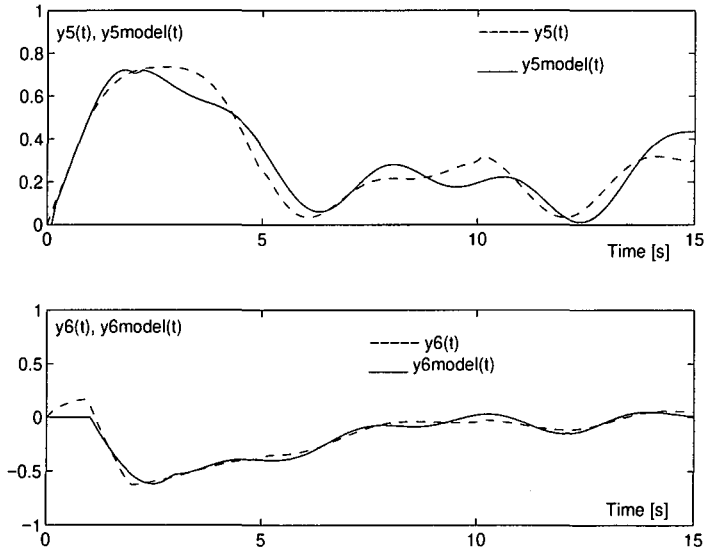


Figure 4-14: Results of Phase 1, outputs #5, 6

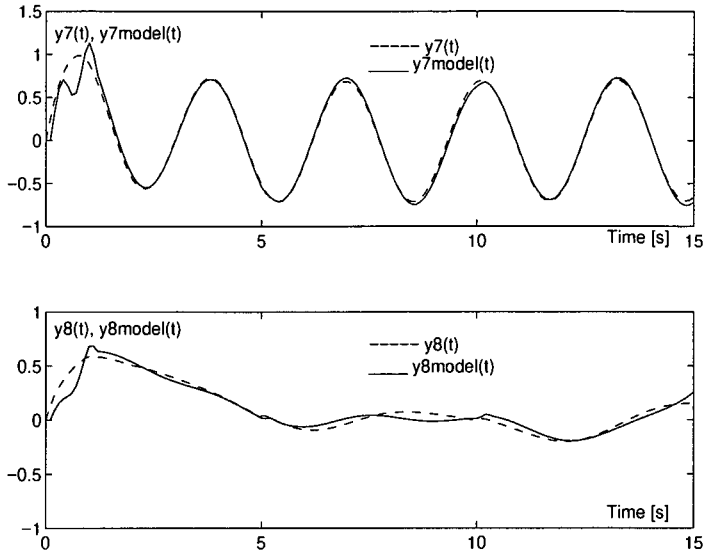


Figure 4-15: Results of Phase 1, outputs #7, 8

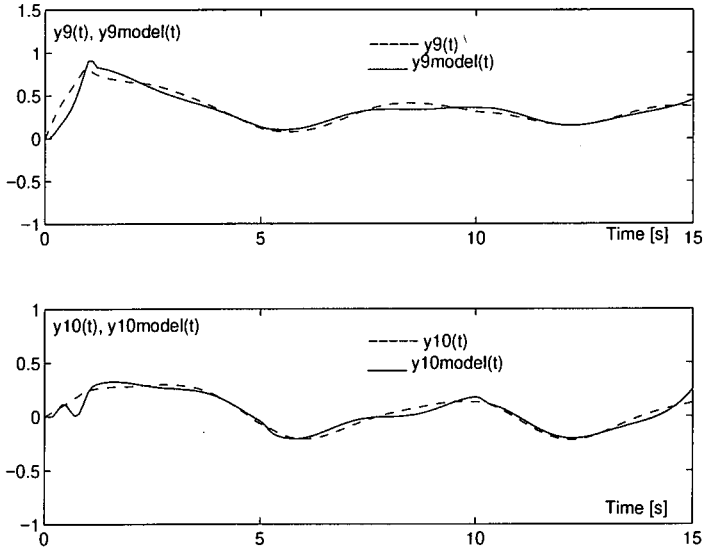


Figure 4-16: Results of Phase 1, outputs #9, 10

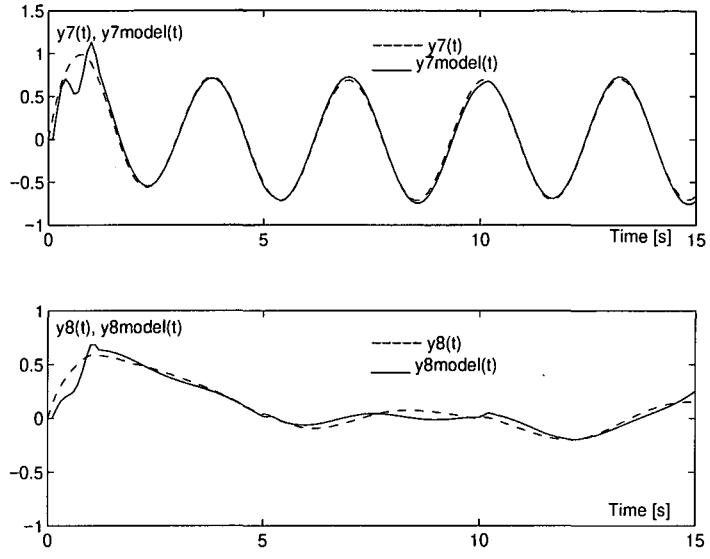


Figure 4-15: Results of Phase 1, outputs #7, 8

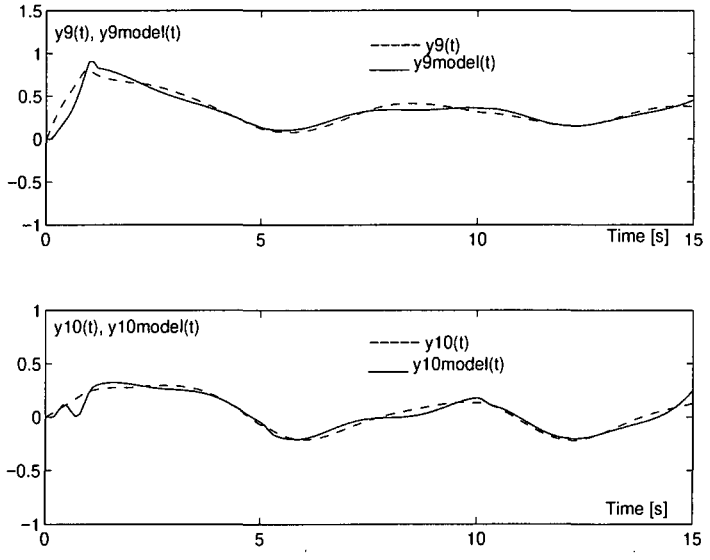


Figure 4-16: Results of Phase 1, outputs #9, 10

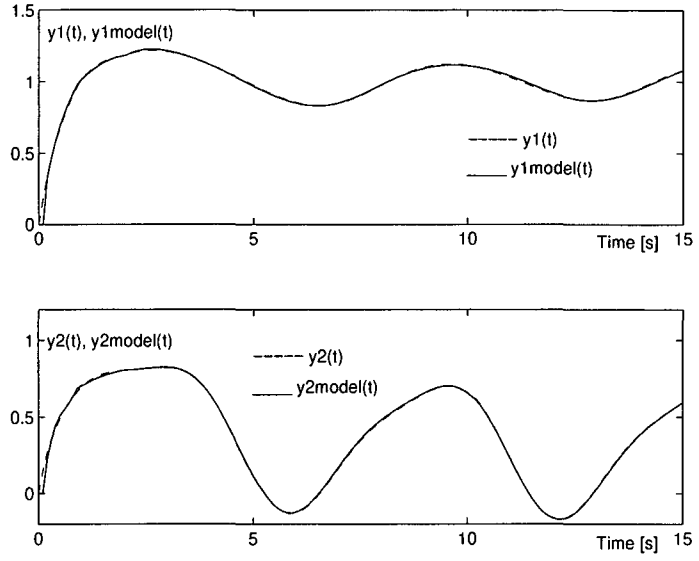


Figure 4-17: Results of Phase 2, outputs #1, 2

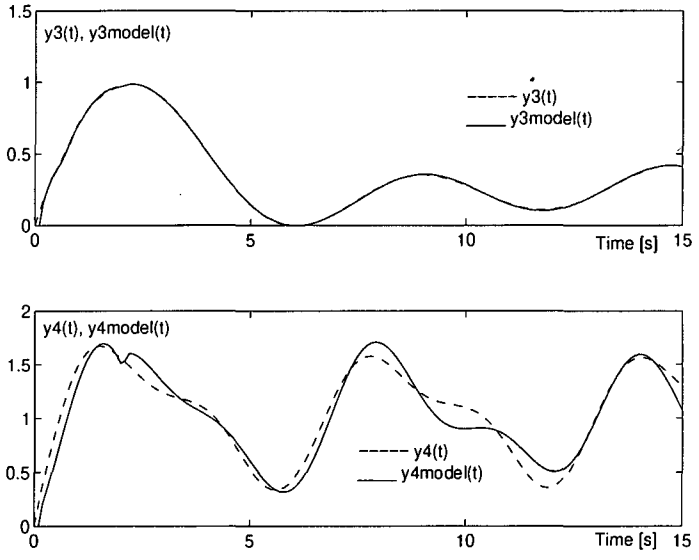


Figure 4-18: Results of Phase 2, outputs #3, 4

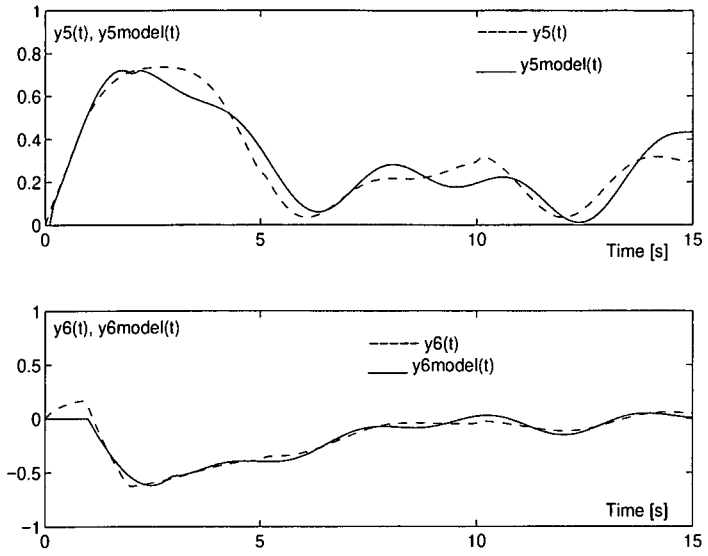


Figure 4-19: Results of Phase 2, outputs #5, 6

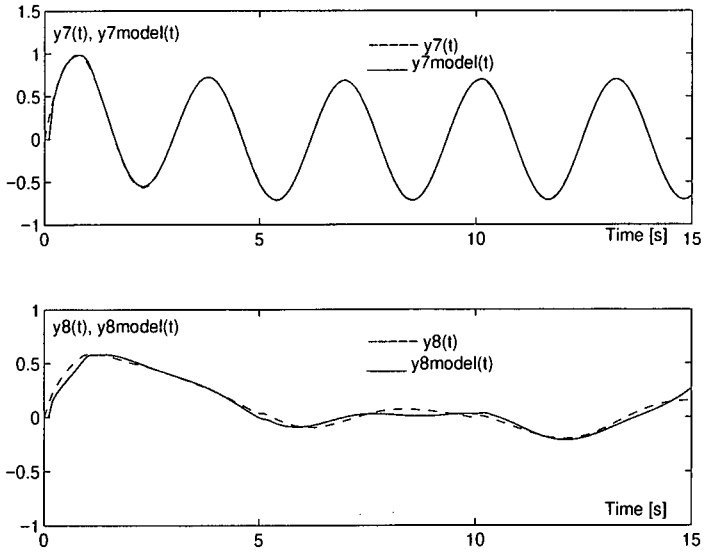


Figure 4-20: Results of Phase 2, outputs #7, 8

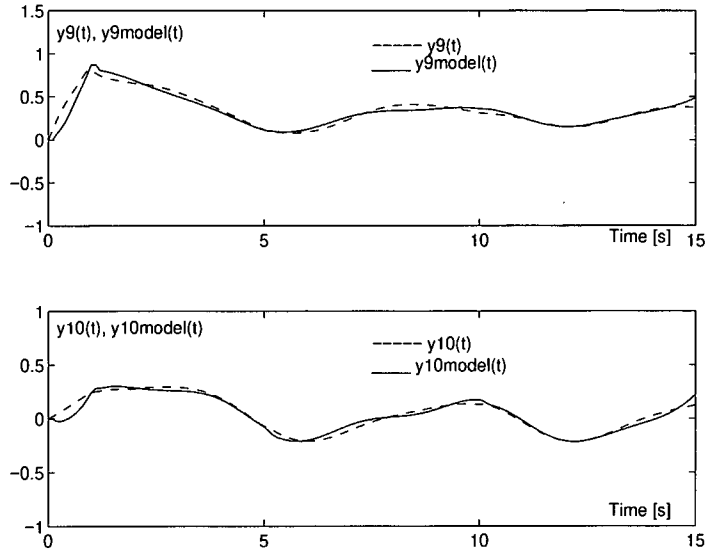


Figure 4-21: Results of Phase 2, outputs #9, 10

inputs and outputs. Identification utilizing this approach has given very good results. Possibilities regarding optimization (in the sense of its size) of input the distribution matrix should be further investigated, as this would allow for utilization of global output feedback without significant increase in the number of parameters.

Chapter 5

Conclusion

The main goal of this study is to develop a method for reducing optimization complexity associated with the optimization-based identification of LSS. This complexity depends crucially on the number of parameters of the model that simultaneously enter the optimization routines in the process of identification. If one-shot identification approach is employed this number of parameters may be very large. The consequence of this is that such identification may take too long time to complete and the numerical difficulties could be significant. It was therefore necessary to consider alternative approaches to the one-shot identification of LSS.

The approach investigated in this study is the one of hierarchical identification. The aim of the hierarchical identification method developed was to decompose the initial identification problem into a set of smaller-size identification problems. In addition to solving the smaller-size problems, another important objective was to solve some of them in parallel, thus greatly increasing the speed of such identification. The result of these considerations was a two-phase hierarchical identification procedure presented in Chapters 3 and 4. Although conceptually very similar, separate procedures have been developed for hierarchical identification of SISO and MIMO systems, in order to address some specific difficulties associated with breaking up the initial identification problem into smaller size problems in each of these cases.

The main problem with decomposition of the identification problem was that the model of some physical process is the output of the identification process, and, therefore, there is initially nothing to decompose. The problem has been addressed by identifying composite models which would enable parallel identification of some of its components (subsystems). These subsystems were identified in the first phase of the hierarchical identification. In the second phase, interactions among the subsystems were determined. The choice of composite models depended on whether the identified model is a model of a SISO or MIMO system.

Hierarchical identification of a SISO system poses particular conceptual difficulties as it has one measurable input and one measurable output, so there are no clues from input-output sequences about the possible ways of decomposing the system. In the case of a multivariable system, some information about the existence and interactions of subsystems may be deduced by studying the correlation between input and output signals of that system. In the first phase of hierarchical identification of SISO systems, transfer functions of various orders are identified. As each of these identification problems shares the same set of input-output data, identification of these subsystem transfer functions can be done in parallel. This would constitute Phase 1 of the hierarchical identification of SISO systems. As these subsystems are of relatively low order, and as such may not capture the dynamics of the system adequately, they are coupled together via an interaction module into a composite model. The composite model is usually chosen in such a way that the resulting composite model's order is equal to the sum of subsystem orders. The dynamics of the composite model are improved by an interaction module which employs output feedback. Identification of parameters of the interaction module is done in Phase 2 of the identification procedure. Four types of composite models were studied: the composite model with fixed, non-zero dead-time for the subsystems, the composite model with subsystem delays fixed to zero, reduced parameter composite model utilizing the local output feedback for each of the subsystems and the composite model which employs a dynamic system as a mean of improving overall model's dynamics

instead of output feedback. Of all of these approaches the one with subsystem dead-times fixed to zero has shown the best results and it yield models of similar quality to one-shot identification approach, but with smaller computational effort. The worst approach amongst those described in this study appears to be the one employing a dynamic system with no feedback loop.

A similar approach was taken w.r.t. MIMO systems. The main difference in this case is that subsystems of the composite model may not have all the necessary input and output information, but only a portion of it. These subsystems are identified in parallel by multivariable models in canonical form. The coupling among the subsystems is done via an input distribution matrix which provides the subsystems with all the information about inputs and via a vector which combines the outputs of all subsystems into the output of the overall composite model. The composite model dynamics are improved by utilizing local output feedback for each of the subsystems. In this way the number of parameters for identification of the interaction module is significantly decreased. This approach has been shown to work well and is numerically much more efficient than one-shot identification.

Chapter 6

REFERENCES

Allemong, J. J. and P. V Kokotovic (1980). "Eigensensitivities in Reduced Order modelling", *IEEE Trans. on Autom. Control*, Vol. 25, pp. 821-822

Anderson, B. D. O. and Y. Liu (1989). "Controller Reduction: Concepts and Approaches", *IEEE Trans. on Autom. Control*, Vol. 34, pp. 802-812

Aoki, M. (1978). "Some Approximation Methods for Estimation and Control of Large Scale Systems", *IEEE Trans. on Automatic Control*, Vol. 23, pp. 178-182

Aoki, M. (1971). "Aggregation", in *Optimization Methods for Large-Scale Systems* (ed. D. A. Wismer), McGraw-Hill, pp. 191-232

Arnold, S. F. (1990). "Mathematical Statistics", Prentice Hall International

Aström, K. J., P. Hagander and J. Sternby (1984). "Zeros of Sampled Systems", *Automatica*, Vol. 20, pp. 31-38

Astrom, K. J. and T. Hagglund (1988). "Automatic Tuning of PID Controllers", *Instrument Society of America* (ISA)

Audley, D. R. and D. A. Lee (1974). "Ill-Posed and Well Posed Problems in System Identification", *IEEE Trans. on Automatic Control*, Vol. 19, page 738

Bai, E. and K. M. Nagpal (1995). "Least Squares Type Algorithms for Identification in the Presence of modelling Uncertainty", *IEEE Trans. on Autom. Control*, Vol. 40, pp.756 - 761

Bajić, V.B. (1994) Private communication on the concept of hierarchical identification and identification of large-scale systems, Durban.

Bajić, V. B. (1989). "A Contribution to the Methods of Qualitative Analysis of Some Classes of Electrical Networks Governed by Semistate Models", (in Serbian), Doctoral dissertation, Faculty of Electrical Engineering, University of Zagreb, Zagreb, Yugoslavia

Bajić, V. B. (1996). "Models of Large-Scale Singular Electrical Networks Suitable for Qualitative Analysis", *Mathematical Modelling and Scientific Computing (*Mathematical Modelling and Scientific Computing in Science and Technology; X.J.R.Avula and Anil Nerode, Editors), Vol.6

Bajić, V. B. and B.Janković (1997). "Hierarchical identification of large-scale process plants", accepted for IFAC-IFIP-IMACS Conference, Belfort, France, May 20-22

Bajić, V. B. and B. J. Petrović (1980). "Quasi-Models of Price Evolution and Their Qualitative Properties", in the book *Global Modelling*, Edited by S.Krčevinac, Lecture Notes in Control and Information Sciences, Springer-Verlag, New York, pp.198-205

Bertsekas, D. P. and J. N. Tsitsiklis (1989). "Parallel and Distributed Computation - Numerical Methods", Prenice-Hall

Bingulac S. P. (1976). "On Mimimal Number of Parameters in Linear Multivariable Systems", *IEEE Trans. on Autom. Control*, Vol. 21 (1976), page 604

Bingulac, S. P., D. L. Cooper (1991). "Use of Pseudo-Observability Indices in Identification of Continuous-Time Multivariable Models", in *Identification of Continous-*

Time Systems (editors N. K. Sinha and G. P. Rao), Kluwer Academic Publishers, Dordrecht, pp. 443-472

Boje, E. (1991). "Identification of Multivariable Continuous-Time Systems", in *Identification of Continuous-Time Systems* (editors N. K. Sinha and G. P. Rao), Kluwer Academic Publishers, Dordrecht, pp. 419-442

Božić, S. M. (1979). "Digital and Kalman Filtering", Edward Arnold, London

Brown, N. R. and J. P. Norton (1982). "Identifiability of large compartmental models", *Large Scale Systems*, Elsevier North Holland, Vol. 3, pp. 159-176

Brucoli, M., F. Torelli and M. Trovato (1987). "Hierarchical computation of decentralized controllers for interconnected power systems", *Large Scale Systems*, Elsevier North Holland, Vol. 9

Buddin, M. A. (1971). "Minimal Realization of Discrete Linear Systems From Input-Output Observations", *IEEE Trans. on Autom. Control*, Vol. 16, page 395

Burley, D. M. (1974). "Studies in Optimization", International Textbook Company

Chen, C. T. (1984). "Linear System Theory and Design", Holt-Rinehart and Winston

Conwey, J. H. (1971). "Regular Algebra and Finite Machines", Chapman and Hall, London

Correa, G. O. and K. Glover (1986). "On the Choice of parameterisation For Identification", *IEEE Trans. on Autom. Control*, Vol. 31, pp. 8 -16

Coxson, P. (1985). " Model Reduction: Identifying Partitions for Structured Aggregates", *IEEE Trans. on Autom. Control*, Vol. 30, pp. 478-480

Crorkin, W. D., K. Malinowski, A. Y. Alladina and Singh M. G. (1985). "Decomposition-Coordination Techniques For Parallel simulation, Part 2: Simulation of Gas Flow in a Pipeline", *Large Scale Systems*, Elsevier North Holland, Vol. 9, pp. 117-127

Davies, W. D. T. (1970). "System Identification for Self Adaptive Control", Willey - Interscience, London

Davison, E. J. (1970). "On Pole Assignment in Linear Systems With Incomplete State Feedback", *IEEE Trans. on Autom. Control*, Vol. 15, pp. 348 - 351

Davison, E. J. (1966). "A Method for Simplifying Linear Dynamic Systems", *IEEE Trans. on Autom. Control*, Vol. 11, pp. 93-101

Davison, E. J. and R. Chatterjee (1971). "A Note on Pole Assignment in Linear Systems With Incomplete System Feedback", *IEEE Trans. on Autom. Control*, Vol. 16, pp. 98-99

Davison, E. J. and S. H. Wang (1975). "On Pole Assignment in Linear Multivariable Systems Using Output Feedback", *IEEE Trans. on Autom. Control*, Vol. 20, pp. 516-518

De Carvalho, M. (1993). "Dynamical Systems and Automatic Control", Prentice-Hall

Denham, M. J. (1974). "Canonical Forms for Identification of Multivariable Linear Systems", *IEEE Trans. on Autom. Control*, Vol. 19, page 646

Dickinson, B. W., T. Kailath and M. Morf (1974). "Canonical Matrix Fraction and State-space description for Deterministic and Stochastic Linear Systems", *IEEE Trans. on Autom. Control*, Vol.19, p. 656

Doram, A. I. (1987). "Coordinate Selection Issues in the Order Reduction of Linear Systems", *Control and Dynamic Systems,* editor C. T. Leondes, Vol. 25, pp. 309-356

Drenick, P. E. (1975). "On the Decomposition of State Space", *IEEE Trans. on Automatic Control*, Vol. 20, pp. 269-271

Eitelberg, E. (1982). "Comments on Model Reduction by Minimizing the equation error", *IEEE Trans. on Autom. Control*, Vol. 27, pp. 1000-1002

El-Sharif, H. and M. A. Maud (1984). "Dedicated Microprocessors for Real-time Identification of Multivariable Systems", *Automatica,* vol 20, pp. 129-131

Elrazaz, Z. and N. K. Sinha (1979). "On the Selection of Dominant Poles of a System to be Retained in Lower Order Model", *IEEE Trans. on Autom. Control*, Vol. 24, pp. 792-793

Enrigh, W. H. and M. S. Kamel (1980). "On Selecting a Low-Order Model Using the Dominant Pole Concept", *IEEE Trans. on Autom. Control*, Vol. 25, pp.976-978

Fletcher, R. (1980). "Practical Methods of Optimization", Vol. 1, John Wiley and Sons

Fortmann, T. E. and K. L. Hiltz (1977). "An Introduction to Linear Control Systems", Marcel Dekker, New York

Franklin, G. F., J. D. Powell and A. Emami-Naeini (1991). "Feedback Control of Dynamic Systems", Addison-Wesley

Gajić, Z., D. Petkovski and X. Shen (1990). "Singularly Perturbed and Weakly Coupled Linear Control Systems", Springer-Verlag

Gevers, M and V. Wertz (1984). "Uniquely Identifiable State-Space and ARMA parameterisation for Multivariable Linear Systems", *Automatica*, vol 20, pp. 337-347

Gill, P. E. and W. Murray (1974). "Methods for Large-Scale Linearly Constrained Problems", in *Numerical Methods For Constrained Optimization*, editor P. E. Gill and W. Murray, Academic Press

Glover, K. and J. C. Willems (1976). "parameterisation of Linear Dynamical Systems: Canonical Forms and Identifiability", *IEEE Trans. on Autom. Control*, Vol. 19, p. 640

Godfrey, K. (1993). "Perturbation Signals for System Identification", Prentice-Hall

Goodwin, G. C. and R. L. Payne (1977). "Dynamic System Identification - Experiment Design and Data Analysis", Academic Press

Grasselli, O. M. and A. Tornambe (1992). "On Obtaining a Realization of Polynomial Matrix Description of System", *IEEE Trans. on Autom. Control*, Vol. 37, pp. 852-856

Grujić, Lj. T., A. A. Martynyuk and M. Ribbens-Pavella (1987). "Large Scale System Stability under Structural and Singular Perturbations", Springer-Verlag

Grewal, M. S. and K. Glover (1976). "Identifiability of Linear and Nonlinear Dynamical Systems", *IEEE Trans. on Autom. Control*, Vol.21, p. 833

Guardbassi, G. (1982). "Composite problem analysis", *Large Scale Systems*, Elsevier North Holland, Vol. 3, pp. 1-11

Gupta, R. D. and F. W. Fairman (1974). "Parameter Estimation for Multivariable Systems", *IEEE Trans. on Autom. Control*, Vol. 19, p.546

Hallager, L., L. Goldschmit L. and S. B. Jorgensen S. B. (1984). "Multivariable adaptive identification and control of a distributed chemical reactor", *Large Scale Systems*, Elsevier North Holland, Vol. 6, pp. 323-336

Hasiewicz, Z. and A. Stankiewicz (1986). "On Applicability of Interaction Balance Method to Global Identification of Interconnected Steady-State Systems", *IEEE Trans. on Autom. Control*, Vol. 31, pp.77 - 80

Hatcher, W. S. (1968)."Foundations of Mathematics", W. B. Saunders Company

Heij, C. (1989). Deterministic Identification of Dynamical Systems", Springer-Verlag

Hickin, J. and N. K. Sinha (1975). "Aggregation Matrices for a Class of of Low-Order Models for Large-Scale Systems", *Electron. Lett.* , Vol. 11, p. 186

Hickin, J. and N. K. Sinha (1980). "Model Reduction for Linear Multivariable Systems", *IEEE Trans. on Autom. Control*, Vol. 25, pp. 1121-1127

Hirshberg, D. and N. Merhav (1996). "Robust Methods for Model Order Estimation", *IEEE Trans. on Signal Processing*, vol 44, pp. 620-628

Hsia, T. C. (1977). "System Identification - Least Square Method", Lexington Books

Jameson, A. (1970). "Design of a Single Input System for Specified Roots Using Output Feedback", *IEEE Trans. on Automatic Control*, Vol. 15, pp. 345-348

Janković, B. and V.B.Bajić (1996a). "Reduced optimization complexity in identification of large-scale SISO processes", *Proceedings of the International AMSE Conference on Communications, Signals and Systems* (CSS'96), Vol.2, pp.321-325, Brno, Czech Republic, September 10-12, 1996

Janković, B. and V.B.Bajić (1996b). "Parallel Hierarchical Algorithm for Identification of Large-Scale Industrial Systems", *Proc. of the 1996 National Research and Development Conference, Durban,* 26-27 Septembar (ed. V. Ram)

Janković, B. and V.B.Bajić (1997a). "Contibution to parallel hierarchical identification of large-scale system models", 11^{th} Conference on Mathematical and Computer Modelling and Scientific Computing, St. Louis, USA, accepted

Janković, B. and V.B.Bajić (1997b). "Selection of the feedback structure for hierarchical identification of large-scale systems", 11^{th} Conference on Mathematical and Computer Modelling and Scientific Computing, St. Louis, USA, accepted

Janković, B. and V.B.Bajić (1997c). "Selection of input signals for identification of large-scale systems", submitted, The Third International Conference on Modelling and Simulation, MS '97, October, 29-31, Melbourne-Australia

Janković, B. and V.B.Bajić, (1997d). "Computational complexity reduction in large-scale system identification", submitted, Third International Confeence on Modelling and Simulation, Melbourne, Australia

Kecman, V. (1988). "State-Space Models of Lumped and Distributed Systems", Springer-Verlag

Kimura, H. (1975). "Pole Assignment By Gain Output Feedback", *IEEE Trans. on Autom. Control*, Vol. 20, pp. 509-516

Kreisselmeier G. and M. Mavenkamp (1988). "A Note on Reduced Order Controller Synthesis:, *IEEE Trans. on Autom. Control*, Vol. 33, pp. 878-880

Kron, G. (1963). "Diakoptics", Macdonald, London

Kudva, P. and K. S. Narendra (1974). "An Identification Procedure for Discrete Multivariable Systems", *IEEE Trans. on Autom. Control*, Vol. 19, p. 549

Lancaster, P. and M. Tismenetsky M. (1985). "The Theory of Matrices", 2nd edition, Academic Press

Lee, T. H., Q. K. Wang and E. K. Koh (1994). "An Iterative Algorithm for Pole Placement by Output Feedback", *IEEE Trans. on Autom. Control*, Vol. 39, pp 565-568

Lei, S, A. Y. Allidina and K. Malinowski K. (1986). "Clustering Technique for Rearranging ODE Systems", in *Parallel Processing Techniques for Simulation* (editors M. A. Singh, A. Y. Allidina and B. K. Daniels), Plenum Press, pp. 31-44

Levine, W. S. and M. Athans (1970). "On the Determination of the Optimal Constatnt Output Feedback Gains for Linear Multivariable Systems", *IEEE Trans. on Automatic Control*, Vol. 15, pp. 44-48

Liu, R. and L. C. Suen (1977). "Minimal Realization and Identifiability of Input-Output Sequences", *IEEE Trans. on Autom. Control*, Vol. 22, p. 227-232

Ljapunov, A. M. (1956). "General Problem of Stability of Motion", (in Russian), Moscow, Leningrad: USSR Academy of Science

Ljung, L. (1987). "System Identification: Theory for the User", Prentice-Hall, 1987, *IEEE Trans. on Autom. Control*, Vol. 21, p.779

Ljung, L. (1976). "Consistency of the Least-Squares Identification Method", *IEEE Trans. on Automatic Control*, Vol. 22, pp. 779-781

Lynn, P. A. (1982). "An Introduction to Analysis and Processing of Signals", The MacMillan Press

Mahapatra, G. B. (1977). "A Note on Selecting a Low-Order System by Davisons's Model Simplification Technique", *IEEE Trans. on Automatic Control*, Vol. 22, pp. 677-678

Mahmoud, M.S., M. F. Hassan and M. G. Darwish (1985). "Large-Scale Control Systems - Theories and Techniques", Marcel Dekker

Malinowski, K., A. Y. Alladina, M. G. Singh and W. D. Crorkin (1985). "Decomposition-Coordination Techniques For Parallel simulation, Part 1", *Large Scale Systems*, Elsevier North Holland, Vol. 9, pp. 101-115

Malinowski, K., A. Y. Alladina and M. G. Singh (1986). "Decomposition-Coordination Techniques for Parallel Simulation" in *Parallel Processing Techniques for Simulation* editors M. A. Singh, A. Y. Allidina and B. K. Daniels, Plenum Press, pp. 1-11

Mayne, D. Q. (1972). "A canonical model for identification of multivariable linear systems", *IEEE Trans. on Automatic Control*, Vol. 17, pp. 728-729

Meier, L. and D. G. Luenberger (1967). "Approximation of Linear Constant Systems", *IEEE Trans. on Automatic Control*, Vol. 12, pp. 585-588

Mesarović, M. D. and Y. Takahara (1989) "Abstract System Theory", Springer-Verlag

Milić, M. M. and V. B. Bajić, (1987). "Qualitative analysis of motion properties of semistate models of large-scale systems", *Circuits, Systems and Signal Processing*, Vol.6, No.3, pp. 315-334

Moore, B. C. and L. M. Silverman (1972). "Model Matching by State Feedback and Dynamic Compensation", *IEEE Trans. on Automatic Control*, Vol. 17, pp. 491-497

Niv, S. S. and D. G. Fisher (1996). "Recursive Information Forgetting With Augmented UD Identification", *Int. Journal of Control*, Vol. 63, pp. 623-637

Nünnes, B. M. and G. C. Goodwin (1991). "The Relationship Between Discrete Time and Continuous Time Linear Estimation", in *Identification of Continuous-Time Systems,* Editors N. K. Sinha and G. P. Rao, Kluwer Academic Publishers, Dordrecht, pp. 79-122

Obinata, G. and H. Inooka (1976). " A Method for modelling Linear Time-Invariant Systems by Linear Systems of Low Order", *IEEE Trans. on Automatic Control*, Vol. 21, pp.602 - 603

Ogata, K. (1970). "Model Control Engineering", Prentice-Hall

O'Flynn, M. (1990). "Probabilities, Random Variables, and Random Processes", John Wiley and Sons

Oppenheim, A. V., R. W. Schaffer (1975). "Digital Signal Processing", Prentice-Hall, Englewood Cliffs, N. J.

Ouyang, M., C. M. Liaw and C. T. Pan (1987). "Model Reduction in Power Decomposition and Frequency Response Matching", *IEEE Trans. on Autom. Control*, Vol. 32, pp. 59-62

Owens, D. H. (1978). "Feedback and Multivariable Systems", Peter Peregrinus

Palm, W. (1986). "Control Systems Engineering", John Wiley and Sons

Papoulis, P. (1977). "Signal Analysis", McGraw Hill

Parnebo, L. and L. M. Silverman (1982). "Model Reduction via Balanced State-Space Representation", *IEEE Trans. on Autom. Control*, Vol. 27, pp. 382-387

Pearse, J. G., P. Holliday P and J. O. Gray (1986). "Survey of Parallel Processing in Simulation", in *Parallel Processing Techniques for Simulation* editors M. A. Singh, A. Y. Allidina and B. K. Daniels, Plenum Press, pp. 183-202

Pearson, J. D. (1971). "Dynamic decomposition techniques", in *Optimization Methods for Large-Scale Systems* editor D. A. Wismer, McGraw-Hill, pp. 121-190

Pollard, J. P. and C. B. Brosilow (1985). "Model Selection for Model Based Controllers", Proc. American Control Conference, Boston

Pooch, U. V. and A. Nieder (1973). "A Survey of indexing techniques for sparse matrices", *Computer Surveys* Vol. 5

Poola, K., P. Khargonekar, A. Tikku, J. Krause and K. Nagpal (1994). "A Time Domain Approach To Model Validation", *IEEE Trans. on Autom. Control*, Vol. 39, pp. 951-959

Priestley, M. B., S. T. Rao and T. Howell (1974). "Application of Principal Component Analysis and Factor Analysis in the Identification of Multivariable Systems", *IEEE Trans. on Autom. Control*, Vol. 19, p. 730

Prywes, N. S. (1984). "Distributed large scale computation: With an application to World-wide econometric studies", *Large Scale Systems*, Elsevier North Holland, Vol. 7, pp. 89-97

Rajagoplan, T. (1984). "Pole Assignment With Output Feedback", *Automatica*, vol 20, pp. 127-128

Rao, G. P. and N. K. Sinha (1991). "Continuous-Time Models and Approach", in *Identification of Continuous-Time Systems,* editors N. K. Sinha and G. P. Rao, Kluwer Academic Publishers, Dordrecht, pp. 1-16

Raven, F. H. (1978). "Automatic Control Engineering", McGraw-Hill

Ray, L. M., D. P. Mital and V. L. Skhrikhande (1980). "On Minimal Canonical Realization from Input-Output Sequences", *IEEE Trans. on Autom. Control*, Vol. 25

Rubio, J. E., (1971). "Theory Of Linear Systems", Academic Press, New York

Sage, A. P. (1987). "Software as large-scale system", *Large Scale Systems*, Elsevier North Holland, Vol. 12, pp. 185-188

Sargent, T. J. (1983), foreword to the second edition of *Prediction and Regulation by Least-Square Methods* by P. Whittle

Schoeffler, J. D. (1971). "Static Multilevel Systems", in *Optimization Methods for Large-Scale Systems,* editor D. A. Wismer, McGraw-Hill, pp. 1-46

Shaw, P. (1981). "Logic and its Limits", Pan Books, London

Simon, H. A. (1962). "The Architecture of Complexity", *Proc.of American Philosophical Society*, 106, pp. 467-482

Sinha, N. K. and G. J. Lastman (1991). "Transformation of Discrete-Time Models", in *Identification of Continuous-Time Systems,* editors N. K. Sinha and G. P. Rao, Kluwer Academic Publishers, Dordrecht, pp. 123-138

Singh, M. G., A. Titli and K. Malinowski (1986). "Decentralized control design: an overview:, *Large Scale Systems*, Elsevier North Holland, Vol. 9, pp. 215-230

Smith, O. J. M. (1957). "A controller to overcome dead-time", ISA Journal, Vol. 6

Söderström, T., L. Ljung L. and I. Gustavsson (1976). "Identifiability Conditions for Linear Multivariable Systems Operating Under Feedback", *IEEE Trans. on Autom. Control*, Vol. 21, p. 837

Söderström, T. and P. Stoiça (1989). "System Identification", Prentice-Hall, Englewood Cliffs, N. J.

Sontag, E. D. (1980). "On the Length of Inputs Necessary in Order to Identify a Deterministic Linear System", *IEEE Trans. on Autom. Control*, Vol. 25, p. 120

Stals, L.C. (1996). in Governor's Address at the 76th Ordinary General Meeting of the shareholders of the South African Reserve Bank, held on 27/8/1996

Sultan, M. A., M. F. Hassan, J. L. Calvet and A. Y. Bilal (1985)."A First Order Parallel Decomposition Method for Parameter Estimation Large Scale Systems", *Large Scale Systems*, Elsevier North Holland, Vol. 9, pp. 51 - 62

Sundareshan, M. K. (1976). "A Condition for Decentralization in Model Reference Adaptive Systems", *IEEE Trans. on Autom. Control*, Vol. 21, p. 880

Šiljak, D. D. (1978). "Large-Scale Systems - Stability and Structure", North-Holland, New York

Šiljak, D. D. (1984). "Complex dynamic systems: dimensionality, structure and uncertainty", *Large Scale Systems*, Elsevier North Holland, Vol. 4, pp. 279-294

Takahara, Y. (1982). "Decomposability conditions for general systems", *Large Scale Systems*, Elsevier North Holland, Vol. 3, pp. 57-65

Tse, E. and H. L. Wienert (1975). "Structure Determination and Parameter Estimation for Multivariable Stochastic Linear Systems", *IEEE Trans. on Autom. Control*, Vol. 20, p. 603

Tuch, J., A. Feuer and Z. Palmor (1994). "Dead-Time Estimation in Continuous Linear Time-Invariant Systems", *IEEE Trans. on Automatic Control*, Vol. 39, pp.823-827

Tzafestas, S. G. and P. N. Paraskevopoulos (1976)."On Exact Model Matching Controller Design", *IEEE Trans. on Autom. Control*, Vol. 21, p. 242

Vardulakis, A. I. (1975). "A Sufficient Condition for n Specified Eigenvalues to be Assigned Under Constant Output Feedback", *IEEE Trans. on Automatic Control*, Vol. 20, pp. 428-429

Vitacco, W. R. and A. N. Michel (1977). "Qualitative analysis of interconnected systems with algebraic loop: well-posedness and stability", *IEEE Trabsactions on Circuits and Systems,* Vol. CAS-24, pp625-637, 1977

Wang, S. H. and E. J. Davison (1973). "On the Stabilization of Decentralized Control Systems", *IEEE Trans. on Automatic Control*, Vol. 18, pp. 473-478

Whittle, P. (1983). "Prediction and Regulation by Linear Least-Square Methods", University of Minnesota Press

Wilson, D. A. (1970). "Optimum Solution of Model Reduction Problem", *Proc. of Institute of Electrical Engineering*, Vol. 117, pp. 1161-1165

Wonham, W. M. (1967). "On pole assignment in multi-input controllable linear systems", *IEEE Trans. on Autom. Control*, Vol. 12, pp. 660-665

Wu, Y. C. and Z. V. Rekasius (1980) "Deterministic Identification of Linear Dynamical Systems", *IEEE Trans. on Autom. Control*, Vol. 25, p. 501

Young, P. C.and C. Wang (1987). "Identification and estimation of multivariable dynamic systems", in "Multivariable control for industrial applications", Edited by J. O'Reilly, Peter Peregrinus, pp. 244-279

Youwana, M. and D. E. Seborg (1982). "A New Method for On-Line Controller Tuning", *AICHE Journal*, Vol. 28, p. 434

Zhang, H. M. (1989). "Order-Determination Theorems and System Identification", *IEEE Trans. on Autom. Control*, Vol. 33, pp. 1078-1082

Appendix A

Least-Squares Method

Suppose that variable y is given as

$$y(t) = \theta_1 x_1(t) + \ \ldots \ + \theta_n x_n(t) \tag{A.1}$$

This equation can be written in more compact form as

$$y(t) = \mathbf{x}\boldsymbol{\theta} \tag{A.2}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ are variables and $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)^T$ is a vector of constant parameters that we want to determine. Suppose also that we have m input-output measurements. We can then write

$$\begin{aligned} y_1(t) &= \theta_1 x_1^1(t) + \ \ldots \ + \theta_n x_n^1(t) \\ y_2(t) &= \theta_1 x_1^2(t) + \ \ldots \ + \theta_n x_n^2(t) \\ &\vdots \qquad\qquad \vdots \\ y_m(t) &= \theta_1 x_1^m(t) + \ \ldots \ + \theta_n x_n^m(t) \end{aligned} \tag{A.3}$$

or, in more compact form

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} \tag{A.4}$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_m)^T$, $\boldsymbol{\theta}$ is parameter vector from (A.1), and matrix $\mathbf{X}$ is an $m \times n$ matrix

$$\begin{bmatrix} x_1^1 & x_2^1 & \cdot & \cdot & \cdot & x_n^1 \\ x_1^2 & x_1^2 & \cdot & \cdot & \cdot & x_1^2 \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ x_1^m & x_1^m & \cdot & \cdot & \cdot & x_1^m \end{bmatrix} \tag{A.5}$$

It is clear immediately that in the case of $m = n$, there is unique solution of (A.4), given by

$$\hat{\theta} = \mathbf{X}^{-1}\mathbf{Y} \tag{A.6}$$

where $\hat{\theta}$ is the optimal vector of parameters, providing that matrix $\mathbf{X}$ is nonsingular. In practice, samples may be corrupted by noise, and to obtain better estimates, the number of observations m is normally higher than the number of parameters n. In this case the solution of (A.3) is said to be overdetermined. To the parameters vector $\hat{\theta}$, we define the error as equation error of (A.2):

$$\boldsymbol{\varepsilon} = \mathbf{y} - \mathbf{X}\boldsymbol{\theta} \tag{A.7}$$

The m-component vector ε contains the differences between the measured variable y and its estimate with current values of θ. We can define error as the sum of squares of these differences

$$J = \sum_{i=1}^{m} \varepsilon_i^2 = \boldsymbol{\varepsilon} \cdot \boldsymbol{\varepsilon}^T \tag{A.8}$$

By substituting ε from (A.7) into (A.8) we get

$$J = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T \tag{A.9}$$

It is now necessary to find θ such that E is minimal. In other words,

$$\frac{dJ}{d\theta}_{|\theta=\hat{\theta}} = 0 \tag{A.10}$$

By multiplying the factors from (A.9), and then derivating with respect to θ, (A.10) becomes

$$-2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\hat{\theta} = 0$$

from which we finally obtain

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{A.11}$$

This expression is called least squares estimator (LSE) for $\boldsymbol{\theta}$.

Appendix B

Models Used in Examples

B.1 SISO Systems

B.1.1 Transfer Function Models Obtained for 3-Channel Autopilot System

Example 2.3, Section 2.2.4

First order transfer function

$$\frac{Y_1(s)}{U(s)} = \frac{4.4496}{s + 4.1177} e^{-0.24s}$$

Second order transfer function

$$\frac{Y_2(s)}{U(s)} = \frac{1.1868s + 1.1189}{s^2 + 0.9201s + 1.1189}$$

Third order transfer function

$$\frac{Y_3(s)}{U(s)} = \frac{3.6471s^2 + 8.6297s + 6.9719}{s^3 + 7.4694s^2 + 6.0564s + 6.9719} e^{-0.18s}$$

Fourth order transfer function

$$\frac{Y_4(s)}{U(s)} = \frac{0.8302s^3 + 8.2375s^2 + 32.6406s + 30.0417}{s^4 + 2.9463s^3 + 29.9050s^2 + 24.4409s + 29.9163} e^{-0.09s}$$

Fifth order transfer function

$$\frac{Y_5(s)}{U(s)} = \frac{0.7552s^4 + 7.6143s^3 + 32.2447s^2 + 29.6436s + 0.0202}{s^5 + 2.8482s^4 + 29.7409s^3 + 24.0885s^2 + 29.6271s + 0.00002} e^{-0.07s}$$

Sixth order transfer function

$$\frac{Y_6(s)}{U(s)} = \frac{1.2068s^5 + 7.8619s^4 + 35.5826s^3 + 28.7388s^2 + 13.8817s^1 + 10.5878}{s^6 + 2.6703s^5 + 29.5836s^4 + 24.7543s^3 + 38.2778s^2 + 9.5855s^1 + 10.7152} e^{-0.11s}$$

Example 2.4, Section 2.2.4

First order transfer function

$$\frac{Y_1(s)}{U(s)} = \frac{2.5604}{s + 1.7538} e^{-0.15s}$$

Second order transfer function

$$\frac{Y_2(s)}{U(s)} = \frac{0.867s + 31.7569}{s^2 + 13.2143s + 22.4523} e^{-0.10s}$$

Third order transfer function

$$\frac{Y_3(s)}{U(s)} = \frac{0.8718s^2 + 53.7677s - 3.6218}{s^3 + 22.7273s^2 + 30.2846s + 3.1072} e^{-0.12s}$$

Fourth order transfer function

$$\frac{Y_4(s)}{U(s)} = \frac{1.4417s^3 + 8.0144s^2 + 36.2124s + 25.8867}{s^4 + 2.7328s^3 + 28.1223s^2 + 23.8377s + 25.3739} e^{-0.11s}$$

Fifth order transfer function

$$\frac{Y_5(s)}{U(s)} = \frac{1.7182^4 + 5.7349s^3 + 38.3006s^2 - 0.2595s + 4.2745}{s^5 + 1.9773s^4 + 26.0561s^3 + 13.8873s^2 + 11.1734s + 0.0001} e^{-0.12s}$$

Sixth order transfer function

$$\frac{Y_6(s)}{U(s)} = \frac{0.5990s^5 + 13.2384s^4 + 38.4462s^3 + 189.5375s^2 - 49.8058s^1 + 22.3757}{s^6 + 6.4258s^5 + 33.9285s^4 + 118.6467s^3 + 66.4417s^2 + 12.9024s^1 + 0.0032} e^{-0.05s}$$

B.1.2 Composite Models

Composite model in Example 3.1, Section 3.5

$$\mathbf{A}_{cm} = \begin{bmatrix} -1.0417 & 4.2614 & -0.7646 \\ -2.9362 & 2.8516 & 1.6395 \\ -1.2423 & 1.2727 & 0.5950 \end{bmatrix}$$

$$\mathbf{c}_{cm} = \begin{bmatrix} 0.0453 & -0.4421 & 1.6695 \end{bmatrix}$$

$$L_{proc} = \begin{bmatrix} 4.4532 & 0.7108 & 0.2315 \end{bmatrix}$$

It can be seen that dead-time parameters do not have any physical meaning, but serve

just as another parameter to minimize the error function.

Composite model used in Example 3.2 in Section 3.5

$$\mathbf{A}_{cm} = \begin{bmatrix} -0.2810 & -0.6854 & 0.0020 \\ -0.4341 & 0.3459 & -0.1431 \\ 0.2744 & 0.6583 & 0.1098 \end{bmatrix}$$

$$\mathbf{c}_{cm} = \begin{bmatrix} 0.7191 & -0.3374 & 0.5757 \end{bmatrix}$$

All dead-times are fixed to zero.

Composite model used in Example 3.3 in Section 3.5

$$\mathbf{A}_{cm} = \begin{bmatrix} -5.6497 & 129.1168 & -142.7312 \\ -0.2298 & -1.4748 & 1.1279 \\ -0.0823 & -4.0560 & 3.8988 \end{bmatrix}$$

$$\mathbf{c}_{cm} = \begin{bmatrix} 0.1130 & 4.9926 & -4.1708 \end{bmatrix}$$

$$L = 0$$

Composite model used in Example 3.4 Dead-times fixed to zero

$$\mathbf{A}_{cm} = \begin{bmatrix} 1.6095 & 3.9350 & -2.4923 \\ 0.4044 & 1.5070 & -0.9448 \\ 2.2476 & 0.6548 & -1.5494 \end{bmatrix}$$

$$\mathbf{c}_{cm} = \begin{bmatrix} 0.4571 & 0.6677 & -0.2198 \end{bmatrix}$$

$$L_{cm} = 1.0939$$

Dead-times fixed to 0.1[s]

$$\mathbf{A}_{cm} = \begin{bmatrix} 0.8253 & -12.8063 & 11.7148 \\ 0.0994 & -2.4366 & 2.1755 \\ -0.0401 & -1.8297 & 1.7175 \end{bmatrix}$$

$$\mathbf{c}_{cm} = [\ -0.0424 \quad 0.7989 \quad 0.3941\ ]$$

$$L_{cm} = 1.1144$$

Dead-times fixed to 0.25[s]

$$\mathbf{A}_{cm} = \begin{bmatrix} -6.5947 & -3.5783 & 1.0531 \\ 1.7354 & -11.3241 & 1.9641 \\ -18.0536 & 22.5531 & -3.0953 \end{bmatrix}$$

$$\mathbf{c}_{cm} = [\ 4.6688 \quad -3.8455 \quad 1.2396\ ]$$

$$L_{cm} = 0.9133$$

Dead-times fixed to 0.5[s]

$$\mathbf{A}_{cm} = \begin{bmatrix} 1.3779 & 4.1691 & -2.1631 \\ 0.2702 & 1.8159 & -0.8969 \\ 1.8158 & -0.4884 & -0.5392 \end{bmatrix}$$

$$\mathbf{c}_{cm} = [\ 0.2259 \quad 0.2250 \quad 0.2028\ ]$$

$$L_{cm} = 0.6547$$

Example 3.5 Dead-times fixed to 0[s]

$$\mathbf{A}_{cm} = \begin{bmatrix} -0.1394 & 0.0629 & -1.4646 \\ 0.2699 & -0.0899 & 2.0058 \\ 1.4925 & -0.9254 & 1.3484 \end{bmatrix}$$

$$\mathbf{c}_{cm} = [\ 0.5476 \quad 0.4419 \quad -0.0784\ ]$$

$$L_{cm} = 1.0977$$

Dead-times fixed to 0.1[s]

$$\mathbf{A}_{cm} = \begin{bmatrix} -0.4372 & 0.0879 & -1.8656 \\ 0.4585 & -0.0356 & 3.2465 \\ 1.3228 & -0.5155 & 0.4587 \end{bmatrix}$$

$$\mathbf{c}_{cm} = [\ 0.6118 \quad 0.3377 \quad 0.0150\ ]$$

$$L_{cm} = 1.0559$$

Dead-times fixed to 0.25[s]

$$\mathbf{A}_{cm} = \begin{bmatrix} -3.1335 & -1.1334 & -0.6944 \\ 2.2698 & 0.5555 & -3.7374 \\ 0.6977 & 0.2580 & 0.1710 \end{bmatrix}$$

$$\mathbf{c}_{cm} = [\ 0.1859 \quad 0.0001 \quad 0.8134\ ]$$

$$L_{cm} = 0.9272$$

Dead-times fixed to 0.5[s]

$$\mathbf{A}_{cm} = \begin{bmatrix} 2.0320 & 15.1554 & -17.5541 \\ -0.3231 & -1.1394 & 1.2835 \\ 0.0681 & 1.6673 & -1.9947 \end{bmatrix}$$

$$\mathbf{c}_{cm} = [\ 0.5542 \quad 1.7960 \quad -1.2771\ ]$$

$$L_{cm} = 0.6720$$

Reduced parameter model, Example 3.6

$$\mathbf{A}_{cm} = \begin{bmatrix} -1.0327 & 0 & 0 \\ 0 & -2.5075 & 0 \\ 0 & 0 & 0.0063 \end{bmatrix}$$

$$\mathbf{c}_{cm} = \begin{bmatrix} 0.1785 & -0.2030 & 0.9691 \end{bmatrix}$$

$$L_{proc} = \begin{bmatrix} 0.3808 & 0.3428 & 0.1156 \end{bmatrix}$$

Dynamic compensation module Example 3.7

$$\mathbf{A} = \begin{bmatrix} -33.1359 & -20.5618 \\ 14.3354 & -28.5952 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} -0.2854 & 27.7614 \\ -174.7269 & 77.3113 \end{bmatrix}$$

$$\mathbf{c} = \begin{bmatrix} 1.0277 & 0.5717 \end{bmatrix}$$

$$L_{proc} = 0.0884$$

B.2 MIMO Systems

MIMO Model From Example 1, Section 4.2

This if a four input, four output model. The results for subsystem 1 is

$$\mathbf{A} = \begin{bmatrix} 0 & -4.0190 & 0 & -0.4985 \\ 1 & -4.8825 & 0 & -0.5510 \\ 0 & 0.1483 & 0 & -0.2373 \\ 0 & 0.1357 & 1 & -0.2282 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 1.1513 & -1.5722 & 0.1099 & -0.1254 \\ 0.0604 & -0.0523 & 0.0179 & 0.2396 \end{bmatrix}$$

$$L_{proc} = \begin{bmatrix} 0.0087 & 0.0552 \end{bmatrix}$$

For the second subsystem, we have

$$\mathbf{A} = \begin{bmatrix} 0 & -0.0202 & 0 & -0.3091 \\ 1 & -0.6445 & 0 & 0.0837 \\ 0 & -0.1206 & 0 & -0.2639 \\ 0 & -0.2324 & 1 & -0.4237 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 0.7611 & -0.0873 & -0.0414 & -0.4976 \\ -0.0110 & 0.6142 & 0.8141 & -0.8242 \end{bmatrix}$$

$$L_{proc} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

Parameters of interaction matrices are

$$\mathbf{M}_I = \begin{bmatrix} 1.5287 & -0.0090 & -0.0997 & 0.1311 \\ 0.0221 & 1.5765 & -0.0144 & 0.0132 \\ 0.0132 & 0.2131 & 2.7436 & -0.0187 \\ 0.0046 & -0.0145 & 0.0564 & 2.8715 \end{bmatrix}$$

$$\mathbf{M}_O = \begin{bmatrix} 0.6752 & 0 & 0 & 0 \\ 0 & 0.6341 & 0 & 0 \\ 0 & 0 & 0.3384 & 0 \\ 0 & 0 & 0 & 0.3489 \end{bmatrix}$$

$$\mathbf{A}_{s1} = \begin{bmatrix} -0.0342 & 0.0017 \\ -0.0084 & -0.0023 \end{bmatrix}$$

$$\mathbf{A}_{s2} = \begin{bmatrix} -39 & 38108 \\ 0.9 & -1 \end{bmatrix}$$

$$L_{proc} = \begin{bmatrix} 0.0299 & 0.0547 & 0 & 0 \end{bmatrix}$$

Example 4.2

This is 10-inputs, 10-outputs system. In the Phase 1, three subsystems are determined. For subsystem Ω_1 we have

$$\mathbf{A}_1 = \begin{bmatrix} 0 & -0.0510 & 0 & -0.33834 & 0 & -0.0697 \\ 1 & -0.0226 & 0 & 0.1860 & 0 & 0.5199 \\ 0 & -0.0143 & 0 & -0.0057 & 0 & 0.0503 \\ 0 & -0.0762 & 1 & -0.1067 & 0 & 0.0053 \\ 0 & 0.0716 & 0 & 0.1663 & 0 & -0.0838 \\ 0 & -0.1772 & 0 & 0.0014 & 1 & -0.6503 \end{bmatrix}$$

$$\mathbf{B}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{C}_1 = \begin{bmatrix} 0.4479 & -0.2255 & 0.0720 & 0.0933 & 0.7741 & -0.1079 \\ 0.1551 & -0.1505 & 0.4288 & -0.0989 & 0.6167 & -0.3337 \\ 0.4857 & -0.1448 & 0.0214 & 0.0073 & 0.2259 & 0/2855 \end{bmatrix}$$

$$L_{proc1} = \begin{bmatrix} 0.3860 & 0.8045 & 0.0003 \end{bmatrix} \times 10^{-3}$$

For subsystem Ω_2 we have

$$\mathbf{A}_2 = \begin{bmatrix} 0 & 0.0040 & 0 & -0.0484 & 0 & 0.1678 \\ 1 & -0.0067 & 0 & 0.1771 & 0 & 0.0068 \\ 0 & 0.4049 & 0 & -0.0525 & 0 & 0.5246 \\ 0 & -0.0469 & 1 & 0.0177 & 0 & 0.0903 \\ 0 & -0.0232 & 0 & -0.0133 & 0 & -0.0210 \\ 0 & 0.0006 & 0 & 0.8470 & 1 & 0.0114 \end{bmatrix}$$

$$\mathbf{B}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{C}_2 = \begin{bmatrix} 1.1260 & -0.4664 & 2.2641 & 1.4967 & 1.2654 & -0.0141 \\ 0.3551 & -0.0358 & 0.6924 & 0.3414 & 0.2179 & 0.1339 \\ -0.3381 & -0.0263 & -1.1482 & -0.4073 & -0.4115 & -0.3257 \end{bmatrix}$$

$$L_{proc2} = \begin{pmatrix} 0 & 0 & 1.0134 \end{pmatrix} \times 10^{-3}$$

The third subsystem is

$$\mathbf{A}_3 = \begin{bmatrix} 0 & 0 & 0.0066 & 0 & 0 & 0.0089 & 0 & 0.0000 & 0 & 0.0103 \\ 1 & 0 & 0.0135 & 0 & 0 & -0.0555 & 0 & -0.0327 & 0 & -0.1870 \\ 0 & 1 & 0.0321 & 0 & 0 & -0.5321 & 0 & -0.2096 & 0 & 0.0458 \\ 0 & 0 & -0.0590 & 0 & 0 & 0.0807 & 0 & -0.0023 & 0 & -0.0039 \\ 0 & 0 & 0.0245 & 1 & 0 & -0.3693 & 0 & -0.1362 & 0 & -0.0022 \\ 0 & 0 & 0.0008 & 0 & 1 & 0.04760 & 0 & -0.1773 & 0 & 0.0167 \\ 0 & 0 & 0.0985 & 0 & 0 & 0.1058 & 0 & 0.0027 & 0 & -0.0014 \\ 0 & 0 & 0.2108 & 0 & 0 & 0.1058 & 1 & -0.3462 & 0 & -0.0375 \\ 0 & 0 & -0.0238 & 0 & 0 & 0.1989 & 0 & 0.0033 & 0 & -0.0959 \\ 0 & 0 & 0.0434 & 0 & 0 & -0.0738 & 0 & 0.2270 & 1 & 0.0097 \end{bmatrix}$$

$$\mathbf{B}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{C}_3^T = \begin{bmatrix} 1.0047 & 0.1071 & 0.1387 & -0.0201 \\ -1.9533 & -0.0246 & -0.0466 & -0.1734 \\ 0.07 & 0.0818 & 0.2685 & -0.4872 \\ 1.4982 & 1.0054 & 1.5428 & 0.3136 \\ 0.7052 & 0.2739 & 0.0597 & 1.1427 \\ 0.0002 & -0.3735 & -0.5778 & -0.0419 \\ 0.1429 & 0.1652 & -0.0058 & -0.3 \\ 0.3183 & 0 & 0.2984 & -0.2109 \\ 1.3616 & 0.2777 & 0.1085 & 0.089 \\ -0.4355 & -0.1423 & -0.2961 & 0.299 \end{bmatrix}$$

These subsystems are coupled together, and the resulting matrices from the Phase 2 are as follows:

$$A_{s1} = \begin{bmatrix} 0.1751 & 0.0954 & -0.2660 \\ 0.0402 & -0.1740 & 0.0824 \\ 0.1524 & -0.0670 & 0.0168 \end{bmatrix}$$

$$A_{s2} = \begin{bmatrix} 0.0283 & 0.0308 & 0.6649 \\ -0.0328 & -0.0418 & -0.1822 \\ -0.0415 & 0.1025 & 0.0874 \end{bmatrix} \times 10^{-3}$$

$$A_{s3} = \begin{bmatrix} -0.0001 & 0.0203 & -0.0057 & -0.0013 \\ -0.0064 & 0.0239 & 0.0051 & 0.0004 \\ -0.1511 & -0.3226 & 0.3063 & 0.3975 \\ -0.0250 & -0.0361 & -0.0105 & -0.1356 \end{bmatrix}$$

$$\mathbf{M}_I = \begin{bmatrix} 0.9552 & -0.0448 & -1.4731 & 0.0674 & 1.6174 \\ -0.0082 & 1.3045 & -0.0932 & -0.2525 & -0.1615 \\ -0.1616 & 0.0358 & 2.1159 & -0.0299 & -0.0003 \\ 0.0000 & 0.0001 & 0.0000 & 0.9999 & -0.0001 \\ 0.0000 & 0.0000 & 0.0000 & 0.0001 & 1.0000 \\ 0.0000 & 0.0000 & 0.0000 & -0.0001 & -0.0001 \\ -0.0057 & -0.0301 & 0.0066 & -0.0013 & 0.0341 \\ 0.0016 & 0.0076 & 0.0188 & -0.0051 & 0.0218 \\ 0.1129 & -0.0921 & 1.4875 & 0.1001 & 1.8344 \\ 0.0376 & 0.0650 & 0.3199 & 0.0090 & 0.4062 \end{bmatrix}$$

$$\begin{bmatrix} 0.2525 & -0.0645 & 1.1961 & -0.0544 & 0.4101 \\ -01863 & 0.4832 & -0.5969 & 0.0686 & 0.7678 \\ 0.0003 & -0.0236 & -1.9767 & 0.0345 & -0.2474 \\ -0.0001 & 0.0002 & 0.0004 & 0.0005 & -0.0004 \\ -0.0001 & 0.0001 & -0.0001 & 0.0002 & 0.0003 \\ 1.0000 & -0.0001 & 0.0002 & 0.0001 & -0.0005 \\ -0.0063 & 1.0596 & 0.0084 & 0.0843 & -0.0349 \\ -0.0023 & -0.0083 & 0.9123 & 0.0240 & -0.0486 \\ 0.0340 & 0.0354 & -3.7631 & 0.4798 & 0.3834 \\ 0.0202 & 0.0445 & -0.8581 & -0.0917 & 0.1427 \end{bmatrix}$$

$$L_{proc} = [\ 0.0013 \quad 0.0510 \quad 0 \quad 0 \quad 0.0001 \quad 1.0049 \quad 0 \quad 0.0024 \quad 0 \quad 0.0587\ ]$$

$$\mathbf{M}_O$$

$$= \ diag[0.9387, 0.9762, 0.9514, 1.0002, 0.9999, 1.0005, 0.9025, 0.9528, 0.9564, 0.7214].$$

Appendix C

Program Descriptions

In this Appendix, programs used in identification are presented. As the number of programs is very large, it is inappropriate to list all of them. Many of these programs are similar to each other; and some of them are often slightly modified to accommodate similar identification approaches. In order to illustrate all important aspects of these programs, they will be presented here in algorithmic or pseudo-language forms. Although the original programs are written as Matlab M-files, the main reason for presenting the programs in pseudo-language is to illustrate the main concepts, enabling easy implementation on any suitable platform.

The programs come into two main groups: those used for identification of SISO models (including both identification of transfer function models and for hierarchical identification by composite models) and those for identification of MIMO models, again one-shot and hierarchical approach.

C.0.1 Programs For Identification of SISO Systems

Presented here are pseudo-programs for identification of SISO systems. First, pseudo-programs for identification of transfer function are presented and then pseudo-programs for hierarchical identification.

Identification of Transfer Function

FIT_TF

```
! Matrix containing input-output sequences is formed outside the program
! Data = [Tin Yout Uin]
! where Tin is the time sequence, Yout measured output and Uin measured input.
! As transfer function is meaningful only for initially relaxed systems;
! input and output signals must be brought to the same level before the
! moment in which input signal is applied.

read (pole_position);  ! Parameter to place the closest pole to the origin
read (numerator);      ! enter the initial values for numerator and denominator
read (denominator);    ! of the transfer function
read (Lproc);          ! Initial value for system dead-time
lam = [numerator denominator Lproc]; ! Forming of parameter vector

! At this point optimization method and error function to be minimized are chosen
lambda=optimize(error_function, lam,pole_position); ! lam is the initial
! value for optimization
! lambda is parameter vector containing the values for which error_function
! is minimized
```

```
function = ERROR_FUNCTION(lambda,pole_position)

! This is an error function to be minimized. It represents the model output error
! and is given in the form of ISE

! extract numerator and denominator from lambda
extract(numerator,lambda);
extract(denominator,lambda);
roots = roots(denominator);
compare(roots,pole_position); ! See if there are any roots to the right from pole_position
modify(denominator); ! set those to the right to the value of pole_position,
! and form new denominator.
ymod = integrate(Tin,Uin,numerator,denominator,Lproc); ! For given numerator,
! denominator and dead-time, calculate the model output
! Form an error that will be minimized
error_function = ISE(Yout, ymod);
```

Programs for Hierarchical Identification of SISO Systems

fit_vms

```
! Input-output matrix is formed in the same way as in the case of fit_tf.
read(number_of_subsystems);
read(lam);        ! Read the initial values for parameters
lambda = optimize(error_function,lam);

function error_function(lambda)

zero(A);     ! initialize matrix A to zero matrix
for i = 1 to dimension(A)      ! set matrix A to its initial values
     for j = 1 to dimension(A)
          A(i,j) = lambda((i-1)*dimension(A)+j);
     end for
end for
for i = 1 to dimension(A)      ! Initialize vector c
     c(i) = lambda(dimension(A)^2 + i);
end for
Lproc = lambda(dimension(A)^2+1:dimension(A)^2+dimension(A)); ! dead-time
Lcm = lambda(dimension(A)^2+dimension(A)+1);
ymod = integrate (composite-model, A, c, Lproc); ! calculate the composite
! model output and then form ISE between model and the measured output
error_function = ISE(Yout,ymod);
```

C.0.2 Programs for Identification of MIMO Systems

In what follows, programs for identification of MIMO systems by one-shot identification in canonical form and for hierarchical identification are sketched.

Identification in Canonical Forms ; The input-output matrix is created outside the program and is of the form

```
; Data = [Tin u1 u2 . . . up y1 y2 . . . yq]
read(p);                    ! number of inputs
read(q);                    ! number of outputs
read(model_order);
read(struct_indices);       ! enter the set of structural indices
! based on this, the program determines the structure of system matrices
[non_zero_indices,one_indices,nzi,ozi]=canon_A(p,order,struct_indices);
! The positions of fixed elements of system matrix A in canonical form
! are returned by canon_A
lamA = read(non-zero elements of matrix A);
non_zero_B_indices = canon_B(p, order, struct_indices); ! Elements of
! the matrix B are zeros and ones in fixed positions
lamC = read(C);     ! elements of matrix C are free
zero(D);     ! matrix D is usually taken as zero matrix
! Parameter vector is formed in the following way
lam = [lamA lamC dead-times];
optimize(error_function,lam);
```

```
function error_function(lambda)
! As parameters are passed as vector, it is necessary to convert this vector
! into system matrices in the canonical form determined by the set of structural
! indices.Various subroutines are used for this purpose.
A = zero(dimension(A));
for i = 1 to ozi (number of fixed ones)
    [a,b] = unwrap_i(one_indices(i));
    A(a,b) = 1;
end for
for i = 1 to nzi (number of non-fixed parameters in A)
    [a,b] = unwrap_i(non_zero_indices(i));
    A(a,b) = lambda(i);
end for

B = zero(order x p);
[bzo bzi] = size(non_zero_B_indices)
for i = 1 to bzi
    [a,b] = unwrap_i(non_zero_B_indices(i));
    B(a,b) = 1;
end for

C = zero(q x order);
counter = nzi + 1;
for i = 1 to q
    C(i,:) = lambda(counter : counter + order - 1);
    counter = counter + 1;
end for
```

```
! After matrices A, B and C are formed, simulation can be done
! yielding q output sequences:
[ymod1 ymod2 . . . ymodq] = integrate(model,A,B,C,inputs);
error_function = 0;      ! error is formed as the sum of ISE for each of the outputs
for i = 1 to q
    error_function = error_function + ISE(Youti - ymodi);
end for
```

Programs for Hierarchical Identification

FITFMIMO

```
! Enter the initial values for matrices FWMI (input distribution matrix),
! FWMO (output distribution matrix and local output feedback matrices FBM
read(FWMI);
vector_FWMI = pack_mat(FWMI); ! pack elements of a matrix into a vector
read(FWMO);
vector_FWMO = pack_mat(FWMO);
read(FBM);
vector_FBM = pack_mat(FBM); ! This has to be done for each of the subsystems
read(dead-times) ! initial values for the dead-times
lam = [vector_FWMI vector_FWMO vector_FBM dead-times]; ! Initial values
lambda = optimize(error_function,lam);
```

```
function ERROR_FUNCTION(lambda)

! in order to calculate the model output, all matrices must be recreated from
! the parameter vector lambda
FWMI = unpack_m(lambda, sizes); ! portion of the parameter vector is
! converted into a matrix according to the given dimensions (sizes)
FWMO = unpack_m(lambda, sizes);
FBM = unpack_m(lambda, sizes);
[ymod1 ymod2 . . . ymodq] = integrate(model,FWMI,FWMO,FBM,inputs);
error_function = 0;      ! error is formed as the sum of ISE for each of the outputs
```

```
for i = 1 to q
    error_function = error_function + ISE(Youti - ymodi);
end for
```

Some Auxiliary Functions

```
function vector = pack_mat(M) ! pack the given matrix M into a vector
[m,n] = size(M); ! M is m x n matrix
vector = zero_vector(1 x m*n);
index = -;
for i = 1 to m
    a(index+1:index+n) = M(i,:);
    index = index + n;
end for


function M = unpack_m(a,m,n) ! unpack vector a into m × n matrix M
na = size(a);
if na ~= m*n then
    error;
else
    M=zero(m x n);
    index = 0;
    for i = 1 to m
        M(i,:) = a(index+1:index+n);
        index = index + n;
    end for
end if


function a = wrap_i(n,order)
```

```
! indexes all elements on n-th column in a square matrix of order n
a = zero_vector(order);
for i = 1 to order
    if (n<10) and (i < 10) then
        a(i) = 10*i + n;
    else
        a(i) = 100*i + n;
    end if
end for


function a = wrap_1(m,n)
! this function translates the position index [m,n] into an integer a
if (n < 10) and (m < 10)
    a = 10*m + n;
else
    a = 100*m + n;
end
```

```
function [a,b,c,d] = canon_A(p,order,struct_indices)

! this function returns system matrix A in canonical form
! vector a contains the positions of free parameters and
! vector b contains the positions of fixed ones. All other elements
! of matrix A are zeros.
nzc(1) = nzc(1) + struct_indices(1);
for i = 2 to p      ! nzc contains indices of the columns containing
    nzc(i) = nzc(i-1) + struct_indices(i);      ! free parameters
end for
a = zeros(1 x order*p);
counter = 1;
for i = 1 to p
    a(counter:counter + order -1) = wrap_i(nzc(i),order);
    ! indices are packed into integers
    counter = counter + order;
end for
c = size(a);

! we now determine the positions of fixed ones
submatrix = zero_vector(p);
for i = 1 to p
    submatrix(i) = nzc(i) - struct_indices(i) +1;
end for
bpom = zero_vector(order-1);
```

```
bcounter = 1;
for i = 1 to p
    for j = 1 to struct_indices(i) - 1
        pom = submatrix(i) + j;
        bpom(counter) = wrap_1(pom,pom-1);
        bcounter = bcounter + 1;
    end for
end for
count = 1;
for i = 1 to order-1
    if bpom(i) ~= 0 then
        b(count) = bpom(i);
        count = count + 1;
    end if
end for
d = size(b);


function a = canon_B(p,order,struct_indices);

! this function returns positions of fixed ones in matrix B in canonical form
pos(1) = 1;
a(1) = [11];
for i = 2 to p
    pos(i) = pos(i-1) + struct_indices(i-1);
    a(i) = wrap_1(pos(i),i);
end
```