30
480
8-20-79

DR. 3017

CONS - 2858-T1

# A DESCRIPTOR-VARIABLE APPROACH TO MODELING AND OPTIMIZATION OF LARGE-SCALE SYSTEMS

FINAL REPORT

MARCH 1976 - FEBRUARY 1979


DONALD N. STENGEL

DAVID G. LUENBERGER

ROBERT E. LARSON

TERRY B. CLINE





SYSTEMS CONTROL, INC.

1801 PAGE MILL ROAD

PALO ALTO, CA. 94304



DATE PREPARED - FEBRUARY 1979





PREPARED FOR THE

DEPARTMENT OF ENERGY

OFFICE OF THE ASSISTANT ADMINISTRATOR FOR ENERGY TECHNOLOGY

DIVISION OF ELECTRIC ENERGY SYSTEMS

WORK PERFORMED UNDER CONTRACT NOS. EX-76-C-01-2090
ET-78-C-01-2858




JB

## DISCLAIMER

Available from:

National Technical Information Service (NTIS)
U.S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22161

Price: Printed copy: $5.25
Microfiche: $3.00

# A DESCRIPTOR-VARIABLE APPROACH TO MODELING AND OPTIMIZATION OF LARGE-SCALE SYSTEMS

## ABSTRACT

A new approach to modeling and analysis of systems is presented that exploits the underlying structure of the system. The development of the approach focuses on a new modeling form, called *descriptor variable* systems, that was first introduced in this research. Key concepts concerning the classification and solution of descriptor variable systems are identified, and theories are presented for the linear case, the time-invariant linear case, and the nonlinear case. Several standard systems notions are demonstrated to have interesting interpretations when analyzed via descriptor variable theory.

The approach developed also focuses on the optimization of large-scale systems. Descriptor variable models are convenient representations of subsystems in an interconnected network, and optimization of these models via dynamic programming is described. A general procedure for the optimization of large-scale systems, called *spatial dynamic programming*, is presented where the optimization is spatially decomposed in the way standard dynamic programming temporally decomposes the optimization of dynamical systems. Applications of this approach to large-scale economic markets and power systems are discussed.

February 1979

USDOE Contract Nos.

EX-76-C-01-2090
ET-78-C-01-2858

## PROJECT STAFF

Robert E. Larson
*(Principal Investigator)*

David G. Luenberger
*(Co-Principal Investigator)*

Donald N. Stengel

Alexander H. Levis

Elizabeth R. Ducot

C.-Y. Chong

Joan W. Cummins *(Sec.)*

Terry B. Cline

Paul L. McEntire

S. Harold Javid

Kent D. Wall

R. Louise Schuler *(Sec.)*

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

# I. INTRODUCTION AND OVERVIEW

## 1.1 PROJECT THEMES

This report contains the results of all research conducted in the project *A Descriptor Variable Approach to Modeling and Optimization of Large Scale Systems*. The project is part of an effort to investigate modeling forms that are appropriate for the analysis of large-scale systems. The guiding philosophy of this research was to develop mathematical system formulations that exploit the special structure of the actual system being modeled. It is believed that a model is most likely to preserve the natural structure if the system representation is given in terms of actual physical or economic variables that describe the system operation. Since the systems studied in this report accommodate such representations, the formulations have been called *descriptor variable* systems.

The research effort of this project had two major themes:

(1) Development of the foundation of descriptor variable theory, including identification of basic concepts and derivation of fundamental analysis techniques.

(2) Investigation of large-scale system optimization, including the development of spatial dynamic programming, a technique that exploits subsystem interconnection structure.

These themes were initially explored somewhat independently, however both themes shared the common philosophy of preserving structure. Therefore, it was not surprising that the themes interacted during this research. This interaction is reflected in sections of this report, including a synthesis of the themes in Chapter VIII. The remainder of this chapter gives an overview of the content of the report.

## 1.2 DESCRIPTOR VARIABLE THEORY

A discrete-time system in descriptor form is a set of (vector) relationships:

$$f_0(x(0),\ x(1),\ u(0)) = 0$$

$$f_1(x(1),\ x(2),\ u(1)) = 0$$

$$\vdots$$

$$f_{N-1}(x(N-1), x(N), u(N-1)) = 0$$

where $x(k)$ is a vector of descriptor variables and $u(k)$ is a vector of inputs associated with increment $k$. The descriptor representation accommodates a wide range of system formulations, including the special class of state-space representations, where the relationships above take the form

$$x(k+1) = h_k(x(k),\ u(k))$$

Many issues examined in the context of state-space systems, such as controllability, optimal control, and system inversion are handled naturally in the descriptor variable framework. The relevance of this framework to systems theory is considered at the end of Chapter II.

A central concern in any descriptor system is the nature of the solution space for the descriptor variables corresponding to some specified trajectory of inputs. Two dual concepts, *solvability* and *conditionability*, characterize a well-defined solution space. Solvability requires that all the descriptor system equations are independent, such that no relationships are redundant or potentially contradictory. Conditionability indicates that a unique solution can be obtained by properly specifying conditions on the initial descriptor vector $x(0)$ and the final descriptor vector $x(N)$. Solvable and conditionable descriptor systems can be solved using a *double-sweep* method. Essentially the solution is determined by propagating the specified conditions on $x(0)$ forward through the time stages and the conditions on $x(N)$ backward through the time stages. The forward and backward propagation at any stage $k$ uniquely characterizes $x(k)$. This double-sweep method is, in fact, a generalization of the single-sweep solution for state-space system with given initial conditions. However, a special class of descriptor systems, called *regular* systems can be solved using only the forward sweep. These concepts are examined in the context of linear descriptor systems in Chapter II.

Time-invariant linear descriptor systems have several special properties, which are discussed in Chapter III. First, any such system is solvable for any number of stages $N$ if and only if it is conditionable for any $N$. Second, there is an equivalence between a time-invariant linear system and a matrix pencil of the form $sE-A$; the system is solvable if and only if the determinant of the matrix pencil is nonzero for some scalar $s$. Third, any system in this class can be expressed in a quasi-state-space form:

$$x(k+1) = \bar{A}x(k) + \bar{B}u(k) + \bar{C}u(k+1) + \ldots + \bar{L}u(k+n-1)$$

Finally, using the equivalence between time-invariant linear systems and their corresponding matrix pencils, these systems can be decomposed into a maximal number of independent subsystems, with each subsystem being a state-space system that propagates forward in time or a state-space system that propagates backward in time. This decomposition is useful for system analysis or characterizing valid initial and final conditions.

The concepts developed in the context of linear descriptor systems extend to nonlinear systems. The nonlinear theory is constructed from the viewpoint of differential topology, allowing both geometric and algebraic interpretations. A manifold in the space of all possible realizations of descriptor vectors characterizes the set of solutions. In a solvable and conditionable system, this manifold can be projected in a one-to-one manner to a manifold in the space of x(0) and x(N) vectors, thus creating the manifold of arbitrary boundary conditions. The selection of a point on this boundary manifold, accompanied by the creation of equivalent boundary manifolds corresponding to shorter time intervals, form the basis for a nonlinear double-sweep of calculating the solution. This approach is described in Chapter IV.

## 1.3 LARGE-SCALE SYSTEM OPTIMIZATION

Large-scale system models can often be expressed in descriptor form, and therefore methods for optimizing descriptor variable system are important tools for large-scale optimization. Descriptor variable systems with separable performance criteria can be optimized using dynamic programming. However, the principle of optimality must be reformulated from the principle developed for state-space systems, since any optimal trajectory must be consistent with the final as well as the initial conditions. A more efficient dynamic programming procedure exists for regular descriptor systems, where a lower order state exists that is not constrained by conditions on x(N). The optimal input trajectory for regular descriptor systems can be always expressed as a feedback control law. These dynamic programming algorithms for linear descriptor systems are considered in Chapter V.

One approach to optimizing large-scale systems is a two-level approach where at one level the interactions between subsystems are selected, and at the other level the subsystems are optimized for the given interactions. The combination of a subsystem model with relationships characterizing the fixed interactions often creates a rectangular descriptor system. Since rectangular systems have more equations than unknowns, these systems are generally solvable for only a restricted class of input trajectories. The optimal control policy corresponding to these subsystem models can be expressed as a feedback control law when the system is regular and satisfies a condition called *uniform complete maintainability*. The description of this condition and its application are contained in Chapter V.

The two-level approach indicated above provides the focus for a comprehensive procedure for large-scale system optimization, called *spatial dynamic programming*. The underlying motivation is that by performing a sequence of smaller optimization problems, the overall optimal solution can be determined in a tractable manner. The theme of spatial dynamic programming is to optimize across interconnected subsystems in the same way that classical discrete dynamic programming optimizes a dynamic systems across time stages. First, one subsystem is optimized for every combination of interconnections, creating a family of solutions that are each parameterized on the interconnections between that subsystem and the rest of the system. Next, another subsystem is added and a family of optimal solutions for the combination of the two subsystems is determined, with each solution parameterized on the interconnections between this composite and the rest of the system. This second iteration of optimizations is eased by the results of the first iteration. The process of adding an additional subsystem and optimizing the new composite, while embedding the results of the previous iterations, is continued until all subsystems have been included in the composite. This procedure leads to a global optimum in a finite number of iterations and can be applied to a wide range of systems and optimization criteria. A skillful sequencing of the subsystem optimizations can efficiently exploit the interconnection structure of system. Further description of spatial dynamic programming and demonstration of its global optimality appear in Chapter VI.

## 1.4 APPLICATIONS

The theoretical results of this study have been applied to two classes of large-scale systems: economic networks and power systems. Economic networks consist of a set of interconnected markets linked by the sectors of economy that participate in those markets. By properly aggregating these sectors into sets, a price-quantity equilibrium model describing the sectors' behavior with respect to these markets can be formulated as a descriptor variable system. For certain classes of networks, the double-sweep solution procedure corresponds to the construction of supply and demand functions at each market, thereby providing substantial insights into the operation of the network in addition to an efficient solution procedure. The formulation as a descriptor model also permits the application of theoretical results derived for descriptor systems such as the effect of particular policy inputs on the network equilibrium and the optimization of policy inputs. This application area is discussed in Chapter VII.

Power networks can be modeled as a large-scale system of interconnected generation nodes and load points. An ongoing concern to power system operators is the determination of a generation mix that minimizes the costs of generation, operation and maintenance, etc. Spatial dynamic programming readily accommodates such systems and cost criteria. This technique does not require simplified linear models of the power flows and easily handles dispersed generation of power; yet it is capable of minimizing costs for large, interconnected networks in a computationally feasible manner. The formulation of the optimal power flow problem and the application of spatial dynamic programming to a power network example appears in the final section of Chapter VI.

## II. CONCEPTS OF DESCRIPTOR VARIABLE SYSTEMS

### 2.1 INTRODUCTION

Dynamic phenomena represent a special version of complexity - where the variables describing a system at one time are interrelated, not only with other variables at that time, but in a special way with variables at other times. A general formulation of a set of dynamic relations is provided by a set of equations of the form

$$g_0(x(0),x(1),u(0))=0$$

$$g_1(x(1),x(2),u(1))=0$$

$$\vdots \qquad (2.1)$$

$$g_{n-1}(x(N-1),x(N),u(N-1))=0$$

where

- $x(k)$ is an n-dimensional descriptor vector for each $k=0,1,2,\ldots,N$
- $u(k)$ is an m-dimensional input vector for each $k=0,1,\ldots N-1$
- $g_k$ is a function taking values in n-dimensional space.

Throughout this report, a set of equations of this form is referred to as a set of *dynamic equations in descriptor form.*

A special case is represented by the set of linear equations

$$E_{k+1}x(k+1)=A_kx(k)+u(k), \qquad k=0,1,\ldots,N-1 \qquad (2.2)$$

Each $A_k$ and $E_k$ is an nXn matrix. Again each $x(k)$ is an n-dimensional descriptor vector and now each $u(k)$ is an n-dimensional input vector. (In many situations the actual input enters the equations with some coefficient matrix, say $B_k$, but from the present viewpoint this serves only to redefine the input vector.)

The descriptor formulations (2.1) and (2.2) above contain as special cases many standard forms. Some of these, of course, are most conveniently treated directly with standard techniques [ 1]-[ 3], without need for the more general representation. There are, however, several important classes of situations in which the descriptor representation is either a natural starting point or an essential characteristic. Some ways in which such representations arise are outlined below. Other applications are considered in Section 2.6.

*The Descriptor Approach to Modeling*

Typically, the process of modeling a complex situation is initiated by the definition of a collection of variables that, in some sense, is adequate to describe the system. These are conveniently referred to as *descriptor variables*. The descriptor variables generally have inherent meaning, or natural interpretations, within the context of the particular situation. They might represent, for example, positions, velocities, or accelerations in Newtonian systems, prices or quantities in an economic system, etc. Initially, no attempt is made to select a minimal set of variables; the objective being simply to obtain an adequate set. Once the variables are defined, relations among the descriptor variables are developed as dictated by the system laws. Some of the resulting relations will, in general, be dynamic, in that they involve variables at different time instants, and some of the relations will be purely static, representing identity relations that hold between variables. The result of this process of modeling is a set of equations expressed in terms of variables that are natural descriptors of the system. This approach to the modeling of physical systems is emphasized in several standard texts (e.g., [ 4 ]).

*Large-Scale Interconnected Systems*

Often a large-scale system is most effectively regarded as a collection of interconnected subsystems [ 5]. Each subsystem i may have a representation of the form

$$x_i(k+1)=A_ix_i(k)+B_iv_i(k)$$

$$z_i(k)=C_ix_i(k)+D_iv_i(k)$$

where $x_i(k)$, $v_i(k)$, and $z_i(k)$ are, respectively, state, input, and output vectors. These can be combined, in the obvious way, to produce the overall subsystem equations

$$x(k+1)=Ax(k)+Bv(k)$$

$$z(k)=Cx(k)+Dv(k).$$

The interconnections between subsystems and the overall input $u(k)$ and overall output $y(k)$ might be defined by linear relations of the general form

$$v(k)=Kz(k)+Mu(k)+Ny(k).$$

$$y(k)=Pz(k)+Qu(k)+Rv(k).$$

The resulting interconnected system cannot readily be transformed to state vector form. Indeed, it

is known that a state vector representation may not exist [ 6 ]. Nevertheless, the complete set of equations is easily seen to be a special case of (2.2), with descriptor vector equal to (x(k),v(k),z(k),y(k)), and therefore they can always be treated by the methods of this report.

*Nondiagonal E Matrices*

The equations of many large-scale systems (e.g., electric power systems - see [7 ] or [8 ]) or systems of equations representing partial differential equations often have natural representations of the form (2.2) where the corresponding matrices $E_k$ are nonsingular but not diagonal. Sometimes these matrices have simple structure, such as a tridiagonal form, and it may be convenient to maintain the simplicity of this structure in the equations. In such cases one would work with the form (2.2) rather than, or in conjunction with, the more standard form obtained by multiplying by $E_k^{-1}$.

*Perturbation Equations*

A very powerful method for dealing with large systems is that of perturbation analysis where small constants in a system are set to zero to produce a simplified system which serves as basis for an initial control design [9 ]. The singular perturbation method works with systems of the form

$$x_1(k+1)=A_{11}x_1(k)+A_{12}x_2(k)+B_1u(k)$$

$$\varepsilon x_2(k+1)=A_{21}x_1(k)+A_{22}x_2(k)+B_2u(k)$$

where the perturbation parameter $\varepsilon$ is small. The case $\varepsilon=0$ corresponds to a singular perturbation. It changes the dynamic order and leads to a set of equations of type (2.2).

*Noncausal Systems*

A system expressed (or expressible) in state-space form is causal, in that its state vector is not influenced by future inputs. For some purposes, however, causality is a limitation. For instance, some important linear data processing schemes are noncausal but can be represented by the general descriptor representation (2.2).

*Identification Problems*

An important branch of dynamics, encompassing much of the disciplines of standard system theory, econometrics, and various social sciences, is that of identification - where parameters of a dynamic representation are fitted to data. In difficult situations, the structure postulated for identification must be sufficiently general to allow for an uncertainty in the underlying dynamic order, or even for an uncertainty of the causality pattern (i.e., which variables depend on previous values of which others). The structure of general descriptor equations (2.2) is sufficiently rich for these purposes, while more conventional forms often are not.

The exposition of this chapter is restricted to the case of linear equations, although much of the development will be extended (at least in principle) to the nonlinear case in Chapter IV. In the development use is made of an old concept in dynamic systems; that of initial conditions. An initial condition vector can be propagated through the system, much like a state, even when a state vector does not exist. Indeed the solution to most dynamic equations can be obtained by propagating initial conditions forward to the final time point and then solving backward. This is referred to as the "double sweep" method of solution and is one of the main results in Section 2.3.

One aim of the chapter is to determine conditions under which a set of dynamic equations can be decomposed into dynamic and static components. This amounts to determining conditions under which there is a state space dynamic system buried somewhere within the original dynamic equations. The existence of a state implies that the solutions to the original equations can be determined recursively as the successive u(k)'s are specified. Equations that fulfill these requirements are referred to as *regular* dynamic equations, and are considered in Section 2.4.

In a completely initialized state-space system, a state variable will be sensitive to previous inputs, but not present or future inputs. These sensitivities are readily determined from the state-space formulation. In the more general descriptor formulation descriptor variables can be sensitive to previous and future inputs. However, descriptor systems decompose into a state-space system plus an independent system that behaves as a state-space system moving backward through time. This decomposition allows a convenient determination of variable sensitivities, as described in Section 2.5.

The results in this chapter concerning descriptor variable systems assume the existence of well-defined solution sets. The notion of a well-defined solution set is characterized by two dual concepts, solvability and conditionability. These two concepts and the dual nature are described in the next section.

## 2.2 SOLVABILITY, CONDITIONABILITY, AND DUALITY

In the general linear case (to which the formal development is restricted) dynamic equations have the form

$$E_{k+1}x(k+1)=A_kx(k)+u(k), \qquad k=0,1,\ldots,N-1 \tag{2.2}$$

where, as before, each $A_k$ and $E_k$ is an nXn matrix, each x(k) is an n-dimensional descriptor vector, and each u(k) is an n-dimensional input vector. This set can be written out in block matrix form as

$$\begin{bmatrix} -A_0 & E_1 & & & \\ 0 & -A_1 & E_2 & & \\ & \ddots & \ddots & & \\ & & & E_{N-1} & 0 \\ & & & 0 \; -A_{N-1} & E_N \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \\ x(N) \end{bmatrix} = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{bmatrix} \qquad (2.3)$$

The block matrix form, with each block being nXn, explicitly displays the fact that the set of dynamic equations can be regarded as one (large) system of linear equations.

*Solvability*

In (2.3) there are N+1 unknown x(k) vectors (each of which is n-dimensional), but there are only N matrix equations (each of which n-dimensional). There is, therefore, an excess of one vector unknown over equations - or in terms of scalar quantities, an excess of n unknowns to equations. Under standard nondegeneracy conditions, one expects accordingly that the system (2.2) will possess not one but a family of n linearly independent solutions. This is formalized by the notion of solvability introduced below.

For convenience, denote the coefficient matrix of the system (2.3) by F(0,N). It can be regarded as an N X (N+1) block matrix, or in ordinary terms as an nN X n(N+1) matrix.

*Definition:* A set of dynamic equations (2.3) is said to be *solvable* if its coefficient matrix F(0,N) is of full rank.

In considering the set of dynamic equations (2.2) it is often convenient to consider a subset obtained by deleting some of the first (or sometimes the last) equations. (This deleting is done, of course, by dropping whole groups of n equations from the block form, not individual equations from the detailed form.) This corresponds to a restriction to a subinterval of the full time interval over which the original set of equations is defined. If a number of equations are deleted in this way so that the first unknown descriptor vector is $x(k_0)$ and the last is $x(k_1)$, the associated coefficient matrix of the reduced set of equations is $F(k_0,k_1)$. It is clear that if the equations of the original set were linearly independent the equations of this reduced set will be also. This shows that solvability is preserved under the fundamental device of time restriction, i.e., *solvability of the whole implies solvability of a subset.*

*Conditionability*

As shown above, a set of dynamic equations always has more unknowns than equations, and therefore, if a solution exists, it will not be unique. Additional relations, or conditions, must be specified to define a unique solution. There is usually great flexibility available for this specification. These additional relations might, for example, specify fixed values for various descriptor variables at certain values of k, or they might specify values for various linear combinations of descriptor variables at various values of k.

In the study of dynamic equations, it is most natural to define the required additional conditions in terms of the descriptor variables at the end points of the given time interval. However, for some sets of dynamic equations a unique solution can be specified only by imposing additional requirements on descriptor variables at intermediate time periods. Such equations are in a certain sense dynamically degenerate since they contain variables which are not influenced by conditions at either end. A criteria for assuring that this does not occur is made formal by the definition of conditionability given below.

Corresponding to the set of equations (2.3), denote by G(0,N) the matrix (expressed in block form)

$$G(0,N) = \begin{bmatrix} E_1 & & & & \\ -A_1 & E_2 & & & \\ & -A_2 & \ddots & & \\ & & \ddots & & \\ & & & E_{N-1} \\ & & & -A_{N-1} \end{bmatrix}$$

The matrix G(0,N) is the submatrix of F(0,N), obtained by eliminating the first n and the last n columns. It is referred to as the *condition matrix.*

*Definition:* A set of dynamic equations (2.3) is said to be *conditionable* if the matrix G(0,N) is of full rank.

It can be seen that conditionability is equivalent to the property that any two distinct solutions to the set of equations (2.3) must differ in at least one end-point descriptor variable. Suppose to the contrary that x(0),x(1),...,x(N) and x(0),x(1),...x(N-1),x(N) are both solutions. Then the difference of these solutions, which is zero in the first and last descriptor vectors, must satisfy the homogeneous equation corresponding to (2.3). However, when the first and last vectors are excluded, the coefficient matrix

of this homogeneous system is G(0,N). There will be solutions to this restricted homogeneous equation if and only if that matrix is of less than full rank.

Conditionability is also equivalent to the property that conditions in terms of end-points are sufficient to uniquely specify a solution. This interpretation, which of course motivates the terminology, is discussed in Section 2.3.

Conditionability, just as is solvability, is preserved under the formation of subsets of dynamic equations. That is, *conditionability of the whole implies conditionability of a subset.*

*Duality*

Solvability and conditionability are in a very natural sense dual concepts. Corresponding to a set of equations (2.2), we define the *subdual* set of dynamic equations by

$$E_k^T\lambda(k-1)=A_k^T\lambda(k)+v(k), \qquad k=1,2,\ldots,N-1.$$

This set of dynamic equations must be interpreted as having time running backward, since each equation involves $\lambda(k),\lambda(k-1)$, and $v(k)$. It should be noted that the subdual set is smaller than the primal, since it has descriptor vectors $\lambda(k)$ only for $k=0,1,\ldots,N-1$. This is necessary because the original set contains only N-1 pairs of $E_k,A_k$ matrices with the same time indices. It is for this reason that the term subdual rather than dual is employed.

In terms of this definition one may easily state the following two duality results.

*Theorem 2.1:* A set of dynamic equations is conditionable if and only if its subdual is solvable.

*Proof:* The coefficient matrix of the subdual set of equations is $G(0,N)^T$. It will be of full rank if and only if its transpose is of full rank. ■

*Theorem 2.2:* If a set of dynamic equations is solvable then its subdual is conditionable.

*Proof:* Solvability of the original set is equivalent to the statement the coefficient matrix F(0,N) is of full rank. It follows that the submatrix F(1,N-1) is also of full rank. The transpose of this submatrix, however, is the condition matrix of the subdual set of equations. ■

*Time-Invariance*

According to the basic definition, a set of dynamic equations is defined with respect to a specific time interval of finite length. Systems of infinite duration are considered to be solvable or conditionable only if the corresponding finite sets of equations, terminating at a fixed N, are solvable or conditionable, respectively, for every value of N.

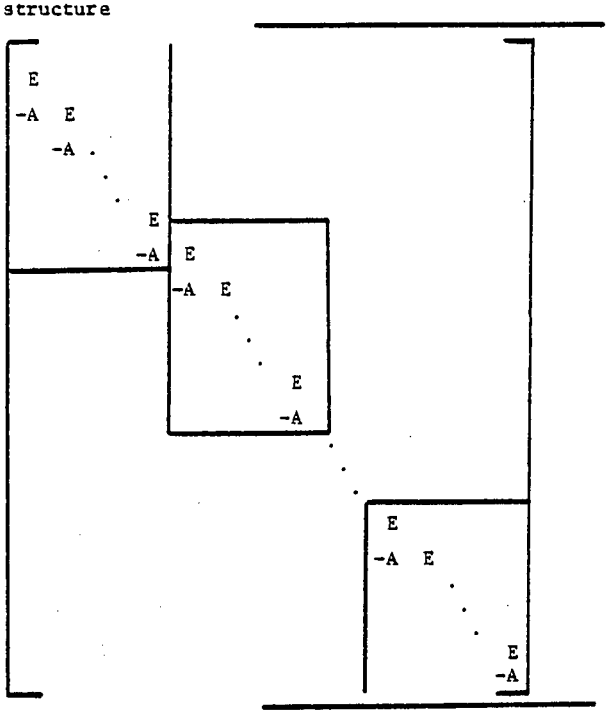
The notion of infinite duration dynamic equations is especially valuable in connection with equations defined by a given pair of E and A matrices, and corresponding to

$$Ex(k+1)=Ax(k)+u(k), \qquad k=0,1,2,\ldots . \quad (2.4)$$

Then, since this infinite duration process is solvable or conditionable only if the corresponding rank conditions hold for all N, it follows that these properties depend only on the pair of matrices E and A. The following result shows that in this case the two properties are not only linked by duality, but they are identical.

*Theorem 2.3:* A time invariant set of dynamic equations is solvable if and only if it is conditionable.

*Proof:* The proof is by contradiction. Suppose that there is an N such that the corresponding finite set of dynamic equations is not conditionable. This means, equivalently, that the matrix G(0,N) is not of full rank. Its rank is, in fact, no greater than n(N-1)-1, since the matrix contains n(N-1) columns. Now let $\bar{N}=2n(N-1)+1$. The larger matrix $G(0,\bar{N})$, corresponding to the set of equations terminating at time $\bar{N}$, will have the structure




It has a total of $n(\bar{N}-1)=2n^2(N-1)$ columns and consists of 2n blocks of the same basic structure that made up the original G(0,N). Accordingly, the rank of this matrix can be no greater than 2n[n(N-1)-1].

The corresponding coefficient matrix $F(0,\bar{N})$ is identical with G(0,N) except that it has 2n additional columns. Thus, the maximum rank that this coefficient matrix can have is $2n[n(N-1)-1]+2n=2n^2(N-1)$. However, to be of full rank this

matrix must be of rank $nN=2n^2(N-1)+n$, since this is the number of rows it contains. Therefore, the set of equations defined over the interval 0 to N is not solvable. ■

It follows from the above argument that if a time-invariant set od dynamic equations is solvable it is also conditionable. The reverse implication follows from the duality result of Theorem 2.1.

Time invariant systems of form (2.4) are included within the scope of earlier work (see e.g. [10]) employing techniques of modern algebra and related to classical transform methods. A standard assumption in this framework is that $\det|A-sE|\neq 0$. This condition will be shown to be equivalent to solvability (Chapter III). Therefore the two definitions of solvability and conditionability together appear to be the natural extension of the standard assumption, providing the bridge between existing time-invariant theory and a more general theory for time-varying systems. The time-invariant case is investigated further in Chapter III.

## 2.3 CONDITION VECTORS AND TIME DOUBLE SWEEP

### *Initial and Final Conditions*

The set of additional conditions required to completely specify a solution to a set of dynamic equations may take a variety of forms. The equations will each, in general, involve descriptor and input variables at several time points. A special form of additional relation, however, which is of great importance, is the pure initial or pure final condition which involves, respectively, only x(0) or only x(N), and no input values. The next theorem shows the universality of this special form of relation.

*Theorem 2.4:* If a set of dynamic equations is conditionable, then a complete set of additional conditions can be specified in terms of pure initial and pure final conditions.

*Proof:* Consider a set of equations defined on the interval 0 to N. As before, let F(0,N) and G(0,N) denote, respectively, the coefficient and condition matrix of the set of equations. Under the assumption that the set is conditionable, the matrix G(0,N) has full rank, equal to n(N-1). It follows that the matrix F(0,N) must have rank at least this great, but of course it can be no greater than the number of its rows. Thus, the rank of F(0,N) must lie between n(N-1) and nN. Denote this rank by r, and the difference nN-r by d. (In the usual case where F(0,N) is of full rank, d=0.)

The matrix G(0,N) is defined as the submatrix of F(0,N) obtained by deleting the first and last n columns. Now imagine adding some of these columns back to G(0,N) in order to get a total of r columns, all of which are linearly independent. A suitable selection is always possible since the full matrix F(0,N) has rank r. There will be n-d columns added in this way.

Now let $I_o$ be the set of indices i corresponding to those columns not selected in the first group of n columns. Similarly, let $I_N$ denote the set of indices corresponding to those columns not selected in the last group of n columns. (Both index sets $I_o$ and $I_N$ consist of integers between 1 and n). Together $I_o$ and $I_N$ consists of n-d elements.

The conditions $x_i(0)=a_i, i\in I_o$ and $x_j(N)=b_j$, $j\in I_N$ where the $a_i$ and $b_j$ are real numbers, can be specified arbitrarily and the total set of equations, consisting of these and the original set of dynamic equations, will be of full rank. ■

Suppose that a system is both solvable and conditionable and that a complete set of n auxiliary conditions is specified in terms of m initial conditions and n-m terminal conditions. It is convenient to view the conditions as the components of two separate condition vectors. Thus, one defines the vector $y(0)=\Gamma_o x(0)$ where $\Gamma_o$ is an mXn matrix of rank m, and specifies the $m^o$ initial conditions in the form $y(0)=a$ where $a$ is an arbitrary m-dimensional vector. Likewise a final condition vector of dimension n-m is defined by $z(N)=\Lambda_N x(N)$, and the final condition is written $z(N)=b$. A solution to the solvable and conditionable system is specified uniquely in terms of the values of its initial and final condition vectors y(0) and z(N).

### *Propagation of Condition Vectors*

Suppose initial and final conditions are specified for a set so that a unique solution is defined. Consider the subset of equations, obtained by deleting the first (n-dimensional) equation, corresponding to a set of dynamic equations starting at time 1 rather than time 0. A set of initial conditions for this subset can be specified, along with the original final conditions, so that the unique solution of this subset of equations will agree in x(k), k=1,2,...,N with the previously defined solution of the original full set of equations. These initial conditions, now specified at k=1, can be regarded as the propagation through one stage, of the original initial conditions. The propagation process can be repeated successively all the way to the final endpoint.

There are a number of ways to display the explicit formula for condition vector propagation. Since the essential ingredient of the procedure is the solution of a rectangular system of linear equations, many methods convenient for computation involve matrix partitioning, matrix augmentation, or formation of pseudoinverses. The method employed below is chosen for its relatively streamlined notation.

*Lemma:* Let $\Gamma$ be an mXn matrix of rank m, and let A be an nXn matrix. Then there exists an mXn matrix P of rank m and an mXm matrix R such that $R\Gamma=PA$.

*Proof:* The (n+m) X n composite matrix

$$\begin{bmatrix} \Gamma \\ A \end{bmatrix}$$

has rank no greater than n. Thus, there are m linearly independent row vectors of dimension n+m that are orthogonal to the columns of this composite matrix. Let these form the rows of an mX(m+m) matrix [R,-P], where R is mXm, and P is mXn. By construction $R\Gamma=PA$.

It remains to be shown that P is of full rank. Suppose to the contrary that there was a nonzero, m-dimensional vector $\lambda$ such that $\lambda^T P=0$. It would then follow that $0=\lambda^T(R\Gamma-PA)=\lambda^T R\Gamma$. However, since $\Gamma$ is of full rank this would imply $\lambda^T R=0$. Thus, $\lambda^T[R,-P]=0$, contradicting the original construction. ■

*Theorem 2.5:* Suppose the set of dynamic equations (2.3) is both solvable and conditionable. Let $y(0)=\Gamma_0 x(0)$ be an initial condition vector of dimension m. Then the corresponding successive condition vectors $y(k)=\Gamma_k x(k)$ are defined by the recursive relations

$$y(k+1)=R_k y(k)+P_k u(k) \qquad (2.5a)$$

$$R_k\Gamma_k=P_k A_k \qquad (2.5b)$$

$$\Gamma_{k+1}=P_k E_{k+1} \qquad (2.5c)$$

where in each case $P_k$ is chosen to have full rank.

*Proof:* Assume that $\Gamma_k$ has been defined and $y(k)=\Gamma_k x(k)$ computed. Starting with the original equation

$$E_{k+1}x(k+1)=A_k x(k)+u(k) \qquad (2.6)$$

multiply the matrix $P_k$ to obtain

$$P_k E_{k+1}x(k+1)=P_k A_k x(k)+P_k u(k).$$

Using the definition of $P_k$ from (5b) this reduces to

$$P_k E_{k+1}x(k+1)=R_k y(k)+P_k u(k). \qquad (2.7)$$

Now, since 1) y(k) is independent of u(k), 2) u(k) is arbitrary, and 3) $P_k$ has full rank, it follows that the right-hand side of (2.7) is arbitrary. Since the system is solvable it must follow that $P_k E_{k+1}$ is of full rank. Thus, it is legitimate to set $\Gamma_{k+1}=P_k E_{k+1}$ leading to

$$y(k+1)=R_k y(k)+P_k u(k). \quad ■$$

An interesting and important corollary of the formula $\Gamma_{k+1}=P_k E_{k+1}$ derived in the proof is that the rank of $\Gamma_{k+1}$ can be no greater than the rank of $E_{k+1}$. Thus, the $E_k$ of minimal rank establishes an upper bound on the number of initial conditions.

In a similar manner, the final conditions can be propagated backward through the time stages. The recursive propagation process is characterized by the following theorem:

*Theorem 2.6:* Suppose the set of dynamic equations (2.3) is both solvable and conditionable. Let $z(N)=\Lambda_N x(N)$ be a final condition vector of dimension n-m. Then the corresponding successive previous condition vectors $z(k)=\Lambda_k x(k)$ are defined by recursive relations

$$z(k-1)=S_k z(k)-Q_k u(k-1)$$

$$S_k\Lambda_k=Q_k E_k$$

$$\Lambda_{k-1}=Q_k A_{k-1}$$

where in each case $Q_k$ is chosen to have full rank.

The proof is similar to the proof of Theorem 2.5.

*The Double Sweep Method of Solution*

Any solvable and conditionable set of dynamic equations, augmented by any complete set of n initial and final conditions, can be solved by a double sweep method. One starts with the given initial conditions and propagates them forward, as described above, all the way to the termination point N. At this point n independent relations will be determined, and x(N) can be calculated. Knowing this vector and all previous condition vectors, one can then progress backward solving one at a time for all other descriptor vectors.

The backward phase of the solution progresses as follows. Given x(k+1) one considers the equations

$$\Gamma_k x(k)=y(k)$$

$$E_{k+1}x(k+1)=A_k x(k)+u(k).$$

These are the only equations involving x(k) in the subset of equations on the time interval k to N. It must therefore be possible to solve for x(k) is x(k+1),y(k) and w(k) are specified. In particular the composite matrix

$$\begin{bmatrix} \Gamma_k \\ A_k \end{bmatrix}$$

must be of full rank. Let $[L_k,M_k]$ be a left inverse of this composite, $L_k$ is nXm, $M_k$ is nXn, and $L_k\Gamma_k+M_k A_k=I$. Then one may write the backward recursion

$$x(k)=L_k y(k)+M_k E_{k+1}x(k+1)-M_k u(k). \qquad (2.8)$$

This is the back sweep which determines the successive descriptor vectors.

An alternative explanation of the double sweep can be constructed, as a combination of a forward propagation of the initial conditions and a backward propagation of the final conditions. By determining the forward condition vector y(k) and the backward condition vector z(k), the descriptor vector x(k) can be recovered by the

equation

$$x(k)=\begin{bmatrix}\Gamma_k & \Lambda_k\end{bmatrix}^{-1}\begin{bmatrix}y(k)\\ z(k)\end{bmatrix}$$

where $\begin{bmatrix}\Gamma_k & \Lambda_k\end{bmatrix}$ is always square and nonsingular for solvable and conditionable systems. Thus, instead of a forward propagation of y(k) followed by a reverse propagation to recover x(k), y(k) and z(k) can be independently propagated and then processed to determine x(k).

There are many special versions of this general double sweep method, depending on the nature of the detailed structure of the original set of equations and on the form of the additional conditions. If, for example, the original set were actually static (with $E_{k+1}=0$ for all k=0,1,2,..., N-1), then all n conditions must be specified at the final time. The forward sweep degenerates to nothing, and each stage could be solved individually. If, as another example, the set represented a standard dynamic system and n initial conditions were specified, the whole solution would be determined by the forward sweep.

## 2.4 STATE VECTORS AND REGULAR SYSTEMS

### *State Vectors*

In conjunction with systems in descriptor form, the following definition of a state is employed.

*Definition:* A condition vector $y(k)=\Gamma_k x(k)$ is a state for a set of dynamic equations if knowledge of its value and the value of u(k) are sufficient to uniquely determine the descriptor vector x(k).

In preparation for addressing the problem of characterizing the situations in which a given condition vector $y(k)=\Gamma_k x(k)$ is a state, it is useful to apply some slight manipulation to the original set of equations (2.2). Suppose that the matrix $E_{k+1}$ has rank m. Then by elementary row operations (that is, by forming linear combinations of the various equations) it is possible to transform (2.2) to the form

$$\begin{bmatrix}T_{k+1}\\ 0\end{bmatrix}x(k+1)=\begin{bmatrix}C_k\\ D_k\end{bmatrix}x(k)+\begin{bmatrix}u_1(k)\\ u_2(k)\end{bmatrix} \qquad (2.9)$$

where $T_{k+1}$ is an mXn matrix of rank m, and $C_k$ and $D_k$ are respectively, mXn and (n-m)Xn matrices. The u(k) vectors are partitioned, correspondingly, into sections of height m and n-m. The new $A_k$ matrix (partitioned into $C_k$ and $D_k$) is not the same as before, since it also has had its rows linearly combined. Likewise, the new u(k) vector is not quite the same as before. The definition of the vector x(k) remains unchanged. Rather than introduce new notation for the corresponding new unpartitioned $E_k$, $A_k$, and u(k), however, it is assumed that the system (2.2) as originally defined is in the form (2.9).

*Theorem 2.7:* A condition vector $y(k)=\Gamma_k x(k)$ is a state for the set of equations (2.9) if and only if the matrix

$$\begin{bmatrix}\Gamma_k\\ D_k\end{bmatrix} \qquad (2.10)$$

is square and nonsingular.

*Proof:* To solve for x(k) in terms of y(k) and u(k) one must be able to recombine the given linear equations in such a way as to produce a nonsingular set of n equations from which x(k) can be determined. However, none of the equations from time periods other than k can appear in such a combination because each of them involves a separate arbitrary input. Of the equations corresponding to time k, only those without x(k+1) may be used. Since $T_{k+1}$ has full rank, it follows that the only equations that can be used are

$$\begin{aligned}\Gamma_k x(k)&=y(k)\\ D_k x(k)&=-u_2(k).\end{aligned} \qquad (2.11)$$

Then for these to uniquely determine x(k), the matrix (2.10) must be of rank n. However, the number of rows in the matrix $\Gamma_k$ is never greater than the number of rows in $T_{k+1}$ (as explained in the discussion following Theorem 2.5). Therefore, the matrix (2.10) never has more than n rows. It follows that it must be both square and nonsingular.

The above argument establishes that the nonsingularity condition is necessary for $y(k)=\Gamma_k x(k)$ to be a state. The sufficiency follows directly, since x(k) can be found from (2.11). ■

The criterion for a condition vector to serve as a state depends on the time period k. It is entirely possible that, starting with a given condition vector, its propagation through successive time periods may produce state vectors at some time periods but not others. This is likely if the ranks of the $E_{k+1}$'s vary with k, for a condition vector can only be a state vector for periods k where it has dimension equal to the rank of $E_{k+1}$.

If a point is reached where the condition vector is a state vector, then that point serves as a kind of regeneration point. The descriptor vectors for all previous time periods can be determined without further propagation of the condition vector. Thus, state vectors greatly simplify the solution of a set of dynamic equations, even if they occur at only some time points.

### *Regular Systems*

From a computational viewpoint, a most ideal situation is where the set of dynamic equations has a state vector at every time instant. Sets of equations with this property are said to be *regular*. Regularity is essentially equivalent to "real-time representation" or causality, and is an extremely important special case of the general

descriptor variable framework.

*Definition:* A set of dynamic equations is said to be *regular* if there is an initial condition vector which when propagated forward serves as a state vector for every time period.

It can be noted, in order to relate this definition to the previous sections, that if a state exists at every time period, the set of equations is both solvable and conditionable. It is solvable because there is a unique x(k) for every u(k). It is conditionable because there is a complete set of initial (and final) conditions.

There is a simple characterization of regularity in terms of the structure of the matrices involved. It is best described in terms of the partitioned form of equations used in Section IV. Specifically, the equations are written as

$$\begin{bmatrix} T_{k+1} \\ 0 \end{bmatrix} x(k+1) = \begin{bmatrix} C_k \\ D_k \end{bmatrix} x(k) + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} \qquad (2.12)$$

where $T_{k+1}$ has full rank, $C_k$ has the same dimension as $T_{k+1}$, and u(k) is partitioned consistently with the rest of the equation. Although there are the same number of $T_k$ and $D_k$ matrices in these equations, their indexing is displaced by one unit. They can be considered in corresponding pairs, however, if a definition of $T_o$ and of $D_N$ is introduced. This notational device is used in the characterization of regularity.

Before stating the formal version of the result, it is worth pointing out one of its immediate consequences. The characterization of regularity is that the matrices

$$\begin{bmatrix} T_k \\ D_k \end{bmatrix}$$

all be square and nonsingular. Since the dimension of $D_k$ is determined by the dimension of $T_{k+1}$ and not by that of $T_k$, it follows immediately that each of the $T_k$ matrices must have the same dimension. Thus, to be regular a set must have constant rank in its $E_k$ (or equivalently $T_k$) matrices.

*Theorem 2.8:* The set of dynamic equations (2.12) is regular if and only if there is a $T_o$ and a $D_N$ such that all the matrices

$$\begin{bmatrix} T_k \\ D_k \end{bmatrix} \qquad (2.13)$$

k=0,1,...,N, are square and nonsingular.

If this condition is fulfilled, one may set

$$y(k) = T_k x(k)$$

and the equations can be represented in state-space form as

$$y(k+1) = C_k S_k y(k) - C_k V_k u_2(k) + u_1(k) \qquad (2.14a)$$

$$x(k) = S_k y(k) - V_k u_2(k) \qquad (2.14b)$$

where

$$\begin{bmatrix} S_k & \vdots & V_k \end{bmatrix} = \begin{bmatrix} T_k \\ D_k \end{bmatrix}^{-1} \qquad (2.14c)$$

*Proof:* First consider the necessity of the condition. Suppose that there is a state of dimension m at every time period defined by $y(k) = \Gamma_k x(k)$. Then by Theorem 6 it follows that the matrices (2.13) k=0,1,2,...,N-1 must all be square and nonsingular. In particular, this means that all $D_k$ matrices must have dimension (n-m)Xn. Also from the proof of Theorem 2.5 we know that $\Gamma_k$, k>0 can be written in the form $\Gamma_k = Q_k T_k$ for some mXm matrix $Q_k$. In fact $Q_k$ must be nonsingular since $\Gamma_k$ is of full rank. It follows that (2.13) is nonsingular for k=1,2,...,N-1. Now let $T_o = \Gamma_o$, and let $D_N$ be any (n-m)Xn matrix which will make (2.13) nonsingular for k=N. Then the required statements are all fulfilled.

To prove sufficiency, suppose that the nonsingularity property holds and suppose $T_o$ and $D_N$ are found so that the matrices

$$\begin{bmatrix} T_k \\ D_k \end{bmatrix}$$

k=0,1,2,...,N are all square and nonsingular. Suppose that the common dimension of the $T_k$'s is mXn. Let

$$\begin{bmatrix} S_k & \vdots & V_k \end{bmatrix} = \begin{bmatrix} T_k \\ D_k \end{bmatrix}^{-1} \qquad (2.14c)$$

where $S_k$ is nXm and $V_k$ is nX(n-m). Define the state

$$y(k) = T_k x(k).$$

Then given y(k) and u(k), x(k) can be found as the unique solution of

$$T_k x(k) = y(k)$$

$$D_k x(k) = -u_2(k)$$

as

$$x(k) = S_k y(k) - V_k u_2(k). \qquad (2.14b)$$

Then, y(k+1) can be computed from

$$y(k+1) = C_k x(k) + u_1(k)$$

or

$$y(k+1) = C_k S_k y(k) - C_k V_k u_2 + u_1(k). \qquad (2.14a)$$

Together (2.14a)-(2.14c) give the explicit state-space solution. ∎

In the case of regular sets of dynamic equations, it is possible to recombine the state representation with the static relation for the determination of the descriptor vector to produce a single forward recursive solution formula. The formula somewhat hides the essentially reduced dimensionality of the underlying state, but it may sometimes be convenient for numerical computation.

The recursion is found by substituting (2.14b) into (2.14a) and using $S_k T_k = I$. Thus,

$$\begin{aligned} x(k+1) &= S_{k+1} y(k+1) - V_{k+1} u_2(k+1) \\ &= S_{k+1} C_k S_k T_k x(k) - S_{k+1} C_k V_k u_2(k) \\ &\quad + S_{k+1} u_1(k) - V_{k+1} u_2(k+1) \end{aligned}$$

or in the final form

$$x(k+1) = S_{k+1} C_k x(k) + S_{k+1} u_1(k) - S_{k+1} C_k V_k u_2(k) - V_{k+1} u_2(k+1) \tag{2.15}$$

This gives an explicit recursion for x(k). The initial x(0) must, however, be determined from (2.14b).

## 2.5 SENSITIVITY ANALYSIS AND BOUNDARY CONDITIONS

In analyzing the solution to a descriptor system, the response of the descriptor variables to modifications of particular inputs or endpoint conditions is occasionally of interest. Straightforward sensitivity results exist for state-space systems where the descriptor vector x(k) is also a state. In decomposing a general solvable and conditionable descriptor system into forward and backward condition systems, forward and backward condition vector transition function can be derived that conveniently display these sensitivities.

Recall that the forward condition system for linear descriptor systems is given by

$$y(k+1) = R_k y(k) + P_k u(k) \tag{2.16}$$

where y(0) is given, and $y(k) = \Gamma_k x(k)$. The backward condition system is represented by

$$z(k) = S_{k+1} z(k+1) - Q_{k+1} u(k) \tag{2.17}$$

where z(N) is given, and $z(k) = \Lambda_k x(k)$. As noted in Section 2.3, the matrices $R_k, P_k, \Gamma_k, S_{k+1}, Q_{k+1}$ and $\Lambda_k$ are generated recursively from matrices in the system equations (2.3).

Suppose one is interested in the effect of input u(i) on descriptor vector x(j). Using the forward condition system, one can determine a relationship expressing y(j) as a function of y(0), u(0), u(1)...,u(j-1):

$$y(j) = R_{j-1} R_{j-2} \cdots R_0 y(0) + \sum_{k=0}^{j-2} R_{j-1} R_{j-2} \cdots R_{k+1} P_k u(k) + P_{j-1} u(j-1) \tag{2.18}$$

Similarly, the backward condition system will generate a relationship expressing z(j) as a function of u(j),u(j+1),...,u(N-1), z(N):

$$z(j) = S_{j+1} S_{j+2} \cdots S_N z(N) - \sum_{k=j+1}^{N-1} S_{j+1} S_{j+2} \cdots S_k Q_{k+1} u(k) - Q_{j+1} u(j) \tag{2.19}$$

Recall that the condition vectors y(j) and z(j) uniquely determine x(j) by the relationship

$$x(j) = L_j y(j) + M_j z(j) \tag{2.20}$$

where $L_j$ and $M_j$ are defined by:

$$\left[\begin{array}{c|c} L_j & M_j \end{array}\right] = \left[\begin{array}{c} \Gamma_j \\ \hline \Lambda_j \end{array}\right]^{-1} \tag{2.21}$$

Therefore x(j) can be expressed as a function of the inputs:

$$x(j) = \bar{D}\,\bar{u} + \sum_{k=0}^{N-1} D_k u(k) + \tilde{D} z(N) \tag{2.22}$$

where

$$\bar{D} = L_j R_{j-1} R_{j-2} \cdots R_0 \tag{2.23}$$

$$D_k = \begin{cases} L_j R_{j-1} R_{j-2} \cdots R_{k+1} P_k & \text{if } 0 \le k \le j-2 \\ L_j P_{j-1} & \text{if } k = j-1 \\ -M_j Q_{j+1} & \text{if } k = j \\ -M_j S_{j+1} S_{j+2} \cdots S_k Q_{k+1} & \text{if } j+1 \le k \le N-1 \end{cases} \tag{2.24}$$

$$\tilde{D} = M_j S_{j+1} S_{j+2} \cdots S_N \tag{2.25}$$

The effect of the initial condition vector y(0), the final condition vector z(N), or any input vector u(k) on a particular descriptor variable appears explicitly in (2.22)-(2.25). The noncausality of a general descriptor system is clearly evident in (2.22), in that a descriptor vector x(j) can be influenced by the final conditions and future inputs. The influence of future variables, however, also depends on the manner in which the initial and final conditions are structured, namely the matrices $\Gamma_0$ and $\Lambda_N$, and there is generally some freedom in selecting the structure and dimension of these matrices. If the structure is modified, i.e. not simply changing the condition vectors y(0) and z(N),

the matrix coefficients in (2.22) will also be modified.

The range of valid and complete sets of initial and final conditions can be determined from the matrix F(0,N) of coefficients in the system (2.3). The row vectors of this matrix span the subspace of solutions to the descriptor system, in which a point is uniquely determined for a particular trajectory of inputs by the matrix equation (2.3). Thus, the added conditions must span a subspace such that the direct sum of this subspace with the subspace characterized by F(0,N) is the entire solution space. Solvability and conditionability guarantee this second subspace can be defined purely in terms of x(0) and x(N), and has dimension n. Such a subspace is called the *boundary manifold*. The conditions $\Gamma_o x(0) = y(0)$ and $\Lambda_N x(N) = z(N)$ are a valid and complete set of boundary conditions if the row vectors represented by the matrices

$$[\Gamma_o \; 0 \ldots 0]$$

and

$$[0 \ldots 0 \; \Lambda_N]$$

form a basis for the boundary manifold. Additional discussion about boundary manifolds appears in the analysis of nonlinear descriptor systems in Chapter IV.

## 2.6 APPLICATIONS TO SYSTEMS THEORY

The descriptor framework accommodates a broad spectrum of problems in the system theory area. The theory unifies a collection of existing techniques and provides a basis for development of new techniques. This section illustrates this feature by showing how several standard results can be viewed in terms of descriptor variable theory.

### *Dynamic Leontief Systems*

A dynamic Leontief model describes the time pattern of production in n interrelated economic production sectors. The model has the form

$$x(k) = Ax(k) + B[x(k+1) - x(k)] + d(k) \qquad (2.26)$$

The components of the n-dimensional vector x(k) are the levels of production in the sectors at time k. This production is divided into three parts, corresponding to the three terms on the right-hand side of (2.26). The first term, Ax(k), is the amount required as direct input for the current production. The nXn matrix A is the *input-output matrix* and has nonnegative entries. The second term is the amount required for capacity expansion, in the form of capital, in order to be able to produce x(k+1) in the next period. The matrix B is called the *capital coefficients* matrix and also has nonnegative entries. The third term, d(k), is the amount of production going to current demand.

Typically the capital coefficients matrix B has nonzero entries in only a few rows, corresponding to the fact that capital is formed from only a few sectors. Thus, B is singular, and the dynamic Leontief system is described in the implicit form that is characteristic of the descriptor representation. Development of efficient techniques for manipulation of such systems, and delineation of the conditions under which such systems are causal, are important problems which fall in the domain of the theory presented in this report. Further disucssion of this important example is contained in [11].

### *Reachability*

Consider an n-dimensional system

$$x(k+1) = Ax(k) + bu(k) \qquad (2.27)$$

Such a system is *completely reachable* if the state vector can be driven from the origin to an arbitrary point within n steps. That is, given x(0)=0 and $\bar{x}$ arbitrary, there is the set of inputs u(0), u(1),...,u(n-1) such that the solution to (2.27) has $x(n) = \bar{x}$.

There is, of course, a standard test for complete reachability. Let us briefly observe how the reachability problem can be converted to a descriptor variable problem and how the standard test can be derived from that viewpoint.

Replace the system (2.27) by the 2n-dimensional system

$$x(k+1) = Ax(k) + be_1^T y(k) \qquad (2.28)$$

$$y(k+1) = Dy(k)$$

In this system x(k) and y(k) are both n-dimensional vectors. The vector $e_1$ is the first coordinate vector. Thus $e_1^T y(k) = y_1(k)$. The n x n matrix D has entries all 0, except that those on the super-diagonal (the diagonal immediately above the main diagonal) are all 1's. It is then clear that (2.27) is equivalent to (2.28) with the association that $u(k) = y_{k-1}(0)$. In other words, the vector y(0) defines the input sequence.

In terms of descriptor variable theory, the question of the reachability of (2.27) is a question concerning the boundary manifold of (2.28). The intersection of this manifold with x(0) = 0 must contain all values of x(n). The system (2.28) is solvable and conditionable, since in fact it is in state-space form. The boundary manifold is easily determined in this case (as discussed in Section 2.5, and is exceedingly well-known) to be

$$\begin{bmatrix} x(n) \\ y(n) \end{bmatrix} = \begin{bmatrix} A & be_1^T \\ 0 & D \end{bmatrix}^n \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}$$

The $n^{th}$ power of the transition matrix is easily calculated to be

$$\begin{bmatrix} A^n & C^T \\ 0 & D^n \end{bmatrix}$$

where C is the controllability matrix

$$[b,\ Ab,\ A^2b,\ldots,A^{n-1}b].$$

For points of the form $x(0) = 0$, $x(n) = \bar{x}$, with $\bar{x}$ arbitrary, to be in the boundary manifold it is necessary and sufficient that C be nonsingular. This is the usual criterion.

*System Inversion*

Consider the linear system

$$x(k+1) = Ax(k) + Bu(k) \tag{2.29a}$$

$$y(k) = Cx(k) + Du(k) \tag{2.29b}$$

This is a state-space system with state x, input u and output y. As a descriptor variable system, the descriptor variables are x and y; the input is u. From either viewpoint, the usual problem is to specify an input sequence and determine the corresponding output sequence.

The problem of system inversion is to interchange the roles of u and y. Given an output sequence, one wishes to determine the corresponding input sequence. This problem is solved routinly by descriptor variable methods by simply regarding x and u as descriptor variables and y as the input variable. With this identification (2.29) is still a standard descriptor variable system. The general methods discussed in this report can be used to solve for the descriptor variables given the sequence y. Or, more generally, the admissible relations between y and u can be determined.

An explicit structure for the inverse system can be derived in the linear case by application of the shuffle algorithm (Section 3.4). This is equivalent to the inversion procedure developed by Silverman [12]. Using the framework of Chapter IV, the shuffle algorithm can be extended to some nonlinear systems, thus providing a methodology for inversion in the nonlinear case.

*Dynamic Programming*

Consider the variational problem of finding a stationary value of the objective

$$J = \sum_{k=0}^{N} g(x(k),u(k)) \tag{2.30a}$$

for the dynamic system

$$x(k+1) = f(x(k),\ u(k)) \tag{2.30b}$$

subject to a given initial condition $x(0) = x_0$. As is well-known the necessary conditions for this problem can be expressed by introducing the sequence of adjoint variables $\lambda(k)$. Together the optimal control vector, optimal input, and adjoint variable sequences satisfy the two-point boundary problem

$$x(k+1) = f(x(k),\ u(k)) \tag{2.31a}$$

$$\lambda(k) = \lambda(k+1)f_x(x(k),u(k)) + g_x(x(k),u(k)) \tag{2.31b}$$

$$\lambda(k+1)f_u(x(k),u(k)) + g_u(x(k),u(k)) = 0 \tag{2.31c}$$

The boundary conditions are $x(0) = x_0$, and $\lambda(N) = 0$.

The system (2.31) is a nonlinear dynamic system in descriptor form, with boundary conditions at each end. If the system is consistent, these boundary conditions must lie on the boundary manifold. Assuming this to be the case, it is natural to attempt to solve the system (2.31) by the double-sweep method (described in Section 2.3). This leads (essentially) to dynamic programming. The details are briefly sketched below.

As is conventional for this problem, we shall sweep backwards first. Specifically, the terminal boundary constraint $\lambda(N) = 0$ is swept backward. This leads to an n-dimensional relation between $\lambda$ and x at each time point k. In this particular problem, the relation can be taken to have the form $\lambda(k) = \lambda(x(k),k)$. That is, $\lambda$ is a function of x.

The sweep procedure is defined by the following two equations:

$$\lambda(x(k),k) = \lambda(f(x(k),\ u(k)),\ k+1)f_x(x(k),\ u(k)) + g_x(x(k),u(k)) \tag{2.32a}$$

$$\lambda(f(x(k),\ u(k)),\ k+1)f_u(x(k),u(k)) + g_u(x(k),\ u(k)) = 0 \tag{2.32b}$$

Assume that the function $\lambda(x,k+1)$ is known. Given an arbitrary x(k), one may then solve (2.32b) for u(k). That is, (2.32b) determines a function u(x(k),k). Substitution of this function into equation (2.32a) leads to an explicit equation for the function $\lambda(x(k),k)$. This procedure can be continued all the way to the initial point k=0. At that point x(0) and $\lambda(0)$ are known and the system can be solved by a forward sweep. This is a special instance of the double-sweep method.

In this particular case there is an important further observation. It is easily verified that for each k, the Jacobian matrices $\lambda_x(x,k)$ are symmetric. This means that they are gradients of functionals. These functionals are easily found to be the optimal return functionals. That is, $\lambda(x,k) = J_x(x,k)$, where J(x,k) is the optimal return from state x at time period k. In practice, of course, it is most convenient to use this fact to reformulate the double-sweep method in terms of the functionals rather than their gradients.

Dynamic programming can also be applied to optimizing the inputs to descriptor variable systems, where (2.30b) would have the form

$$0 = g(x(k+1), x(k), u(k)).$$

Dynamic programming for linear descriptor systems is discussed in Chapter V.

*The Linear Double-Sweep and Riccati Equations*

In the linear case, the bulk of the effort of the double-sweep can be carried out "off-line", without knowledge of the particular input sequence u(k). Once the input sequence u(k) becomes available, the sweep can then be executed quite simply.

A special but important instance of this linear double-sweep algorithm is the standard method for solving the linear two-point boundary value difference equation arising in optimal control. The "Riccati-type" difference equation that is fundamental to this method is the result of the preparation phase of the linear double-sweep. This is, of course, a special instance of dynamic programming.

*Price System*

We present now a simple example which illustrates a class of important potential applications of this theory. This example illustrates how a descriptor variable framework serves to generalize the traditional techniques discussed above.

Consider an inventory system governed by

$$x(k+1) = x(k) + h(k) - d(k) \qquad (2.33a)$$

Here x(k) is an n-dimensional vector of commodity inventories at period k, h(k) is a vector of commodity production, and d(k) is the vector of demands. We assume that the production is a fixed sequence h(k). Demand on the other hand is determined by a demand function

$$d(k) = f(p(k)) \qquad (2.33b)$$

where p(k) is the n-dimensional vector of commodity prices at period k. We assume that the inventory x(k) is held by a very large number of independent agents. Each of them faces storage costs of c dollars per unit and per time period. Assuming that prices are determined rationally [13], the equilibrium condition for prices is

$$p(k) = p(k+1) - c \qquad (2.33c)$$

There is a known initial inventory vector $x(0) = x_0$, and the final inventory must be $x(N) = 0$. The system (2.33) together with the boundary conditions determines the equilibrium prices and inventories. This is a descriptor variable system with boundary conditions at both ends (and, in this case, the boundary conditions are all on x).

This system can be solved by the double-sweep method. In a manner similar to that used in the example on dynamic programming, we define the functions x(p,k). Working backward we begin with

$$x(p,N) = 0$$

We then employ the recursion

$$x(p,k) = x(p - c, k + 1) - h(k) + f(p)$$

This recursion is continued backward until x(p,0) is determined. Then the equation $x(p(0),0) = x_0$ can be solved for p(0). The solution to (2.33) can then be determined by a forward sweep from the known initial conditions.

In general, this is as efficient as the scheme can be made. With additional assumptions, however, simplification is possible - much like the dynamic programming example above.

For example, suppose that the function $f_p(p)$ is symmetric. Then f is the gradient of some function: say, $f(p) = F_p(p)$. A scalar-valued recursion can now be defined as follows: Let $J(p,N) = 0$ and

$$J(p,k) = J(p-c, k + 1) - h(k)p + F(p)$$

This scalar-valued recursion is continued to $k = 0$. As before, p(0) is then found by solving

$$J_p(p(0),0) = x_0.$$

As a further assumption, suppose that F(p) is concave. In that case, the functions J(p,k), which represent maximum social surplus from period k, are all concave. The issue of calculating p(0) can be expressed as a maximization problem

$$\text{Maximize } J(p,0) - x_0 p$$

More complicated examples can be handled in a similar fashion. In general, if the problem is well-defined, the double-sweep method will provide a general method of attack. Additional structure and symmetry can be used to simplify the general procedure.

# III. TIME-INVARIANT LINEAR DESCRIPTOR SYSTEMS

## 3.1 INTRODUCTION

It is often natural and convenient to express the equations governing a dynamic process by a system of equations of the form

$$Ex(k+1) = Ax(k) + Bu(k), \quad k=0,1,2,\ldots,N-1 \tag{3.1}$$

Such a system is said to be a (discrete-time) linear time-invariant system in descriptor form. Equation (2.1) is referred to as a time-invariant system, since the matrices E, A, and B are fixed, independent of k. As one would expect, stronger conclusions, especially concerning structure, can be deduced for the time-invariant case than for the more general case, and this chapter presents these results. It should be pointed out however, that in the time-invariant case there are several alternative approaches (notably including polynomial methods [14], but see also [15] and [16]. These of course yield results that overlap with some of those presented here. The important distinction of the present work is that the fundamental concepts and the basic approach are not limited to the time-invariant case. Thus, although some of the results presented in this chapter are not strictly new, one of the objectives of the chapter is simply to illustrate the form of the general descriptor variable theory when specialized to the time-invariant case. The fact that in the time-invariant case the descriptor variable results are consistent with those obtainable by other procedures would appear to indicate that the general framework is perhaps a natural one.

The structural character and the behavioral pattern of a system of the form (3.1) can be surprisingly complex. Thus: the system may not have a solution; if it does have a solution, that solution may correspond to pure prediction of the input; and the number of degrees of freedom in the initial condition cannot always be determined by inspection. The first few sections of this chapter examine the general structural properties of time-invariant descriptor systems culminating in the presentation of a canonical form.

From a practical viewpoint one is concerned primarily with those systems of form (3.1) which are well-behaved, and represent reasonable models of reality. Interest then turns to the development of simple procedures to test that a system has the desired structural characteristics, and for converting the system to a form that can be easily solved. Both of these objectives are met by the development of the *shuffle algorithm* described in Section 3.4.

Using the canonical form derived in Section 2.3, time-invariant descriptor systems can generally be decomposed into independent, time-invariant subsystems that are individually very special types of descriptor systems. This decomposition eases the analysis of the dynamic system, such as controllability, stability, aggregation, and parameter sensitivity. The canonical decomposition also provides a convenient framework for characterizing valid boundary conditions for the system. These topics are considered in the final two sections of the chapter.

## 3.2 SOLVABILITY AND CONDITIONABILITY

Consider briefly the time-varying situation, defined by a set of equations of the form

$$E_{k+1}x(k+1) = A_k x(k) + B_k u(k), \quad k = 0,\ldots,N-1. \tag{3.2}$$

Following Chapter II, such a system is said to be *solvable* if the matrix

$$F(0,N) = \begin{bmatrix} -A_0 & E_1 & & & \\ & -A_1 & E_2 & & \\ & & \ddots & \ddots & \\ & & & -A_{N-1} & E_N \end{bmatrix}$$

is of full rank. The system (3.2) is said to be *conditionable* if the matrix

$$G(0,N) = \begin{bmatrix} E_1 & & & & \\ -A_1 & E_2 & & & \\ & -A_2 & \ddots & & \\ & & \ddots & E_{N-1} & \\ & & & -A_{N-1} & \end{bmatrix}$$

is of full rank.

The corresponding definitions in the time-invariant case with constant matrices E and A are exactly the same, with the additional requirement that the full rank criteria be satisfied for all N. Thus, (3.1) is solvable (or conditionable) if F(0,N) (or G(0,N)) is of full rank for every

integer N > 0. Theorem 2.3 demonstrated that in the time-invariant case, a system is solvable if and only if it is conditionable. A simple criterion for solvability (and therefore of conditionability) is contained in the statement of Theorem 3.1 below.

Before proceeding to the theorem, however, it is appropriate to recall a few facts about equivalent matrices. A matrix P(s) whose elements are polynomials in s is a *polynomial matrix*. A square polynomial matrix whose determinant is a constant, independent of s, is said to be *unimodular*. Two polynomial matrices P(s) and R(s) are said to be *equivalent* if there are nonsingular unimodular matrices U(s) and V(s) such that U(s)P(s)V(s) = R(s). An alternate, but equivalent, characterization is that P(s) and R(s) are equivalent if P(s) can be transformed into R(s) by a series of elementary row and column operations. Elementary row (column) operations consist of either (i) multiplication of a row (column) by a constant, (ii) interchange of two rows (columns), or (iii) addition of a polynomial multiple of one row (column) to another. In terms of the relation U(s)P(s)V(s) = R(s) the matrix U(s) defines the row operations and can itself be obtained by performing these same row operations on the identity matrix. Similarly, V(s) represents the column operations.

*Theorem 3.1:* The system (3.1) is solvable (and conditionable) if and only if the determinant $|A-sE|$ does not vanish identically.

*Proof:* The matrix F(0,N) has more columns than rows. Therefore it is of less than full rank if and only if there is a linear dependency among its rows. This in turn is true if and only if there is a row vector $q \neq 0$ such that $qF(0,N) = 0$, in which case the vector q explicitly displays the row dependency. Write such a vector in the form $q = [q_1 q_2, \ldots, q_N]$ where each $q_i$ is of dimension n. By the structure of the matrix F(0,N), it is clear that it is of less than full rank if and only if the polynomial matrix

$$P(s) = \begin{bmatrix} -A & Es & & & \\ & -As & Es^2 & & \\ & & -As^2 & Es^3 & \\ & & & \ddots & \ddots \\ & & & & -As^{N-1} \quad Es^N \end{bmatrix}$$

is of less than full rank

Moreover, a q vector which explicitly displays the dependence of rows for either F(0,N) or for P(s) also displays it for the other. For a given q, define the n-dimensional row vector q(s).

$$q(s) = q_1 + q_2 + q_3 s^2 + \ldots + q_N s^{N-1}.$$

Now it is apparent that the relation qP(s)=0 is equivalent to the relation q(s)[A-Es]=0. Therefore, the original matrix F(0,N) has linearly dependent rows if and only if the polynomial matrix A-Es has linearly dependent rows, allowing for rows to be multiplied by powers of s up to N-1. ■

For the system (3.2) to be solvable, the full rank condition must hold for every N. Therefore, solvability is equivalent to the requirement that the matrix A-Es be of full rank with respect to all polynomial combinations of its rows; that is, the system is solvable if and only if A-Es is not equivalent (in the sense of polynomial matrices) to a matrix with a zero row. Alternatively, (and finally) the system is solvable if and only if $|A-Es|$ does not vanish identically.

## 3.3 CANONICAL STRUCTURE OF A SOLVABLE SYSTEM

Equivalence is a natural concept in the study of systems in descriptor form. Consider the time-invariant system

$$Ex(k+1) = Ax(k)+u(k) \qquad (3.3)$$

where for simplicity the input coefficient matrix is taken to be the identity. Multiplication on the left by a nonsingular matrix V and introduction of the nonsingular change of variable x(k)=Wy(k) yields the system

$$VEWy(k+1) = VAWy(k)+v(k) \qquad (3.4)$$

where v(k) = Vu(k) is the new vector of arbitrary inputs. The matrices E and A in the original system have been replaced by equivalent matrices $E_1$ and $A_1$, each obtained by the same equivalence transformation. It is therefore quite natural to investigate the range of possible equivalent (E,A) pairs.

The study of simultaneous equivalence transformations of A and E is most conveniently investigated by consideration of the polynomial matrix A-Es referred to as a *matrix pencil*. Two pencils, $A_1-E_1 s$ and A-Es, are equivalent if there are nonsingular matrices V and W such that $V[A-Es]W = A_1-E_1 s$. In this case, unlike the situation for general polynomial matrices, one requires that the matrices V and W be constant matrices. This is often emphasized by referring to this relation as *strict equivalence*. Certainly within the context of the system (3.3) and its alternative representation (3.4) attention is restricted to strict equivalence.

A matrix pencil A-Es which is square and for which A-Es does not vanish identically is traditionally termed *regular* (see for example [17] or *nonsingular* (see for example [18], and strong characterization results exist for this case. With either terminology, this condition precisely coincides with the concept of solvability, and hence the associated characterization results for these pencils can be directly applied.

In what follows it is convenient to refer to the *degree d* of the solvable system (3.3) or of

the pencil A-Es as the degree of the (nonzero) polynomial $|A-Es|$. Also, before stating the structure theorem itself, we consider the structure of the pure predictor, which occurs in the canonical form of a system in descriptor form.

*The Pure Predictor*

Consider the system (3) with

$$E = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & & & & \cdot \\ & & & & \cdot \\ & & & & \cdot \\ & & & & 1 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ & & & & \cdot \\ & & & & \cdot \\ & & & & \cdot \\ & & & & 0 \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

and k=0,1,2,...,N. This system is easily verified to be solvable, for indeed $|A-Es| = 1$.

The corresponding individual equations are

$$\begin{aligned} x_2(k+1) &= x_1(k) + u_1(k) \\ x_3(k+1) &= x_2(k) + u_2(k) \\ &\cdot \\ &\cdot \\ &\cdot \\ 0 &= x_n(k) + u_n(k). \end{aligned} \qquad (3.5)$$

These equations can be solved explicitly, starting with the last one, yielding

$$\begin{aligned} x_n(k) &= -u_n(k) \\ x_{n-1}(k) &= -u_n(k+1) - u_{n-1}(k) \\ &\cdot \\ &\cdot \\ &\cdot \\ x_1(k) &= -u_n(k+n-1) \\ &\quad -u_{n-1}(k+n-2) - \dots - u_1(k) \end{aligned}$$

where the equation for $x_i(k)$ is valid for k=0,1,2,...,N-n+i. The system represents a pure predictor, with the variable $x_1(k)$ depending on $u_n(k+n-1)$. No initial conditions can be arbitrarily specified in this system. The n arbitrary constants in the solution are the *final* values of the variables.

An important special case of the general predictor system (3.5) is the case n=1. This yields the scalar system

$$0 = x(k)+u(k)$$

which is a static equation without actual prediction. It is conventional to regard such a system as causal, while for any n > 1 the system (3.5) is noncausal.

*Structural Theorem*

A structure theorem for solvable systems follows directly from the classic result due to Weierstrass, see [17], on the canonical decomposition of a nonsingular matrix pencil. In the following, $I^{(r)}$ denotes the r x r identity matrix, $H^{(r)}$ denotes the r x r matrix whose elements are all zero except that those along the diagonal directly above the main diagonal are equal to 1. The matrix $N^{(r)}$ is defined as $N^{(r)} = I^{(r)} - H^{(r)}$s.

*Theorem 3.2:* (Weierstrass). A nonsingular matrix pencil of degree d, A-Es, is strictly equivalent to the pencil having the diagonal block form

$$[N^{(r_1)}, N^{(r_2)}, \dots, N^{(r_m)}; \; C-Is]$$

where the final block is dxd. The integers $r_1, r_2, \dots r_m$ are unique, and correspond to the infinite elementary divisors of the pencil.

Of course the matrix C in the final block can be transformed by a similarity transformation to any of the standard canonical forms for square matrices. For the present purposes, however, it is not necessary to further specify C.

The system version of the theorem is the following (for a previous systems-theoretic application of the canonical form theory of pencils to this problem see [16]):

*Theorem 3.2:* A solvable system (3.3) of degree d is strictly equivalent to the direct sum of a number of pure predictors, purely static relations, and a system in state variable form. The dimension of the state is d.

An important special case is when each of the $r_i$'s in the canonical representation is I. In this case the system is purely causal consisting of a dynamic part and a static part. Such systems were defined as *regular* (Section 2.4).

## 3.4 THE SHUFFLE ALGORITHM

Although the canonical form derived from the classical theory of matrix pencils provides deep insight into the underlying structure of time-invariant descriptor systems, it does not always provide a convenient framework for actual computation. The main drawback is that the canonical form entails a change of variable. In most practical situations, one is usually reluctant to execute a variable change, since the original descriptor variables have contextual as well as structural significance, and since there may be additional implicit constraints, such as nonnegativity constraints, on the variables. In addition, of course, the canonical form can be difficult to compute. Thus, interest turns toward

the development of techniques which are computationally efficient and do not require a change of variable.

This section describes the basic shuffle algorithm as used to check solvability of a system. The extended version of the algorithm is deferred to momentarily.

Solvability is a property of only the matrices E and A. Accordingly, the matrix B plays no role in the simplified version of the algorithm. The algorithm works by modifying an n x (2n) array.

Begin with the array

$$E \quad A.$$

If E is nonsingular, the procedure terminates - the system is solvable.

Otherwise, perform row operations on the whole array, bringing it to the form

$$\begin{matrix} T & A_1 \\ 0 & A_2 \end{matrix}$$

where T is of full rank. (T has n columns, but less than n rows.) The matrices $A_1$ and $A_2$ are a partition of the second side of the array after the row operations. $A_1$ is the same size as T.

Next 'shuffle' the array to form

$$\begin{matrix} T & A_1 \\ A_2 & 0 \end{matrix}$$

If the n x n matrix on the left side of the array is nonsingular, the procedure terminates - the system is solvable.

The algorithm continues in this fashion, performing row operations in order to create null rows on the left side, and then shuffling the corresponding rows from the right side to the left. The algorithm terminates in one of two ways: (1) a point is reached where the left half becomes nonsingular, in which case the system is solvable, or (2) a point is reached where there is a zero row all the way across the array, in which case the system is not solvable. The algorithm always terminates, one way or the other, in at most n steps.

*Example 1:* Starting with the E A array below, the shuffle progresses as indicated.

| E | | | A | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |

Row operations yield

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | -1 | 1 | 0 |

A shuffle yields

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| -1 | 1 | 0 | 0 | 0 | 0 |

More row operations yield

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | -1 | 0 | 1 |

Another shuffle yields

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| -1 | 0 | 1 | 0 | 0 | 0 |

The algorithm terminates because the left side is nonsingular. Thus, the system is solvable.

*Justification*

An easy way to see that the shuffle algorithm checks for solvability is to consider the determinant of A-sE. According to Theorem 3.1, solvability is equivalent to the condition that this determinant not vanish identically.

Row operations on A-sE at most influence the determinant by a nonzero multiplicative constant. Thus, one may as well check the determinant when E has the special form obtained by the first step of the algorithm. The shuffle of $A_2$ over to the other side of the array is equivalent to multiplication of the lower rows by -s, and each such multiplication multiplies the determinant by -s. Thus, it is clear that the shuffle algorithm is equivalent to a transformation of the original matrix pencil to a new pencil whose determinant is the original determinant multiplied by a nonzero constant and $s^b$ where b is the total number of rows shuffled. If a point is reached where an entire row is zero, the determinant is zero. If a point is reached where the (new) E is nonsingular, the determinant is then seen to be nonzero. One of these two situations must arise within n steps, for every row shuffled increases the degree of the determinant of the (modified) matrix pencil by one, and the maximum possible degree is n.

*The General Shuffle Algorithm*

The general shuffle algorithm accounts for the input structure of a system and produces a recursive system, equivalent to the original system. In developing the more general version, it seems best to regard the algorithm as operating

directly on the original descriptor system equations (3.1). The general shuffle algorithm consists of the repetition of two basic operations on these equations. The first operation is that of row combination, corresponding to linearly combining individual equations. One performs such operations with the objective of obtaining an E matrix with one or more zero rows. The second operation, the shuffle, is a reindexing operation. Each (row) equation in (3.1) is valid for all $k \geq 0$, and hence $k + 1$ can be substituted for $\bar{k}$ in any row if desired. Such a substitution is used in an equation corresponding to a zero in E. This then transfers the corresponding row in A to one in E and shifts the input terms from k to k+1. Any sequence of such row operations and time reindexing is permissible- the shuffle algorithm is a systematic procedure for obtaining a desired final form.

In the general algorithm it is often useful to restrict the kind of row operations performed, so that the final form will have a structure that is easily converted to recursive form. There are numerous variations possible, depending on the particular objectives of the situation. Two methods are outlined here.

*Non-reduced Form*

The general shuffle algorithm begins with the array

$$E \quad A \quad B$$

By row operations this is brought to the form

$$\begin{matrix} T & A_1 & B_1 \\ 0 & A_2 & B_2 \end{matrix}$$

A shuffle is performed yielding

$$\begin{matrix} T & A_1 & B_1 & 0 \\ A_2 & 0 & 0 & -B_2 \end{matrix}$$

This corresponds to writing $0 = A_2x(k)+B_2u(k)$ from the previous array, as $A_2x(k+1) = -B_2u(k+1)$. In general, any shuffle to the left of rows of A is accompanied by a shuffle to the right, and a change in sign, of all input structure elements in the same row. The array, therefore, grows toward the right as the algorithm progresses.

When the algorithm is complete, the array will have the form

$$\bar{E} \quad \bar{A} \quad \bar{B} \quad \bar{C} \ldots$$

If the system is solvable $\bar{E}$ will be nonsingular. Thus one may write

$$x(k+1)=\bar{E}^{-1}\{\bar{A}x(k)+\bar{B}u(k)+\bar{C}u(k+1)+\ldots\} \qquad (3.6)$$

which is a recursive structure for x(k). This is termed a non-reduced form, since the recursion is in terms of the full descriptor vector x(k). This is usually not the most convenient form, however, and it is slightly misleading. The vector x(0) cannot be selected arbitrarily, for there are additional equations at $k = 0$, relating x(0) and u(0), which were lost in the shuffle procedure. The procedure below employs a 'back shuffle' which recovers these lost equations.

*Reduced Form*

The reduced form is obtained by restricting the class of row operations employed during the shuffle algorithm in order to preserve the zero rows created in A. Thus after reaching the stage

$$\begin{matrix} T & A_1 & B_1 & 0 \\ A_2 & 0 & 0 & -B_2 \end{matrix} \qquad (3.7)$$

rows from the upper portion are never added to the lower portion. Arbitrary row operations are permitted within each portion, and multiples of lower rows may be added to upper rows. This rule does not actually restrict the functioning of the algorithm.

Assuming the system is solvable, a final stage is reached having the form

$$\begin{matrix} T & A_1 & B_1 & C_1 & D_1 \ldots \\ A_2 & 0 & 0 & -B_2 & -C_2 \ldots \end{matrix}$$

The left-hand n x n matrix can, by the allowed row operations, be brought to the special form

$$\begin{bmatrix} T \\ A_2 \end{bmatrix} = \begin{bmatrix} I & 0 \\ A_{21} & I \end{bmatrix} \qquad (3.8)$$

which is nonsingular.* Once this final stage is reached, the array is 'back shuffled,' yielding the array

$$\begin{matrix} T & A_1 & B_1 & C_1 \ldots \\ 0 & A_2 & B_2 & C_2 \ldots \end{matrix} \qquad (3.9)$$

Using the assumed special structure for $A_2$ and T, combinations of lower rows can be subtracted from upper rows to yield an array of the form

$$\left[\begin{array}{cc|cc|c|c} I & 0 & A_{11} & 0 & B_1 & C_1 \\ 0 & 0 & A_{21} & I & B_2 & C_2 \end{array}\right] \ldots \qquad (3.10)$$

The matrices $B_1$, $C_1$,... will generally have different entries than in (3.9).

Let x be partitioned, consistent with (3.10), as

$$= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

*Actually in some cases it may be necessary to permute the variables $x_i$, i=1,2,...n to obtain this form. We do not account for this possible permutation in our notation.

Then (10) yields

$$x_1(k+1) = A_{11}x_1(k)+B_1u(k)+C_1u(k+1)+... \quad (3.11a)$$

$$-x_2(k)=A_{21}x_1(k)+B_2u(k)+C_2u(k+1)+... \quad (3.11b)$$

which is the reduced recursive form. The vector $x_1(k)$ is the dynamic part, and $x_2(k)$ is the static part of the descriptor vector. The dimension of $x_1(k)$ is d, the degree of the system (see Section 3.3). Equation (3.11a) can be solved forward once $x_1(0)$ is specified, although values of future inputs may be required. Equation (3.11b) can be solved once $x_1(k)$ is known.

If the system is actually causal, then $C = D = ... = 0$ and (3.11) is a state vector representation of the system.

*Example 2.* Consider the E, A combination of Example 1, with input matirx

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

The row operations employed in Example 1 violate the rules that are used to obtain the reduced form, so the steps below follow a different path. The sequence of arrays is given without explanation.

| E | | | A | | | B | | C | | D | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | | | | | |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | | | | | |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | | | | | |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | | | | | |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | | | | | |
| 0 | 0 | 0 | -1 | 1 | 0 | 0 | -1 | | | | | |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | | | |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | | | |
| -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | | | |
| 0 | 0 | 0 | 1 | 0 | -1 | -1 | 1 | 0 | -1 | | | |
| -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 1 | |
| -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| -1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | -1 | |

This is the final stage, which in the last step has been brought to the special form (3.8).

The array is now back shuffled, and then brought to form (3.10).

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | | |
| 0 | 0 | 0 | -1 | 1 | 0 | 0 | -1 | 0 | 0 | | |
| 0 | 0 | 0 | -1 | 0 | 1 | 1 | -1 | 0 | 1 | | |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | -1 | | |
| 0 | 0 | 0 | -1 | 1 | 0 | 0 | -1 | 0 | 0 | | |
| 0 | 0 | 0 | -1 | 0 | 1 | 1 | -1 | 0 | 1 | | |

Thus the new representation is

$$x_1(k+1)=x_1(k)+u_2(k)-u_2(k+1)$$

$$x_2(k)=x_1(k)+u_2(k)$$

$$x_3(k)=x_1(k)-u_1(k)+u_2(k)-u_2(k+1).$$

## 3.5 CANONICAL DECOMPOSITION OF TIME-INVARIANT DESCRIPTOR SYSTEMS

One method of solving a properly conditioned and solvable linear descriptor system is to sweep the initial conditions forward and the final conditions backward through the time stages. This sweeping process creates a forward condition system and backward condition system. These condition systems have a (forward or backward) state-space representation and are useful analytical tools in addition to solving for the unknown descriptor vectors. Unfortunately, the structure of these condition systems varies with different initial and final conditions, and may be time-varying even for time-invariant descriptor systems. However, any time-invariant descriptor system has some set of initial and final conditions for which the corresponding condition systems are time-invariant. This motivates the examination of descriptor system decomposition that is not generated by particular initial and final conditions.

Consider the canonical structure of a solvable time-invariant system discussed in Section 3.3. Using an appropriate change of variables, such a system can be decomposed into subspaces $Lx(k)$, $M_1x(k),...,M_mx(k)$ with a state-space subsystem

$$Lx(k+1) = CLx(k) + Pu(k), \quad (3.12)$$

and a set of pure predictors of the form

$$M_ix(k) = H^{(r_i)}M_ix(k+1) - Q_iu(k), \quad (3.13)$$

where $H^{(r_i)}$ is the $r_i \times r_i$ matrix with ones on the upper diagonal and zeroes on the main diagonal. By using elementary row and column operations to convert the matrix C to Jordan canonical form, the subsystem (3.12) can be decomposed into a set of independent subsystems of the form

$$L_ix(k+1) = (\alpha_iI + H^{(q_i)})L_ix(k) + P_iu(k) \quad (3.14)$$

where $H^{(q_i)}$ has the same form as above.

Since no subsystem of the form (3.13) or (3.14) can be further decomposed, this operation characterizes a maximal decomposition of a solvable linear descriptor system. Note that subsystems of the form (3.14) with $\alpha_i = 0$ have the same form as the pure predictor subsystems, except these subsystems propagate forward in time. These subsystems are important in choosing initial and final conditions and are called *pure delay* subsystems. Also, the parameter $\alpha_i$ in (3.14) is the only eigenvalue of the subsystem, with geometric multiplicity $q_i$. These (forward) eigenvalues comprise the set of scalar solutions to the matrix equation

$$\det [A - sE] = 0,$$

and are sometimes called the generalized eigenvalues.

As an illustration of this decomposition, consider the example placed in shuffle form in the previous section. This system can be decomposed into a first-order subsystem of the form (3.14)

$$x_2(k+1) = x_2(k)$$

and a second-order pure predictor subsystem

$$x_3(k) - x_2(k) = x_1(k+1) - x_2(k+1) - u_1(k)$$

$$x_1(k) - x_2(k) = u_2(k)$$

or equivalently,

$$x_3(k) = x_2(k) + u_2(k+1) - u_1(k)$$

$$x_1(k) = x_2(k) + u_2(k)$$

From this representation, it is clear that one way of conditioning the solution is to specify $x_1(N)$, $x_3(N)$, and either $x_2(0)$ or $x_2(N)$. The components $x_1(k)$ and $x_3(k)$ are controllable (except at stage $N$), while the $x_2(k)$ component is obviously uncontrollable. The eigenvalue of the only forward subsystem is one and therefore the system is only marginally stable.

The above example suggests that the canonical decomposition of linear time-invariant descriptor systems can aid to valid conditioning of the solution and to system analysis. Suppose the issue of concern is the controllability of the system over a sufficiently long time horizon. If one of the subsystems of form (3.14) is not controllable, the invariant subspace independently governed by the subsystem will not be controllable by the overall system. Likewise, the pure predictor subsystems must be reverse-time controllable if the overall system is to be controllable. This provides a necessary condition for controllability that can be checked a subsystem level following a canonical decomposition. (Note that controlling a descriptor vector at a particular time stage may involve manipulating inputs at future time increments if any pure predictor system is greater than first order).

In the case of zero-input stability, a necessary and sufficient condition is that the state-space subsystem (3.12) be stable. In other words, the eigenvalues $\alpha_i$ corresponding to each subsystem of the type (3.14) must fall inside the unit circle. The pure predictor subsystems do not affect zero-input stability since the variables in these subsystems are determined entirely by inputs over an infinite time horizon.

If a time-invariant descriptor system is to be aggregated in a way that preserves the dominant dynamic modes, the aggregation can be efficiently accomplished using the canonical decomposition. A smaller model can be created by simply extracting the forward state-space subsystems having the dominant eigenvalues. This procedure does not generally eliminate a subset of original descriptor variables from the system but rather projects the system onto a lower dimensional linear manifold of the original vector space. However, if the intent is to remove some of the original variables from the model, the decomposition will reveal all static identities embedded in the pure delay and pure predictor subsystems.

## 3.6 BOUNDARY CONDITIONS FOR TIME-INVARIANT LINEAR DESCRIPTOR SYSTEMS

The actual solution of a solvable time-invariant system requires the specification of n conditions, which are often given as initial and final conditions. As noted earlier, these conditions cannot be specified arbitrarily. One method of checking whether a set of conditions is valid and sufficient is to actually try to solve the system by a double sweep. However, such a procedure is extremely inefficient. The canonical decomposition accommodates a more elegant characterization of proper boundary conditions.

If the aim is to find any set of boundary conditions that is valid and sufficient, the canonical form comprised of subsystems of types (3.13) and (3.14) suggests a natural choice. That is, for the forward state-space subsystems (3.14), specify the initial location of the corresponding subspaces, and for the pure predictor subsystems, select the final location in the corresponding subspaces. Since each subsystem is a forward or backward state-space system, the conditions for each subsystem propagate directly in the invariant subspaces, and the original descriptor variable values are uniquely determined.

In some cases, a candidate set of boundary conditions may already be given, and the concern is whether those conditions constitute a proper set. A necessary and sufficient test can be derived using the decomposition of a time-invariant descriptor system into a forward state-space subsystem and a backward state-space subsystem. For example, (3.12) can be the forward system and the combination of all pure predictor subsystems, represented by

$$Mx(k) = SMx(k+1) - Qu(k) \qquad (3.15)$$

can be the backward system. The test is derived as follows:

*Theorem 3.3:* A set of boundary conditions

$$\Gamma_o x(0) = y_o$$

$$\Lambda_N x(n) = z_N$$

is a valid set of boundary conditions to a system decomposed into (3.12) and (3.15) if and only if

$$\det \begin{bmatrix} \Gamma_o & & \\ -R^N L & L & \\ M & -S^N M & \\ & & \Lambda_N \end{bmatrix} \neq 0 \qquad (3.16)$$

where N is the number of time stages.

*Proof:* Since each subsystem is a state-space system, it is easily established that

$$Lx(N) - R^N Lx(0) = \sum_{i=0}^{N-1} R^{N-i-1} Pu(i) \qquad (3.17)$$

$$Mx(0) - S^N Mx(N) = -\sum_{i=0}^{N-1} S^i Qu(i) \qquad (3.18)$$

These relationships indicate the subspace of x(0) and x(N) that is controlled by the system inputs. The n remaining dimensions of the space of x(0) and x(N) can and must be determined by the boundary conditions. In order to accomplish this, the conditions must be linearly independent of each other and of the subspace determined by (3.17) and (3.18). A necessary and sufficient characterization of this independence property is given by (3.16). ■

The number of boundary conditions that can be on x(0) and the number that can be on x(N) are governed by the orders of the pure delay subsystems and pure predictor subsystems that result from the canonical decomposition. By observing the special property of the system matrices $H^{(q_i)}$ and $H^{(r_j)}$ of the pure delay and pure predictor subsystems, bounds on the number of initial and final conditions can be established:

*Theorem 3.4:* Let I be the set of indices corresponding to pure delay subsystems and J be the set of indices corresponding to pure predictor subsystems for a canonical decomposition of a time-invariant descriptor system. If N is the number of time stages, then there must be at least

$$\sum_{i\in I} \min [q_i, N]$$

initial conditions and

$$\sum_{j\in J} \min [r_j, N]$$

final conditions, where $q_i$ is the order of the respective pure delay subsystem and $r_j$ is the order of the respective pure predictor subsystem.

*Proof:* Assume that the system matrices R and S used in Theorem 3.3 have been transformed to block diagonal form using appropriate row and column operations. Each block of R corresponding to a pure delay subsystem will equal $H^{(q_i)}$ and hence the the same block of $R^k$ will equal $\left[H^{(q_i)}\right]^k$. It is easily verified that the matrix $H^{(q_i)}$ has the property that

$$\text{rank } \left[H^{(q_i)}\right]^k = q_i - \min [q_i, k] , \quad k \geq 0$$

Therefore, the rank of $R^k$ cannot exceed

$$\text{rank } L - \sum_{i\in I} \min [q_i, k]$$

Now for the matrix in (3.16) of Theorem 3.3, the full rank property will hold only if the submatrix

$$\begin{bmatrix} \Gamma_o \\ -R^N L \\ M \end{bmatrix}$$

has rank n. From the above observation, the matrix

$$\begin{bmatrix} -R^N L \\ M \end{bmatrix}$$

cannot have a rank exceeding

$$n - \sum_{i\in I} \min [q_i, N].$$

Hence, the lower bound on the rank of $\Gamma_o$, or the number of initial conditions, will hold. A similar argument establishes the lower bound on the number of final conditions. ■

When the number of time stages exceeds the order of any pure delay or pure predictor subsystem, the boundary conditions must be specified in a way that determines the location of x(0) in the subspaces corresponding to pure delay subsystems and the location of x(N) in subspaces corresponding to pure predictor subsystems. However, it is not necessary that the conditions be formulated purely in terms of those subspaces.

The subsystems that are not pure delays or pure predictors have the property of being able to

operate like a state-space system either forward or backward through the time stages. For this reason, the choice of the number of initial and final conditions pertaining to these subsystems is arbitrary for any number of time stages, although the actual structuring of those conditions is *not* arbitrary. Therefore, the general lower bounds indicated by Theorem 3.4 are as large as possible.

If the system has an infinite time horizon, the specification of initial conditions for all subsystems of type (3.14) is necessary and sufficient. The conditions on subspaces corresponding to the pure predictor subsystems can be neglected. The reason for this is that the final conditions on any pure predictor subsystem will affect, at most, the final $r_j$ time stages, due to the nilpotent subsystem matrix $H_j^{(r_j)}$. Since there is no explicit final stage with an infinite time horizon the system can be solved without these conditions via the shuffle algorithm.

# IV. NONLINEAR DESCRIPTOR SYSTEMS

## 4.1 INTRODUCTION

The general structural form for models of systems in descriptor form is given by a set of equations

$$\begin{aligned} &g_0(x(0),x(1),u(0)) = 0 \\ &g_1(x(1),x(2),u(1)) = 0 \\ &\quad\vdots \\ &g_{N-1}(x(N-1),x(N),u(N-1)) = 0 \end{aligned} \tag{4.1}$$

where

- $x(k)$ is an n-dimensional descriptor vector for each $k = 0,1,2,\ldots,N$
- $u(k)$ is an m-dimensional input vector for each $k = 0,1,2,\ldots,N-1$
- $g_k$ is a function taking values in n-dimensional space.

A fairly complete theory for these systems has been outlined in the linear case. The intent of this chapter is to show that there is a satisfactory theory for nonlinear descriptor systems.

For comparison, one can consider the analogous state-vector system

$$x(k+1) = h_k(x(k),u(k)) \tag{4.2}$$

for $k = 0,1,2,\ldots,N-1$. Assuming only that $h_k$ is sell-defined, there is a unique solution $x(k)$ to (4.2) corresponding to each set of initial conditions and input sequence. Furthermore, this solution can be found recursively, progressing sequentially from $k = 0$ to $k = N$. For the more general descriptor variable framework (4.1) the situation is known to be far more complex: there may not be a solution; if there is it may not be uniquely specified in terms of boundary conditions; and the formation of recursive solutions is a difficult problem. Nevertheless, these issues all can be resolved quite satisfactorily. This chapter shows that it is possible to extend several standard concepts and procedures, including: (1) the state-transition function, (2) forward propagation of initial conditions, and (3) forward recursion when a complete set of initial conditions is specified. These generalized results provide a broad framework for addressing descriptor variable systems.

The importance of the general descriptor framework has been long recognized in connection with differential equations. General theories concerned with these structures, however, have almost exclusively focused on the linear time-invariant case. The most significant contribution in this area is canonical form of a matrix pencil, discussed in Section 3.3. This result still underlies much modern work.

Most general theories, including those based on the theory of matrix pencils, have three severe limitations. First, as in the case of canonical forms, the results often require a change of variables, which is undesirable in terms of relating specific results to the original problem context. Second, although many of these theories illuminate internal structural relations, they give little attention to the formation of recursive solutions. Third, these theories are strongly wedded to the assumptions of linearity and time-invariance, precluding their extension to more general situations. Although the theory of descriptor variables presented here overlaps previous theories in the linear time-invariant case, it is unique in that it does generalize quite naturally to nonlinear time-varying systems.

A theory involving nonlinear equations, such as the one proposed here, can be presented within various analytical formats. It is, of course, clear at the outset that the required computational procedures are likely to be cumbersome; but it does not necessarily follow that the theory must be equally cumbersome. It is quite possible to develop an elegant theoretical structure. On the other hand, to be useful, a theory must be closely related to computational procedures. Such considerations have motivated the choice of format selected here. The results are presented within the framework of manifold theory and differential topology. It must be emphasized that this selection is primarily a choice of viewpoint, rather than of technique. The *viewpoint* of differential topology allows one to translate essentially local analytical results (such as those stemming from the implicit function theorem) to global geometric relationships. Presented in this framework, each piece of the theory of nonlinear descriptor systems has both global geometric interpretations and algebraic (or computational) implications. Thus the chapter simultaneously unfolds two distinct but interrelated developments: the geometric (which is in some sense conceptually cleanest) and the algebraic (which is perhaps most relevant for practice).

## 4.2 SOLVABILITY AND CONDITIONABILITY

The objective of this section and the next two is to characterize the structure of the solution set of a descriptor system, and initiate how specific solutions might be computed. For this purpose the role of the input sequence $u(k)$, $k = 0,1,\ldots,N-1$, is somewhat incidental. It is sufficient to consider the system without input

$$f_0(x(0),x(1)) = 0$$

$$f_1(x(1),x(2)) = 0$$

$$\vdots \qquad (4.3)$$

$$f_{N-1}(x(N-1),x(N)) = 0$$

A particular input sequence in (4.1) merely defines a particular set of the form (4.3).

Throughout the investigation the functions $f_k, k = 0,1,2,\ldots,N-1$ are assumed to be continuously differentiable with respect to all variables, at least in some open D to which attention is confined. Structural assumptions are often expressed as assumptions on the derivatives.

*Solvability*

Define M as a set of solutions to (4.3) in the domain D. M is a subset (possible empty) of $R^{n(N+1)}$. In general, M may be a quite complicated set. We formulate below a simple requirement (one that is standard in the studies of nonlinear equations) that guarantees that M is actually a manifold.

For any solution $x(0),x(1),x(2),\ldots,x(N)$ to equation (4.3) define the derivatives matrix

$$F(0,N,x) = \begin{bmatrix} \frac{\partial f_0}{\partial x(0)} & \frac{\partial f_0}{\partial x(1)} & & & \\ & \frac{\partial f_1}{\partial x(1)} & \frac{\partial f_1}{\partial x(2)} & & \\ & & \ddots & & \\ & & & \frac{\partial f_{N-1}}{\partial x(N-1)} & \frac{\partial f_{N-1}}{\partial x(N)} \end{bmatrix}$$

The matrix F is essentially the coefficient matrix of a linearized version of (4.3). This matrix is defined in terms of n x n blocks. Indeed the matrix F has N (block) rows and N + 1 (block) columns. The maximum possible rank is equal to the number of rows, nN.

*Definition:* The system (4.3) is said to be *solvable* if M is not empty, and if for every point in M the matrix F(0,N,x) has full rank.

In algebraic terms, the assumption of solvability implies that there are n degrees of freedom in the solution of (4.3), since there are n more variables than equations. The geometric significance of the assumption of solvability is stated by the following theorem* (see for example [19] for this standard result).

*Theorem 4.1:* If the system (4.3) is solvable, then M is an n-dimensional manifold.

Much of the ensuing work is directed at further characterizing the manifold M, and imposing additional assumptions so that this manifold has certain desirable properties.

*Conditionability*

If the system of equations (4.3) is solvable, its solution is not unique. To define a unique solution, n additional equations, or conditions, must be imposed. In general, suitable additional equations may take a variety of forms, involving variables at various time points. From our underlying perspective of dynamic systems, however, it is natural to think in terms of end-point conditions (that is, conditions specified only in terms of x(0) and x(N)). Special cases are pure *initial* conditions, involving only x(0), and pure *final* conditions involving only x(N). For an arbitrary system of the form (4.3), however, such end-point conditioning is not always possible. It may be that the degrees of freedom in the solution are restricted to descriptor variables at certain intermediate points, with the end-points having less than full flexibility. Such systems are in some sense dynamically degenerate, and are of little real interest for our purpose. This is the algebraic motivation for the concept of *conditionability*.

*Definition:* A system of the form (4.3) is said to be *conditionable* if

(1) the matrix

$$G(0,N,x) = \begin{bmatrix} \frac{\partial f_0}{x(1)} & & & \\ \frac{\partial f_1}{\partial x(1)} & \frac{\partial f_1}{\partial x(2)} & & \\ & \ddots & & \\ & & & \frac{\partial f_{N-1}}{\partial x(N-1)} \end{bmatrix}$$

is of full rank for all $x \in M$.

(2) No two solutions have identical end-points x(0), x(N).

(3) Any unbounded sequence in M entails an unbounded sequence of end-points.

*The set M is an n-dimensional manifold if it is locally diffeomorphic to $R^n$. That is, each point $m \in M$ possesses a neighborhood V in M which is diffeomorphic to an open set U in $R^n$. A diffeomorphism $\phi: U \to V$ is a *parameterization* of V. The inverse diffeomorphism $\phi: V \to U$ is a *coordinate system* on V. Thus, more loosely, M is an n-dimensional manifold if at every point an n-dimensional coordinate system can be constructed in a neighborhood.

The first requirement is the basic local requirement for conditionability. The other two requirements insure suitable global properties as explained later in this section.

Let us focus on the first requirement. Note that the matrix G(0,N,x) is a submatrix of the matrix F(0,N,x). It is a rectangular matrix having n more rows than columns. One interpretation of the requirement on rank is obtained by recalling that F(0,N,x) represents the coefficient matrix of a linearized version of (4.3). If x(0) and x(N) are fixed, the matrix G(0,N,x) represents the coefficient matrix of the equations that must be satisfied by variations in the other descriptor variables. If this matrix is of full rank, these intermediate variables are uniquely determined. Thus, the rank requirement is a linear condition implying that (at least locally) no two solutions have identical end-points. Thus, this requirement is consistent with the algebraic motivation that all degrees of freedom should be reflected in the end-points.

Now let us consider the geometric motivation for conditionability. Assuming that the system (4.3) is solvable, M is an n-dimensional manifold in the space $R^{n(N+1)}$ of descriptor variables. This is not, however, a very economical description for the solution set, since it is defined in the (relatively large dimensional) space $R^{n(N+1)}$. It seems appropriate to seek a representation of this n-dimensional manifold M within some space of dimension much lower than n(N+1). One obvious approach at simplification is to consider the projection of M into various subspaces of $R^{n(N+1)}$. In general, however, such projections are not n-dimensional, and indeed not even manifolds. (For example, the surface of a sphere is a 2-dimensional manifold in $R^3$; but its projection on a plane is a closed disk, which is not a manifold.) In particular, the projection of M onto the n-dimensional subspace corresponding to a descriptor variable x(k) for a fixed k will rarely be an n-dimensional manifold (for then it would equal the entire subspace). There is, however, a clean solution to the representation problem if the assumption of conditionability is introduced.

Consider the 2n-dimensional subspace of $R^{n(N+1)}$ corresponding to the first n and last n coordinates; that is to the coordinates associated with x(0) and x(N). Let B denote the projection of M on this subspace. The set B is the set of possible boundary points of solutions to (4.3).

*Theorem 4.2:* If the set of equations (4.3) is solvable and conditionable, then B, the projection of M into the 2n-dimensional space of end-points, is an n-dimensional manifold. In fact B represents an embedding of M.

*Proof:* Let x be a point in M. By solvability, the matrix F(0,N,x) has rank nN. By conditionability, the n(N-1) middle columns of this matrix are linearly independent. Therefore, n additional columns can be selected from among the first n and last n columns of F(0,N,x) to form a total of nN independent columns. The n variables corresponding to columns not selected can serve as a (local) basis for M. This follows from the implicit function theorem, since these variables can be varied arbitrarily (locally) to determine an overall solution to (4.3). These basis variables are a subset of the end-point descriptor variables. Such a basis exists at every point x. This shows that the projection operation from M onto B is a local immersion.

The second requirement (2) guarantees that the projection is one-to-one. Finally, the third requirement (3) guarantees that the immersion is *proper* (that is, the preimage of compact sets are compact). Thus the projection is an embedding [19]. ■

In view of this result, the set B is called the *boundary manifold* of the system (4.3). One interpretation of it is that M can be projected down to B without loss in information. As a side comment, one might compare this result with Whitney's Theorem, which states that any n-dimensional manifold can be embedded in $R^{2n}$. Thus Theorem 4.2 represents the best that might be hoped for in terms of the most economical representation of M. Finally, it should be pointed out that the projection of M onto the 2n-dimensional subspace generated by descriptor variables at two other time points need not yield an n-dimensional manifold. Only the end-point projection will work.

*Subsystems*

The solution procedures developed in the next two sections exploit the dynamic structure inherent in well-behaved descriptor systems. This structure is expressed in terms of a nested family of subsystems of the original descriptor system. In order to develop this line of reasoning, the notions of solvability and conditionability must be suitably extended to subsystems.

The original system (4.3) is defined on the time points k = 0,1,2,...,N. A subsystem is obtained by deleting some of the original equations, leaving a subset of equations corresponding to a contiguous subset of time points. We let $S_{i,j}$ be the subsystem defined over the integers k = i, i + 1,...,j. Thus in this notation the original system is $S_{0,N}$. Likewise, we denote by $M_{i,j}$ and $B_{i,j}$ the set of solutions and set of end-points corresponding to the system $S_{i,j}$. Again, we have $M = M_{0,N}$ and $B = B_{0,N}$.

We shall primarily focus attention on the special subsystems $S_{0,N}$, $S_{1,N}$,...,$S_{N-1,N}$. We say that the original system S is *forward solvable* and *forward conditionable* if each of these subsystems are solvable and conditionable, respectively. (Clearly there is an analogous definition of backward solvability and backward conditionability.)

In some sense the requirements of forward solvability and forward conditionability are not

really additional requirements. Rather they simply extend the original requirements to larger sets. To see this, let $\tilde{M}_{i,N}$ be the projection of $\tilde{M}$ onto the subspace containing the variables x(i), x(i+1),...,x(N). Since any solution of the original system induces a solution on the subsystem, it follows that $\tilde{M}_{i,N} \subset M_{i,N}$. Solvability of the original system implies that F(0,N,x) is of full rank for $x \in \tilde{M}$. Any subset of the rows of F is also of full rank, so F(0,N,x) is of full rank for all $x \in \tilde{M}_{i,N}$. Likewise conditionability of the original system implies that the conditionability rank condition holds on $\tilde{M}_{i,N}$. The assumptions of forward solvability and forward conditionability extend the rank requirements from $\tilde{M}_{i,i}$ to $M_{i,i}$. (In the case of linear systems, the above discussion shows that solvability or conditionability of the original system implies forward solvability, or forward conditionability, respectively, of any subsystem.)

## 4.3 CONSTRUCTION OF BOUNDARY MANIFOLDS

We have seen that the solution manifold of a solvable and conditionable descriptor system can be embedded on the boundary manifold B. If the system is forward solvable and forward conditionable, the solution manifolds of each of the forward subsystems can be similarly embedded on a corresponding boundary manifold. The successive boundary manifolds are, of course, related. By exploiting this relationship, it is possible to construct the boundary manifolds recursively, starting with the simplest. Thus, we begin by constructing $B_{N-1,N}$, which is the boundary manifold for the simplest subsystem. From this we construct $B_{N-2,N}$, etc., working all the way to $B_{0,N}$.

The specific construction is, by necessity, local in character. With this caveat in mind, we shall nevertheless represent boundary manifolds in simple implicit form. Specifically, we represent the boundary manifold $B_{k,N}$ by the system of equations

$$\Phi_k(x_k, x_N) = 0 \tag{4.4}$$

where $\Phi_k$ is an n-dimensional function of full rank on $B_{k,N}$. The functions $\Phi_k$ are generalizations of the state transition functions in ordinary state-space theory. Equation (4.4) is completely equivalent (locally) to the last N-k equations in (4.3), or equivalently, to $S_{k,N}$. Solutions to (4.4) are in direct correspondence to end-points of $S_{k,N}$.

To justify the construction it is necessary to consider a lemma which applies in the linear case.

*Lemma:* Consider the system

$$Ax_{k-1} + Bx_k = 0$$

$$Cx_k + Dx_N = 0$$

where A, B, C, D are n x n matrices. Suppose this system is of full rank on $R^{3n}$ and of full rank with respect to $x_k$. Then these equations can be linearly combined to yield a relation of the form

$$VAx_{k-1} + WDx_N = 0$$

which is of rank n on $R^{2n}$.

*Proof:* The matrix $\begin{bmatrix} B \\ C \end{bmatrix}$ has at most rank n, hence there are n x n matrices V and W such that [V,W] is of full rank and VB + WC = 0. Combining the original equations this way gives the result. ■

We now turn to the details of the general construction process. We assume that the system is forward solvable and conditionable. The procedure is:

Let

$$\Phi_{N-1}(x_{N-1}, x_N) = f_{N-1}(x_{N-1}, x_N)$$

Clearly $\Phi_{N-1}(x_{N-1}, x_N) = 0$ is an implicit representation for $B_{N-1,N}$. In general, suppose that $\Phi_k(x_k, x_N) = 0$ is a suitable representation for $B_{k,N}$. $\Phi_k$ is of dimension n and has full rank.

Now consider the equations

$$\begin{aligned} \Phi_k(x_k, x_N) &= 0 \\ f_{k-1}(x_{k-1}, x_k) &= 0 \end{aligned} \tag{4.5}$$

As verified below, this system is of full rank at any solution to $S_{k-1,N}$. It is also of rank n with respect to $x_k$. Thus, $x_k$ can be eliminated yielding the n-dimensional relation

$$\Phi_{k-1}(x_{k-1}, x_N) = 0.$$

This $\Phi_{k-1}$ will have rank n.

To verify the statements concerning rank, imagine that all operations are carried out on the linearized system. In this case each $\Phi_k$ is a linear combination of the $f_i$'s i=k,k+1,...,N, as indicated by the lemma. The fact that (4.5) is of full rank follows from forward solvability, since this system is essentially a representation for the n-dimensional solution manifold $M_{k-1,N}$. Then since the derivatives of the nonlinear $\Phi_k$ at the nominal solution is equal to that of the linear version, the rank result holds. Likewise, the fact that (4.5) is of full rank with respect to $x_k$ follows from forward conditionability. If $x_{k-1}$ and $x_N$ are given it must be possible to solve (4.5) (and its linearized version) uniquely for $x_k$.

As mentioned earlier, the functions $\Phi_k$, k=0,1,2,...,N-1 representing the boundary manifolds are generalizations of the state transition functions in ordinary state space theory. Indeed, for a system described by x(k+1) = Ax(k) one can easily calculate that $\Phi_0(x(0),x(N)) \equiv x(N) - A^N x(0)$.

*A Method of Solution*

Once the $\Phi_k$'s are known, it is possible to

calculate solutions to the original system recursively. The equation $\Phi_0(x(0),x(N)) = 0$ defines the boundary manifold for the entire system $S_{0,N}$. In order to specify a single solution to the system, a single point on this n-dimensional boundary manifold must be specified. Generally this specification takes the form of n additional relations, independent of those in the $\Phi_0$. These additional relations specify a unique point on the boundary manifold and hence a unique x(0),x(N).

Next consider the equations

$$f_0(x(0),x(1)) = 0$$

$$\Phi_1(x(1),x(N)) = 0$$

This system of equations has full rank with respect to x(1), and x(0) and x(N) are known from the initial stage of the solution process. Thus, this system can be solved for x(1).

This value of x(1) and the value of x(N) are then used in the system

$$f_1(x(1), x(2)) = 0$$

$$\Phi_2(x(2),x(N)) = 0$$

to determine x(2), etc. In this way the entire sequence of descriptor vectors is determined recursively.

The procedure for the (backward) recursive determination of the $\Phi_k$'s followed by the (forward) recursive determination of a specific solution has several potential implementations. In some situations the (global) operations can be carried out directly. (This is often true if the system itself is linear or if a dynamic programming procedure is employed.) In other cases, however, only linear approximations to the $\Phi_k$'s are calculated, leading to a solution procedure which involves successively sweeping back and forth through the system until convergence is achieved.

## 4.4 CONDITION VECTORS

A unique solution of a solvable and conditionable descriptor system is determined by the specification of a point on the boundary manifold B. As pointed out earlier, this point is usually specified in the form of n additional relations or *conditions*. These might have the form

$$\Gamma_0(x(0), x(N)) = \alpha_0 \qquad (4.6)$$

where the function $\Gamma_0$ and the vector $\alpha_0$ are n-dimensional. If the functions $\Gamma_0$ are independent of other system relations, the relation (4.6) determines various points on the boundary manifold as $\alpha_0$ varies. Thus $\alpha_0$ serves to parameterize the boundary manifold and, corresponding, the solution manifold as well.

More explicitly, we know from the previous section that (at least locally) the boundary manifold $B_{0,N}$ can be represented in implicit form $\Phi_0(x(0),x(N)) = 0$, where $\Phi_0$ is a n-dimensional function of rank n. If $\Gamma_0(x(0),x(N))$ is an n-dimensional function of rank n that is independent of $\Phi_0$ (locally), then the system

$$\Gamma_0(x(0), x(N)) = \alpha_0$$

$$\Phi_0(x(0),x(N)) = 0 \qquad (4.7)$$

determines a unique x(0),x(N). An equivalent statement, not requiring the knowledge of $\Phi_0$, is that (4.6) appended to the system (4.3) uniquely specifies a solution. The function $\Gamma_0$ are termed *conditions* and parameterization vector $\alpha_0$ is termed *condition vector*. In terms of manifold theory, $\Gamma_0$ defines a coordinate system for $B_{0,N}$.

### *Propagation of Conditions*

We consider now a solution procedure, termed the *double-sweep method*, that is in some sense the dual to the procedure discussed in the previous section. This procedure more closely resembles the standard recursive procedure used for solving ordinary dynamic systems. (Indeed, in the case of a state space system this procedure is equivalent to normal forward recursion of the state vector.) It must be pointed out, however, that although the procedure is in many senses a more natural one, it has potential hazards (as is discussed below under the heading of catastrophes).

The propagation of condition vectors is again based on consideration of the nested family of descriptor systems $S_{0,N}$, $S_{1,N},\ldots,S_{N-1,N}$. In this case, however, the initial propagation moves forward, working from the full system to progressively smaller subsystems.

One starts with the system $S_{0,N}$ and a suitable set of conditions $\Gamma_0(x(0),x(N)) = \alpha_0$. This specifies a unique solution to the full system $S_{0,N}$. By deleting the first term, x(0), the resulting sequence of descriptor vectors is a solution to the subsystem $S_{1,N}$. Thus, there must be a set of consistent conditions for the subsystem $S_{1,N}$ that yield that reduced solution. This new set of conditions represents a propagation of the original conditions on the full system to the subsystem. This procedure is continued over successively smaller subsystems. The procedure thus generates a sequence of condition functions $\Gamma_k$, k=0,1,2,..., N-1, and an associated sequence condition vectors $\alpha_0, \alpha_1, \alpha_2,\ldots, \alpha_{N-1}$.

The procedure begins with a suitable n-dimensional constraint of the form $\Gamma_0(x(0),x(N)) = \alpha_0$, having rank n. This is swept forward one step by consideration of the system of equations

$$\Gamma_0(x(0),x(N)) = \alpha_0$$

$$f_0(x(0),x(1)) = 0$$

This system must be of rank n with respect to x(0), for if x(1) and x(N) are known x(0) must be determined from these equations. Hence, with respect to x(0) there are at least n functional dependencies among the functions $\Gamma_0$, $f_0$. That is,

there is an n-dimensional function $\psi_1$ of rank n such that $\psi_1(\Gamma_0(x(0),x(N)), f_0(x(0),x(1)))$ is independent of x(0). This function is equal to some function $\Gamma_1(x(1), x(N))$. We thus have

$$\Gamma_1(x(1),x(N)) = \psi_1(\Gamma_0(x(0),x(n),f_0(x(0),x(1)))$$
$$= \psi_1(\alpha_0,0)$$

In other words,

$$\Gamma_1(x(1),x(N)) = \alpha_1 \qquad (4.8)$$

where

$$\alpha_1 = \psi_1(\alpha_0,0) .$$

This is the equivalent set of conditions for the subsystem $S_{1,N}$. As in the preceeding section a check of the linear case verifies that all necessary rand conditions are satisfied by this updated procedure.

The procedure is continued to determine, in general, $\psi_k$, $\Gamma_k$, and $\alpha_k$ for k=0,1,2,...,N-1. At the final point the 2n equations

$$\Gamma_{N-1}(x(N-1), x(N)) = \alpha_{N-1}$$
$$f_{N-1}(x(N-1), x(N)) = 0$$

can be solved for x(N-1) and x(N). At this point one may work backward through the system to solve for the successive descriptor vectors. Specifically, given x(k+1) (and x(N)) one solves the system

$$\Gamma_k(x(k),x(N)) = \alpha_k$$
$$f_k(x(k),x(k+1)) = 0$$

for x(k).

*Example:* Consider the linear state-space system x(k+1) = Ax(k). We may take $\Gamma_0 = x(0)$. Thus $x(0) = \alpha_0$ is the initial condition. The vector x(0) can be eliminated from the equations

$$x(0) = \alpha_0$$
$$x(1) = Ax(0)$$

to produce

$$x(1) = \alpha_1$$
$$\alpha_1 = A\alpha_0$$

which is (4.8) for this special case. This implies $\Gamma_1 = x(1)$. Thus this procedure reduces to normal state recursion.

*Catastrophes*

The procedure outlined above works well, provided that the initial n-dimensional function $\Gamma_0$ does in fact represent a legitimate coordinate system on the boundary manifold $B_{0,N}$ (or equivalently, that the functions $\Gamma_0$ are independent of the system equations). For an arbitrary function $\Gamma_0$, the procedure outlined above may progress successfully for several steps until a point is reached where the equations

$$\Gamma_k(x(k), x(N)) = \alpha_k$$
$$f_k(x(k), x(k+1)) = 0$$

are inconsistent. At that point, no further progress is possible.

Such a break-down results from improper selection of the original $\Gamma_0$, but unfortunately there is no way to insure that a given $\Gamma_0$ is suitable by inspection of the properties of the system near the initial point k = 0. Whether the given $\Gamma_0$ is a suitable condition coordinate system is dependent on the entire structure of the system over the entire time interval. In general, a procedure equivalent to that in Section 4.3 to calculate $\phi_0$, must be employed to insure that a given $\Gamma_0$ is appropriate.

A similar phenomenon occurs in continuous-time systems and essentially forms the basis for the subject of catastrophe theory. In the continuous-time case, however, it is possible to rationalize the sudden inconsistency by hypothesizing an instantaneous jump in the value of some descriptor variables. In the discrete-time case there seems to be no equivalent way out of the difficulty.

As an example consider the linear system

$$E_{k+1}x(k+1) = x(k)$$

Assume that $E_k$ is of full rank for k=0,1,2,3, but that $E_4$ is singular. The choice $\Gamma_0 = x(0)$ seems reasonable at first, for indeed $\Gamma_1 = x(1)$, $\Gamma_2 = x(2)$, and $\Gamma_3 = x(3)$ will follow naturally. However, there is a catastrophe at k = 4 since $E_4x(4) = x(3)$ is not independent of $x(3) = \alpha_3$. The situation can be rectified only by the selection of a different $\Gamma_0$, having less than full rank with respect to x(0).

These comments concerning catastrophes are not meant to imply that the double sweep method is seriously flawed. On the contrary, in practice suitable choices for $\Gamma_0$'s are often clear from the context of the problem. One must, however, be aware of the potential difficulties.

*Inputs*

At this point, let us briefly consider the original system in descriptor form with inputs (4.1). During the forward sweep of the procedure outlined in the first part of this section, it is necessary only to use the functions $h_k$ (and consequently the inputs u(k)) sequentially. Thus the forward sweep is a causal operation. Therefore propagation of the condition vector can be thought of in terms of a normal causal dynamic operation. In this sense the condition vector

acts much like the state vector of a (normal) dynamic system. The backward sweep uses the functions $h_k$ and the corresponding u(k)'s in the reverse order. Thus the overall procedure is noncausal, although the two portions forward and backward are each causal in their respective direction.

## 4.5 FORWARD RECURSION

It is sometimes possible to solve a descriptor variable system by a single forward sweep. In order that this be possible, two general properties must be present. First, the system must possess appropriate structure. Second, the specified end-point conditions must be such that a maximum number of initial conditions are specified. An example, of course, is a state-space system. Such a system has a structure that allows for solution by forward recursion, but such a recursion is possible only if, in addition, the n boundary conditions are all initial conditions. Similar requirements hold for general descriptor variable systems.

This section outlines the technique for forward recursion in the simple case of *regular* systems. The technique is actually a slight modification of the double sweep method developed in Section 4.4. In order to develop the method, we first explain the generalization of solvability and conditionability to systems with inputs, next we explore the role of initial conditions in descriptor systems, and finally we define regular systems.

### *General Framework*

It is appropriate at this point to show how the general descriptor system (4.1) can be embedded in the framework of Section 4.2 for systems without input (4.3). This extension is important in forward recursion techniques, since the order in which the u(k)'s are processed is important.

Define $\mathcal{M}$ as the set of solutions to (4.1) in some appropriate domain. $\mathcal{M}$ is a subset (possibly empty) of $R^{n(N+1)} \times R^{mN}$. The system (4.1) is *solvable* if $\mathcal{M}$ is not empty, and if for every point in $\mathcal{M}$ the Jacobian matrix of the system with respect to the descriptor variables has full rank. This assumption implies that $\mathcal{M}$ is an $(n + mN)$-dimensional manifold. Fixing a specific set of inputs corresponds to slicing through the $\mathcal{M}$ manifold and producing a manifold M as in Section 4.2. The notion of conditionability is generalized in a similar fashion.

In this generalization it is clear that the role of the inputs is secondary. However, when it is important to keep track of the role of inputs, this general framework is available.

### *Initial Conditions Always Exist*

The concept of conditions was discussed in Section 4.4. Conditions provide a (local) parameterization of the boundary manifold, and hence of the solution manifold as well. In general, one expects that suitable condition functions may involve both end-point vectors x(0) and x(N) simultaneously. However, the following result shows that it is always possible to specify a complete set of conditions in terms of pure initial and pure final conditions. This justifies special consideration of pure end-point conditions.

*Theorem 4.3:* If a descriptor system (4.1) is solvable and conditionable, then a complete set of additional conditions can be specified (locally) in terms of pure initial and pure final conditions.

*Proof:* In Section 2.3 this theorem was proved for the linear case. That result shows that the Jacobian matrix, with respect to x, of the system (4.1) can be augmented by n rows, each having entries corresponding either to initial or final conditions, to give full rank. Specifically, the system (4.1) when augmented by conditions of the form $x_i(0) = \alpha_i$, $i \in I_0$ and $x_j(N) = \beta_j$, $j \in I_N$ (where the $\alpha_i$ and $\beta_j$ are real numbers, and the index sets $I_0^i$ and $I_N^j$ together contain n elements) is of full rank with respect to x. Thus, by the inverse function theorem there is a unique solution for all values of $\alpha_i$ and $\alpha_j$ (at least locally). These solutions are on the solution manifold $\mathcal{M}$. ■

### *Regular Systems*

We now consider an important special case, where a descriptor variable system can be solved forward recursively in one sweep. Consider the system (4.1) which can be written as

$$g_k(x(k),\ x(k+1),\ u(k)) = 0 \qquad (4.9)$$

for k = 0,1,2,...,N-1. Suppose that for every k and throughout the rank of $g_k(x(k),x(k+1),u(k))$ is r with respect to x(k+1). Then by suitable manipulation the system (4.9) can be reexpressed (perhaps locally) as

$$c_k(x(k),\ x(k+1),\ u(k)) = 0 \qquad (4.10a)$$

$$d_k(x(k),\ u(k)) = 0 \qquad (4.10b)$$

The function $c_k(x(k),\ x(k+1),\ u(k))$ is of rank r with respect to x(k+1). Many systems are readily expressed in this form. Note in particular that if r = n then $c_k = g_k$, and the system can be solved forward recursively from a given initial x(0) - in fact, this is a state-space system. In general, of course, one has r < n. The state-space case, however, motivates the definition of a regular system.

We assume that the system (4.10) is solvable and conditionable. We further assume that a full set of pure initial and final conditions exist for (4.10), and that the number of initial conditions is equal to the dimension of the upper part of the system. Specifically, the conditions have the form $c_{-1}(x(0)) = \alpha_0$ and $d_N(x(N)) = \beta_N$, where $c_{-1}$ is r-dimensional of rank r with respect to x(0), and $d_N$ is (n-r)-dimensional of rank n-r with respect to x(N). Thus, a full set of pure initial and final conditions are assumed to exist

for (4.10), and the number of initial conditions is equal to the dimension of the upper part of the system.

*Definition:* The system (9) is *regular* if for each k = 0,1,2,...,N-1 the Jacobian with respect to x(k) of the functions

$$c_{k-1}(x(k-1),\ x(k),\ u(k-1))$$

$$d_k(x(k),\ u(k))$$

is of rank n throughout $\mathcal{M}$.

A system which satisfies this definition of regularity can be solved forward recursively with a modification of the double sweep algorithm. We sketch the details below.

First, consider the system

$$c_{-1}(x(0)) = \alpha_0$$

$$d_0(x(0),\ u(0)) = 0$$

By assumption this system is of full rank and hence can be solved for x(0). Second, consider the system

$$c_0(x(1),\ x(0),\ u(0)) = 0$$

$$d_1(x(1),\ x(0),\ u(1)) = 0$$

This system is of rank n with respect to x(1) and hence given x(0) from above and u(0), u(1), this system can be solved for x(1). One continues in this fashion all the way to x(N).

The above is an outline of the procedure. It is possible to trace the condition vector $\alpha_0$ through successive stages. As demonstrated in Section 2.4, the condition vector at any stage serves as a state vector. Regular systems lead to a simple forward recursion solution. More general systems can also be solved forward, in a similar manner, by employment of a generalization of the shuffle algorithm introduced in Section 3.4 when the system is time-invariant.

# V. CONTROL OF LINEAR DESCRIPTOR SYSTEMS

## 5.1 INTRODUCTION

In the earlier chapters of this report, the emphasis was on the uniqueness and conditionability of the solution to a descriptor system and the determination of that solution. The inputs to the system were treated as additional parameters that affect the location of the solution. In this chapter, the focus will shift to the determination of inputs that create desirable solutions to a descriptor system.

If there is a criterion function that assigns a numerical value characterizing the desirability, an optimization problem can be defined to determine the "best" set of inputs. Dynamic programming is an effective procedure for optimizing the subclass of descriptor systems that have state-space form. This procedure employs the property that the descriptor vector at each stage is a state. However, the procedure can be adapted to systems where there is no state vector at various stages. This revised method is presented in Section 5.2.

Many important dynamic models, however, are (forward-time) causal and have a state at each time instant. By having a state, the system can be solved via a forward sweep through the time instants, without requiring the knowledge of future inputs. In Chapter II, we identified regular systems as systems that can be characterized by a forward condition system plus static relationships at each time instant. This structure can be exploited in designing a more efficient dynamic programming than the one described in Section 5.2. This second procedure is outlined in Section 5.3.

The performance criteria for dynamic economic systems can often be approximated by criterion functions that are quadratic in terms of descriptor variables and inputs and that are additively separable by time increment. Linear state-space systems with such performance criteria have been thoroughly studied, and it is well-known how to compute an optimal feedback control law. Section 5.3 demonstrates that this result extends to any linear, regular system using the special causal structure of such systems. Therefore, a procedure exists for computing an optimal feedback policy without requiring a state-space representation.

Frequently in models, it is useful to specify desired behavior in the form of additional static relationships. One then has a rectangular system, as opposed to a square system, because there will be more relationships than unknowns. In some cases, this representation may be inconsistent and the model is not realizable. However, usually these relationships will have a feasible solution set for a restricted set of inputs. The determination of optimal control in these models is somewhat more complex than the simple unconstrained solvable representation. This chapter will first consider rectangular systems that are regular - the approach will be to separate one set of relationships that form a square, regular system and designate the other relationships as a set of static constraints. The critical issue is whether these constraints can be *maintained* as the unconstrained system progresses in time. This concept of maintainability is carefully described in Section 5.4.

Given that a regular rectangular model for a system satisfies maintainability, it can be demonstrated how optimal feedback policies can be computed when one has quadratic performance criteria. Section 5.5 first considers the special case of a system represented by a state-space model plus static relationships on the state variables. The result derived for this special case is then extended to the more general case where the state-space model is replaced by a square, regular system and where the static constraints can include the inputs.

Often there is not a criterion for evaluating the optimal policy, yet it is desirable to apply inputs in a manner that keeps descriptor variables inside acceptable ranges. As in rectangular descriptor systems, a useful analysis is to translate the constraints on descriptor variables to constraints on inputs that will maintain the desired ranges. If the constraints are applied on individual descriptor vectors, the constraints can be translated to necessary conditions on individual input vectors. These necessary conditions apply to both regular and nonregular systems and are demonstrated in the final section of the chapter.

## 5.2 DYNAMIC PROGRAMMING FOR LINEAR DESCRIPTOR SYSTEMS

Often the major motivation in modeling a system is to provide a framework for determining a set of inputs that optimize the solution of the system model, and hence will serve as a nearly optimal policy for the actual system. In the context of the system being considered, it is reasonable to assume a separable objective function of the form:

$$\underset{u(0),\ldots,u(N-1)}{\text{minimize}} \left\{ \sum_{k=0}^{N-1} h_k(x(k),u(k)) + h_N(N) \right\} \qquad (5.1)$$

Dynamic programming is a technique commonly used to optimize dynamic systems with a separable objective function. Dynamic programming allows such optimization problems to be solved as a series of smaller optimization problems in the same way that condition systems can be used to decompose the solution of descriptor systems. In fact, as will be seen in this section, the condition systems have an important role in efficient optimization.

However, dynamic programming has been developed for state-space systems, which are special cases of descriptor systems. The purpose here is to demonstrate how this technique can be extended to all solvable, linear descriptor systems.

The theoretical basis for determining the globally optimal policy using dynamic programming is the principle of optimality. In the context of state-space systems, this principle states that given any state x(k), if $u^*(k)$, $u^*(k+1),\ldots,u^*(N-1)$ is the optimal trajectory of inputs corresponding to x(k), then $u^*(k+1),\ldots u^*(N-1)$ is the optimal trajectory of inputs corresponding to the state x(k+1) resulting from x(k) and $u^*(k)$. Since a descriptor vector x(k) and input u(k) do not always uniquely determine the following vector x(k+1) in a descriptor system, this principle must be reexpressed to be valid for all well-defined descriptor systems.

Suppose we consider the optimal cost-go-go from some descriptor vector x(i) at instant i to a descriptor vector x(j) at instant j. By the system equations and the objective function, it is clear that this cost depends only on the choice of u(i), u(i+1),...,u(j-1):

$$\text{Cost-to-go from } x(i) \text{ to } x(j) = \sum_{k=i}^{j-1} h_k(x(k),u(k)) \qquad (5.2)$$

Expression (5.2) can be interpreted as the objective function for the cost-to-go from instant i to instant j for specified x(i) and x(j). The value of this reduced objective function will obviously depend only on x(i), x(j), and the inputs u(i),u(i+1),...u(j-1). Note, however, that a set of inputs may not exist to link every pair x(i) and x(j); therefore, we must assume the pair is feasible.

Using (5.2) and minimizing with respect to u(i),u(i+1),...,u(j-1), an optimal trajectory of inputs can be determined. There may be multiple trajectories. Any one optimal trajectory will uniquely determine the descriptor vector x(i+1), using the system equations. From (5.2) it is clear that $u^*(i)$ must minimize the cost-to-go from x(i) to this value of x(i+1), where $u^*(i)$ is part of the optimal input trajectory for (5.2). Likewise, the other inputs of that trajectory, $u^*(i+1),\ldots,u^*(j-1)$ must minimize the cost-to-go from the determined x(i+1) to x(j). Therefore minimizing cost-to-go from x(i) to x(j) is equivalent to finding a consistent x(i+1) that minimizes the sum of the cost-to-go from x(i) to x(i+1) plus the cost-to-go from x(i+1) to x(j). This constitutes a principle of optimality for descriptor systems.

This principle can be exploited in the design of a dynamic programming algorithm for determining the optimal policy for a descriptor system with respect to (5.1). The algorithm recursively determines the optimal input u(k) and cost-to-go from a particular x(k) to any feasible x(N) plus the cost of x(N), for all feasible x(k), first where k=N-1, then where k=N-2, etc.. Now for any feasible x(k), there is a subset of possible inputs u(k) that can be used with x(k). One input u(k) is selected from this subset. This immediately determines y(k+1), the forward condition vector corresponding to set k+1. However, there is possibly an entire range of backward condition vectors z(k+1) for set k+1 that are consistent with x(k) and u(k) (or z(k) and u(k)). For each z(k+1) in this range, the minimal cost-to-go from x(k) to any x(N) (including the cost of x(N)) is computed by summing the costs of x(k), u(k), and the optimal cost-to-go from x(k+1) (determined uniquely by the condition vectors y(k+1) and z(k+1)). This determines an optimal cost-to-go from x(k) for given u(k) and z(k+1). These computations are repeated (if necessary) for other values of z(k+1) in the consistent range, so that an optimal cost from x(k) using u(k) can be identified. One can then consider other feasible values of u(k) and repeat the process of determining the optimal z(k+1), such that an optimal input u(k) and cost-to-go can be determined for x(k). When the overall procedure is repeated for all feasible x(k), the iteration for set k is complete, and one can progress to the iteration for set k-1. This algorithm is repeated in a step-by-step manner below:

Define the following sets: Let $Z_k$ be the set of backward condition vectors z(k) that are reachable from the backward condition vector z(N) These sets can be generated recursively using the backward condition system where

$$z(k) = S_{k+1}z(k+1) - Q_{k+1}u(k)$$

Therefore

$$Z_k = \{S_{k+1}z(k+1) - Q_{k+1}u(k) \,|\, z(k+1)\varepsilon Z_{k+1}, \text{all } u(k)\}$$

and

$$Z_N = z(N)$$

Let $U_k(z(k))$ be the set of u(k) such that z(k) is reachable from some $z(k+1)\varepsilon Z_{k+1}$:

$$U_k(z(k)) = \{u(k) \,|\, z(k)+Q_{k+1}u(k)\varepsilon S_{k+1}Z_{k+1}\}$$

Let $Z_{k+1}(z(k),u(k))$ be the subset of $Z_{k+1}$ such that z(k) is reached from z(k+1) using u(k):

$$Z_{k+1}(z(k),u(k))=\{z(k+1) \,|\, S_{k+1}z(k+1)-Q_{k+1}u(k)=z(k)\}$$

Determining these sets requires solving equations of the form Hx=d for all solutions, where H may not have full row rank. If the solution set is not empty, one solution can be obtained using a pseudoinverse of H. The remaining solutions will be the linear manifold of vectors that are the sum of the determined solution plus an element in the null space of the row vectors in H. It is clear that for any $z(k)\varepsilon Z_k$, $U_k(z(k))$ cannot be empty by definition of $Z_k$. Similarly $Z_{k+1}(z(k), u(k))$ corresponding to $u(k)\varepsilon U_k(z(k))$ cannot be empty.

The algorithm proceeds as follows:

For any feasible $x(N)$, determine the value of $h_N(x(N))$. For the iteration at set k, $1 \le k \le N-1$, starting with $k=N-1$:

1. Consider any $x(k)$ such that $z(k) = \Lambda_k x(k)$, the backward condition vector, is in $Z_k$. The goal is to determine the optimal cost from $x(k)$,

$$\sum_{i=k}^{N-1} h_i(x(i),u(i)) + h_N(x(N))$$

and the corresponding value(s) of $u(k)$:

2. Determine $U_k(z(k))$.

3. Choose a particular $u(k) \in U(z(k))$.

4. Calculate

$$y(k+1) = R_k y(k) + P_k u(k)$$

where $y(k) = \Gamma_k x(k)$ is the forward condition vector for $x(k)$.

5. Determine $Z_{k+1}(z(k),u(k))$.

6. Compute

$$L_k^*(x(k),u(k)) = \min_{x(k+1)} J_{k+1}(x(k+1))$$

where $\Lambda_{k+1} x(k+1) \in Z_{k+1}(z(k),u(k))$

$$\Gamma_{k+1} x(k+1) = y(k+1)$$

Store $L_k^*(x(k),u(k))$ and

$$x_{k+1}^*(x(k),u(k)) = \arg L_k^*(x(k),u(k))$$

7. Repeat steps 4-6 for all remaining $u(k) \in U_k(z(k))$.

8. Compute

$$J_k^*(x(k)) = \min_{u(k) \in U_k(z(k))} \{h_k(x(k),u(k)) + L_k^*(x(k),u(k))\}$$

Store $J^*(x(k))$ and

$$u^*(x(k)) = \arg (J_k^*(x(k)))$$

9. Repeat steps 1-8 for other descriptor vectors $x(k)$ satisfying $\Lambda_k x(k) \in Z_k$, for which $J_k^*(x(k))$ has not been computed.

10. If $k>1$, set $k=k-1$ and repeat steps 1-9. Otherwise proceed with the iteration for the set of $x(0)$.

For the iteration at $x(0)$, the steps 1-9 are repeated with one minor change. Since the initial conditions $\Gamma_0 x(0) = y(0)$ are given, it is only necessary to consider that subspace of $x(0)$. Once the iteration for all feasible $x(0)$ is complete, we determine the value(s) of $x(0)$ that minimizes $J_0(x(0))$. The globally optimal policy and optimal solution can be recovered: The optimal $x(0)$ determines an optimal $u(0)$. The vectors $x^*(0)$ and $u^*(0)$ determine $y(1)$ and $z(1) = \arg L^*(x(0),u(0))$, which equivalently determines $x^*(1)$. Continuing in this fashion one recovers $u^*(1)$, $x^*(2),\ldots,u^*(N-1)$, $x^*(N)$.

The preceding algorithm indicates how the concept of dynamic programming can be employed in optimizing these descriptor systems without insisting that the system be causal, as in state-space systems. For particular types of objective functions, it may be possible to perform the above iterations analytically, that is determine a cost-to-go from $x(k)$ that is a simple function of $x(k)$, rather than repeating the process for various specific values of $x(k)$. For more complex objective functions, some approximations are generally necessary since there are an uncountable number of possible $x(k)$. However, the advanced computational procedures developed for standard dynamic programming problems can also be applied in these optimization problems to determine an approximately optimal policy (see [21]).

When the dynamic programming algorithm described above is applied to a state-space system, it automatically simplifies to the standard dynamic programming algorithm if the maximal number of initial conditions are used. Since there is no backward condition system in this case, the set of inputs $u(k)$ is not systematically constrained by $x(k)$. Furthermore, the descriptor vector is a state vector, so $x(k)$ and $u(k)$ determine a unique $x(k+1)$. This eliminates the need for repetition of steps 5-7. Thus, the recursive procedure automatically assumes the conventional embedding property of state-space dynamic programming.

## 5.3 DYNAMIC PROGRAMMING FOR REGULAR DESCRIPTOR SYSTEMS

Recall that a square descriptor system of the form

$$E_{k+1} x(k+1) = A_k x(k) + B_k u(k) \quad k=0,\ldots,N-1 \tag{5.3}$$

is regular if and only if it can be transformed via elementary row operations into the form

$$\begin{bmatrix} \Gamma_{k+1} \\ \hline 0 \end{bmatrix} x(k+1) = \begin{bmatrix} C_k \\ \hline D_k \end{bmatrix} x(k) + \begin{bmatrix} G_k \\ \hline H_k \end{bmatrix} u(k) \tag{5.4}$$

where $\begin{bmatrix} \Gamma_k \\ \hline D_k \end{bmatrix}$ is square and nonsingular for all $k=0,\ldots,N$, assuming a proper choice of $\Gamma_0$ and $D_N$. Now for regular systems, $y(k)=\Gamma_k x(k)$ serves as a state in the sense that a propagation of $y(k)$ forward in time is sufficient to recover the value of $x(k)$ for all k, since there exist matrices $R_k, P_k, L_k$, and $M_k$ such that

$$y(k) = R_k y(k) + P_k u(k) \tag{5.5}$$

$$x(k) = L_k y(k) - M_k u(k) \qquad k=0,\ldots,N-1 \tag{5.6}$$

$$x(N) = \left[\begin{array}{c}\Gamma_N \\ \hline \Lambda_N\end{array}\right]^{-1}\left[\begin{array}{c}y(N) \\ \hline z_N\end{array}\right] = L_N y(N) + W_N z_N \tag{5.7}$$

where $\Lambda_N x(N)=z_N$ are the final conditions. We will assume that $z_N$ is exogenously given.

Consider the problem

$$\text{minimize} \sum_{j=0}^{N-1} h_j(x(j),u(j)) + \ell(x(N))$$

$$\text{subject to } \Gamma_0\, x(0) = y(0)$$

$$\Lambda_N\, x(N) = z_N$$

For regular systems, the terminal conditions $\Lambda_N x(N) = z_N$ serve only to specify x(N) and do not constrain the inputs or affect previous descriptor variables. Recall that in any solvable system, the descriptor variable vector x(k) can be determined from the forward condition vector y(k) and the backward condition vector z(k). Therefore, if $z(N) = z_N$ is known, y(N) represents a unique x(N). Hence for regular systems the terminal conditions can be removed by replacing $\bar{\ell}(y(N)) = \bar{\ell}(\Gamma_N x(N)) = \ell(x(N))$. The problem for regular systems becomes

$$\text{minimize} \sum_{j=0}^{N-1} h_j(x(j),u(j)) + \bar{\ell}(y(N))$$

$$\text{subject to } \Gamma_0\, x(0) = y(0)$$

In a regular system, the forward condition vector $y(k) = \Gamma_k x(k)$ is a state and thus summarizes all past inputs and descriptor variables as they affect the present and future descriptor variables. In other words, if one is trying to determine x(m), $m \geq k$, it is unnecessary to specify u(j), $j<k$, if y(k) is known. Therefore, one can determine a "cost-to-go" from k to N for a given y(k) without knowing the inputs previous to k. One can then calculate the optimal set of inputs from k to N-1 given y(k), without determining the previous inputs. In addition, the objective function is separable in time, leading to a natural application of the principle of optimality which is presented below:

Let

$$J_N(y(N)) = \bar{\ell}(y(N))$$

and for $0 \leq k \leq N-1$

$$J_k(y(k)) = \min_{u(k),\ldots,u(N-1)} \left\{ \sum_{j=k}^{N-1} h_j(x(j),u(j)) + \bar{\ell}(y(N)) \right\}$$

such that $\Gamma_k x(k) = y(k)$.

Assume $J_{k+1}(y(k+1))$ has been determined for all y(k+1) and consider $J_k(y(k))$.

From separability of the objective function and the propagation of the forward condition (state) vector given by

$$y(k+1) = R_k\, y(k) + P_k u(k)$$

one has

$$J_k(y(k)) = \min_{u(k)} \left\{ h_k(x(k),u(k)) + J_{k+1}(y(k+1)) \right\}$$

$$= \min_{u(k)} \left\{ h_k(x(k),u(k)) + J_{k+1}(R_k y(k) + P_k\, u(k)) \right\}$$

where $\Gamma_k\, x(k) = y(k)$

Since the system is regular, y(k) and u(k) uniquely determines x(k) through the relation

$$x(k) = L_k\, y(k) - M_k\, u(k)$$

Substituting for x(k), one has

$$J_k(y(k)) = \min_{u(k)} \left\{ h_k(L_k y(k) - M_k u(k), u(k)) + J_{k+1}(R_k y(k) + P_k u(k)) \right\}.$$

$J_k(y(k))$ is determined for all y(k), and the corresponding optimal u(k) is stored. This procedure is repeated for time increments k-1, k-2,...,0. At k=0 it is only necessary to calculate $J_0(y(0))$ for the given initial conditions $\Gamma_0(x(0)) = y(0)$. A forward sweep will then recover the entire trajectory of optimal inputs $u^*(k)$ that minimizes $J_0(y(0))$. This procedure corresponds almost identically to the procedure for state-space systems, which are simply regular systems where all descriptor variables serve as state variables. However, in state-space systems y(k) is identical to x(k), while in the general regular system y(k) only restricts x(k) to a linear variety of the descriptor space. Complete specification of x(k) in a general regular system requires knowing the static relationships of the system at time k, which are independent of the lower-order dynamic forward condition (state) system characterizing y(k). Note that the dimension of y(k) plus the number of static relationships equals the dimension of x(k) in a regular system.

This procedure is clearly more efficient than the procedure described in Section 5.2, since by using the cost-to-go from the forward condition vector at time increment k, any input u(k) is feasible. Recall that in determining the

cost-to-go from a specific descriptor vector x(k), there is generally a linear variety of feasible inputs that must be determined prior to computing the optimal control. Therefore, the steps for determining the feasible inputs and reachable descriptor vectors are avoided in this procedure. However, both procedures are equivalent for state-space systems since in this case they are each equivalent to classical state-space dynamic programming.

*Linear-Quadratic Problem for Square, Regular Descriptor Systems*

We now apply this procedure for a regular system with a quadratic cost criterion:

$$\underset{u(0),\ldots,u(N-1)}{\text{minimize}} \quad \frac{1}{2}\sum_{k=0}^{N} x'(k)\, Q_k x(k) + \frac{1}{2}\sum_{k=0}^{N-1} u'(k)\, S_k\, u(k) \qquad (5.8)$$

where it is assumed that $S_k$ is positive definite and $Q_k$ is positive semidefinite. Let $J_k(y(k))$ represent the cost-to-go from y(k) given optimal inputs $u^*(k),\ldots,u^*(N-1)$:

$$J_k(y(k)) = \frac{1}{2}\sum_{j=k}^{N} x'(j)\, Q_j\, x(j) + \frac{1}{2}\sum_{j=k}^{N-1} u^{*\prime}(j)\, S_j u^*(j)$$

where for k=N,

$$J_N(y(N)) = \frac{1}{2}\, x'(N)\, Q_N x(N)$$

Since $x(N) = L_N y(N) + W_N z_N$ for a specified $z_N$, it follows that $J_N(y(N))$ is quadratic in y(N). Using this observation, we will assume (and later prove) that $J_k(y(k))$ is a quadratic function of y(k) for all k, i.e.

$$J_k(y(k)) = \frac{1}{2}\, y'(k)\, K_k y(k) + g_k' y(k) + h_k \qquad (5.9)$$

where $K_k$ is a positive semi-definite matrix and $g_k$ is a constant column vector.

Now the objective of the dynamic programming approach is to determine the optimal u(k) for any given state at time k by embedding the cost-to-go from state y(k+1) at time k+1. As was demonstrated earlier, we have

$$J_k(y(k)) = \min_{u(k)} \left\{ \frac{1}{2}(L_k y(k) - M_k u(k))' Q_k (L_k y(k) - M_k u(k)) + \frac{1}{2}\, u'(k)\, S_k u(k) + J_{k+1}(R_k y(k) + P_k u(k)) \right\} \qquad (5.10)$$

Now using the assumption in (5.9) and substituting the quadratic expression for $J_{k+1}$, (5.10) becomes

$$J_k(y(k)) = \min_{u(k)} \left\{ \frac{1}{2}\begin{bmatrix} y'(k) & u'(k) \end{bmatrix} \begin{bmatrix} \bar{Q}_k & \bar{N}_k' \\ \bar{N}_k & \bar{S}_k \end{bmatrix} \begin{bmatrix} y(k) \\ u(k) \end{bmatrix} + g_{k+1}' \begin{bmatrix} R_k & P_k \end{bmatrix} \begin{bmatrix} y(k) \\ u(k) \end{bmatrix} + h_{k+1} \right\} \qquad (5.11)$$

where

$$\bar{Q}_k = L_k' Q_k L_k + R_k' K_{k+1} R_k \qquad (5.12)$$

$$\bar{N}_k = -M_k' Q_k L_k + P_k' K_{k+1} R_k \qquad (5.13)$$

$$\bar{S}_k = S_k + M_k' Q_k M_k + P_k' K_{k+1} P_k \qquad (5.14)$$

By our assumptions, $\bar{S}_k$ is positive definite and $\bar{Q}_k$ is positive semi-definite.

The optimal input $u^*(k)$ corresponding to y(k) is given by:

$$u^*(k) = -F_k y(k) - v_k \qquad (5.15)$$

where

$$F_k = \bar{S}_k^{-1} \bar{N}_k \qquad (5.16)$$

$$v_k = \bar{S}_k^{-1} P_k' g_{k+1} \qquad (5.17)$$

Hence the optimal u(k) is a linear function of y(k). Substituting (5.15) back into (5.11), we get the general form (5.9) where $K_k$, $g_k$, and $h_k$ are given by:

$$K_k = [I \; -F_k'] \begin{bmatrix} \bar{Q}_k & \bar{N}_k' \\ \bar{N}_k & \bar{S}_k \end{bmatrix} \begin{bmatrix} I \\ -F_k \end{bmatrix} \qquad (5.18)$$

$$g_k = [I \; -F_k'] \begin{bmatrix} \bar{Q}_k & \bar{N}_k' \\ \bar{N}_k & \bar{S}_k \end{bmatrix} \begin{bmatrix} 0 \\ -v_k \end{bmatrix} + [I \; -F_k'] \begin{bmatrix} R_k' \\ P_k' \end{bmatrix} g_{k+1} \qquad (5.19)$$

$$h_k = h_{k+1} + \frac{1}{2}\, v_k' \bar{S}_k v_k - v_k' P_k' g_{k+1} \qquad (5.20)$$

Since $K_k$, $g_k$, and $h_k$ are independent of y(k), it is noted that these can be generated recursively backward using (5.12)-(5.14) and (5.16)-(5.20) starting from the coefficients of $J_N(y(N))$:

$$K_N = L_N' Q_N L_N \qquad (5.21)$$

$$g_N = L_N' Q_N W_N z_N \qquad (5.22)$$

$$h_N = \frac{1}{2}\, z_N' W_N' Q_N W_N z_N \qquad (5.23)$$

We now show that $K_k$ is positive semi-definite for all k, $0 \le k \le N$. It is clear from (5.12)-(5.16), (5.18), and (5.21) that $K_k$ is generated independent of the value of $z_N$ and therefore will have

the same value for any $z_N$. For convenience assume $z_N=\theta$, where $\theta$ is the zero vector. From (5.22) and (5.23) this implies $g_N=\theta$ and $h_N=0$ and, one notes from (5.17)-(5.20), that $g_k=\theta$ and $h_k=0$ for all $k, 0\leq k\leq N$. So in this case (5.9) becomes

$$J_k(y(k)) = \frac{1}{2}\, y'(k)\, K_k y(k) \qquad (5.24)$$

Now the objective function is always positive semi-definite by assumptions on $Q_k$ and $S_k$. Therefore, it follows from (5.24) that $K_k$ must be positive semi-definite.

We demonstrated that $J_k(y(k))$ had quadratic form (5.9) by assuming $J_{k+1}(y(k+1))$ had a similar form. In order to establish the validity of the assumption, we note that $J_N(y(N))$ has this form for all $y(N)$, and therefore the assumption for any $k$, $0\leq k\leq N$, follows by induction.

We now summarize the result in the form of a theorem:

*Theorem 5.1:* Given a square, regular system described by (5.3)-(5.7) and objective (5.8) with arbitrary initial conditions on $\Gamma_0 x(0)$ and specified final conditions $z_N=\Lambda_N x(N)$, the optimal input $u^*(k)$ at time $k$ is a linear function of $y(k)$ described by (5.12)-(5.23) and the optimal cost is given by

$$J_o(y(0)) = \frac{1}{2}\, y'(0) K_o y(0) + g_o' y(0) + h_o$$

Therefore, given one knows the final conditions $z_N=\Lambda_N x(N)$, there exists a linear feedback solution for this problem. It is important to note that the feedback control depends on the state $y(k)$ and not the entire descriptor vector $x(k)$.

## 5.4 MAINTAINABILITY OF CONSTRAINTS IN A STATE-SPACE SYSTEM

Consider a state-space system

$$x(k+1) = A_k x(k) + B_k u(k) \qquad k=0,\ldots,N-1 \qquad (5.25)$$

with equality constraints

$$F_{k+1} x(k+1) = c_{k+1} \qquad k=0,\ldots,N-1 \qquad (5.26)$$

Assume $F_{k+1}$ always has full row rank to avoid possible redundancies and inconsistencies in the constraints.

Clearly one would like to be able to choose inputs $u(0),\ldots,u(N-1)$ such that the state variables generated by (5.25) will satisfy (5.26) for any initial state $x(0)$. Such a system is defined to be *completely maintainable*.

A class of completely maintainable systems that is important for the existence of optimal feedback control laws are systems described by (5.25) and (5.26) such that for any $j$, $0\leq j\leq N-1$, the system equations and constraints defined for $k=j,\ldots,N-1$ are completely maintainable for any $x(j)$. We define such systems to be *uniformly completely maintainable*. A very useful characterization of uniform complete maintainability is given by the following theorem:

*Theorem 5.2:* A system described by (5.25) and (5.26) is uniformly completely maintainable if and only if at all $k$, $0\leq k\leq N-1$, the space spanned by $c_{k+1}$ and the column vectors of $F_{k+1}A_k$ is spanned by the column vectors of $F_{k+1}\,B_k$.

*Proof:* From the definition of uniform complete maintainability, it follows that for $x(j)$ at increment $j$, (5.26) must be satisfied for $k=j$. Using the system equation (5.25), this is equivalent to stating that there must exist, for every $x(j)$, some $u(j)$ such that

$$F_{j+1}\, A_j\, x(j) + F_{j+1} B_j u(j) = c_{j+1} \qquad (5.27)$$

Hence the matrix condition given in the theorem is necessary for uniform complete maintainability.

Similarly, if (5.27) can be satisfied for any $x(j)$, $j=0,\ldots,N-1$, the choice of $u(k)$ at increment $k$ never precludes the existence of some $u(j)$, $j>k$, that will satisfy (5.26) for $x(j+1)$. Therefore, the matrix condition is also sufficient for uniform complete maintainability. ■

Complete maintainability does not imply uniform complete maintainability, however, it does imply that (5.26) is satisfied for all $k$. Equivalently it implies that (5.27) must be true for some $x(j)$ and some $u(j)$, which gives the following theorem:

*Theorem 5.3:* A system described by (5.25) and (5.26) is completely maintainable only if for all $k$, $0\leq k\leq N-1$, $c_{k+1}$ falls in the space spanned by the column vectors of $F_{k+1}\,A_k$ and $F_{k+1}\,B_k$.

According to Theorem 5.2, a sufficient (but not necessary) condition for uniform complete maintainability is that $F_{k+1}\,B_k$ has full row rank for all $k$. Suppose this condition does not hold, then it is useful to cite the following lemma:

*Lemma 5.4:* The set of constraints (5.26) for any uniformly completely maintainable system can be replaced by a set of constraints

$$\bar{F}_{k+1}\, x(k+1) = \bar{c}_{k+1} \qquad k=0,\ldots,N-1 \qquad (5.28)$$

which are equivalent to (5.26) with respect to the system equation (5.25), and for which $\bar{F}_{k+1}\,B_k$ has full row rank.

*Proof:* Suppose $F_{k+1}\,B_k$ does not have full row rank for $k=j$. By performing elementary row operations on $F_{j+1}\,B_j$, one can create a matrix

$$\begin{bmatrix} \bar{F}_{j+1}\, B_j \\ \hline 0 \end{bmatrix} \qquad (5.29)$$

for which $\bar{F}_{j+1}\,B_j$ has full row rank. Suppose we perform the same elementary row operations on $F_{j+1}\,A_j$ and $c_{j+1}$. By Theorem 5.2, the new rows

of $F_{j+1}A_j$ and the new elements of $c_{j+1}$ corresponding to the zero rows of (5.29) must also be zero. Therefore, the all-zero rows can be eliminated, leaving a new set (5.28) for which $\bar{F}_{j+1}B_j$ has full row rank. ■

Uniformly completely maintainable systems expressed by (5.25) and (5.28) are in a convenient form for deriving feedback optimal control laws for regular descriptor systems, as we will demonstrate in Section 5.5. Before proceeding to those investigations, we should investigate further the relationship between complete maintainability and uniform complete maintainability, so that the results derived in future sections can be applied to as wide a class of systems as possible. An important relationship is given by the following theorem:

*Theorem 5.5:* A system described by (5.25) and (5.26) is completely maintainable if and only if it can be transformed to a uniformly completely maintainable system by shifting some of the constraints to become new constraints on state variables at earlier time increments.

*Proof:* Consider the set of constraints on x(j+1) for any j, j=0,...,N-1, expressed in the form of Equation (5.27). Suppose $F_{j+1}$ does not have full row rank, and we perform the elementary row operations to create (5.29), then (5.27) becomes:

$$\left[\begin{array}{c|c}\bar{F}_{j+1} & A_j \\ \hline \tilde{F}_{j+1} & A_j\end{array}\right] x(j) + \left[\begin{array}{c|c}\bar{F}_{j+1} & B_j \\ \hline 0 & \end{array}\right] u(j) = \left[\begin{array}{c}\bar{c}_{j+1} \\ \hline \tilde{c}_{j+1}\end{array}\right] \quad (5.30)$$

Equation (5.30) represents two sets of constraints: one set constraining x(j) and u(j) and a second set constraining just x(j). If we "transfer" the second set from being constraints on x(j+1) to being constraints on x(j), the remaining first set will satisfy the uniform complete maintainability property at j. However, we now have more constraints on x(j). Nonetheless, we have not added any new constraints to the overall system, and therefore the complete maintainability of the system must still hold. Therefore, by Theorem 5.3 the composite set of constraints on x(j) must have a feasible solution.

The above procedure of transferring constraints forms the basis for an algorithm that reformulates any completely maintainable system into an equivalent uniformly completely maintainable system:

STEP 0. Set k=N-1.

STEP 1. Consider $F_{k+1}x(k+1) = c_{k+1}$. If $F_{k+1}B_k$ does not have full rank, perform the elementary row operations necessary to create (5.30). If $F_{k+1}B_k$ does have full row rank, proceed to STEP 4.

STEP 2. Retain $\bar{F}_{k+1}x(k+1) = \bar{c}_{k+1}$ as the constraints on x(k+1).

STEP 3. Add constraints $F_{k+1}A_k x(k) = \tilde{c}_{k+1}$ to the existing constraints on x(k).

STEP 4. If k=0, proceed to STEP 5. Otherwise, set k=k-1, redefine $F_{k+1}$ and $c_{k+1}$ appropriately, and return to STEP 1.

STEP 5. If STEP 3 produced constraints on x(0), by definition, complete maintainability does not hold for the system. Otherwise, the system has been equivalently re-expressed as a uniformly completely maintainable system. ■

Lemma 5.4 and Theorem 5.5 indicate that any completely maintainable system can be expressed in the form of (5.25) and (5.26), where $F_{k+1}B_k$ has full row rank at all k. Therefore, in our discussion of optimal control for completely maintainable systems, we will assume this full row rank property without further justification.

## 5.5 LINEAR-QUADRATIC PROBLEM FOR COMPLETELY MAINTAINABLE RECTANGULAR SYSTEMS

We now apply the concept of complete maintainability in addressing optimal control for systems that are rectangular, or have more relationships than descriptor variables. A quadratic objective function will be assumed. As indicated in Section 5.1, an important special case of this class are state-space systems with state constraints. Because of the special structure of this case, it is helpful to initially address the constrained state-space problem. The derivation of feedback optimal control for completely maintainable state-space systems suggests an approach for deriving a similar result for the more general class of rectangular, regular systems. Both results demonstrate that the optimal control approach can be applied to individual subsystem models for the purpose of large-scale optimization.

Consider a state-space system

$$x(k+1) = A_k x(k) + B_k u(k) \quad k=0,\ldots,N-1 \quad (5.31)$$

with completely maintainable constraints

$$F_{k+1}x(k+1) = c_{k+1} \quad k=0,\ldots,N-1 \quad (5.32)$$

We assume $F_{k+1}B_k$ has full row rank at all k by complete maintainability.

Suppose we have the usual quadratic criterion of the form

$$\min_{u(0),\ldots,u(N-1)} \left\{\frac{1}{2}\sum_{k=0}^{N} x'(k)Q_k x(k) + \frac{1}{2}\sum_{k=0}^{N-1} u'(k)R_k u(k)\right\} \quad (5.33)$$

where $Q_k$ is positive semi-definite and $R_k$ is positive definite for all k. Since the criterion is separable by time increments, we can use the dynamic programming approach where, by successive embedding, we compute the optimal input u(k) and optimal cost-to-go from any x(k) to the final state at time N for k=N-1,N-2,...,0. We assume this optimal cost-to-go has the quadratic form:

$$J_k(x(k)) = x'(k)\,K_k x(k) + g_k' x(k) + h_k \quad (5.34)$$

where $K_k$ is positive semi-definite. By construction of the objective function, it is clear that this is true for k=N. In the process of deriving the optimal control law, we will show that this is indeed the case for all k by induction, since we will show that if the assumption is valid for k=j+1, it is also valid for k=j.

Now consider $J_k(x(k))$ for some k, $0 \le k \le N-1$. Using the system equation (5.31), the cost-to-go becomes

$$J_k(x(k)) = \min_{u(k),\ldots,u(N-1)} \left\{ \frac{1}{2}\sum_{j=k}^{N} x'(j)Q_j x(j) + \frac{1}{2}\sum_{j=k}^{N-1} u'(j)R_j u(j) \right\} \quad (5.35)$$

$$= \min_{u(k)} \left\{ \frac{1}{2}\, x'(k)Q_k x(k) + \frac{1}{2}\, u'(k)R_k u(k) + J_{k+1}(A_k x(k) + B_k u(k)) \right\}$$

subject to constraints (5.32) on x(k+1):

$$F_{k+1}A_k x(k) + F_{k+1}B_k u(k) = c_{k+1} \quad (5.36)$$

We will assume $J_{k+1}(x(k+1))$ has the form (5.34). Furthermore, in order to satisfy (5.36) in the optimization, we adjoin the constraints (5.36) to (5.35) using Lagrange multipliers. The new objective function is:

$$\bar{J}_k(x(k)) = \min_{(u'(k),\lambda'_{k+1})} \left\{ \frac{1}{2}[x'(k),u(k),\lambda'_{k+1}] \begin{bmatrix} \bar{Q}_k & \bar{M}_k' & \bar{N}_k' \\ \bar{M}_k & \bar{R}_k & \bar{P}_k' \\ \bar{N}_k & \bar{P}_k & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \\ \lambda_{k+1} \end{bmatrix} + [x'(k),u'(k),\lambda'_{k+1}] \begin{bmatrix} A_k' g_{k+1} \\ B_k' g_{k+1} \\ -c_{k+1} \end{bmatrix} + h_{k+1} \right\} \quad (5.37)$$

where

$$\bar{Q}_k = Q_k + A_k' K_{k+1} A_k \quad \text{, positive semi-definite} \quad (5.38)$$

$$\bar{M}_k = B_k' K_{k+1} A_k \quad (5.39)$$

$$\bar{N}_k = F_{k+1} A_k \quad (5.40)$$

$$\bar{R}_k = R_k + B_k' K_{k+1} B_k \quad \text{, positive definite} \quad (5.41)$$

$$\bar{P}_k = F_{k+1} B_k \quad \text{, full row rank} \quad (5.42)$$

The first-order necessary conditions are:

$$\begin{bmatrix} \bar{M}_k & \bar{R}_k & \bar{P}_k' \\ \bar{N}_k & \bar{P}_k & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \\ \lambda_{k+1} \end{bmatrix} + \begin{bmatrix} B_k' g_{k+1} \\ -c_{k+1} \end{bmatrix} = 0 \quad (5.43)$$

Now

$$\begin{bmatrix} \bar{R}_k & \bar{P}_k' \\ \bar{P}_k & 0 \end{bmatrix}^{-1} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \quad (5.44)$$

where the existence of submatrices

$$G_{11} = \bar{R}_k^{-1} - \bar{R}_k^{-1}\bar{P}_k'(\bar{P}_k \bar{R}_k^{-1} \bar{P}_k')^{-1} \bar{P}_k \bar{R}_k^{-1} \quad (5.45)$$

$$G_{12} = \bar{R}_k^{-1} \bar{P}_k' (\bar{P}_k \bar{R}_k^{-1} \bar{P}_k')^{-1} \quad (5.46)$$

$$G_{21} = (\bar{P}_k \bar{R}_k^{-1} \bar{P}_k')^{-1} \bar{P}_k \bar{R}_k^{-1} \quad (5.47)$$

$$G_{22} = -(\bar{P}_k \bar{R}_k^{-1} \bar{P}_k')^{-1} \quad (5.48)$$

can be established, Hence (5.43) becomes

$$\begin{bmatrix} u(k) \\ \lambda_{k+1} \end{bmatrix} = -\begin{bmatrix} \bar{R}_k & \bar{P}_k' \\ \bar{P}_k & 0 \end{bmatrix}^{-1} \left\{ \begin{bmatrix} \bar{M}_k \\ \bar{N}_k \end{bmatrix} x(k) + \begin{bmatrix} B_k' & g_{k+1} \\ -c_{k+1} \end{bmatrix} \right\} \quad (5.49)$$

Therefore the optimal input u(k) is a linear function of state x(k), as is the corresponding Lagrange multiplier $\lambda_{k+1}$. Substituting (5.49) into (5.37) yields a quadratic expression for $J_k(x(k))$ of the form (5.34). Since the coefficients of this quadratic expression can be determined from (5.37)-(5.49), we can essentially generate the coefficients of $J_k(x(k))$ from the coefficients of $J_{k+1}(x(k+1))$, the system equation for x(k+1), the constraints on x(k+1), and the performance criteria for x(k) and u(k). This is similar to the matrix Riccati equation that results in unconstrained problems.

In order to complete our inductive proof validating the assumption of the structure of $J_{k+1}(x(k+1))$, we need only establish that $K_k$ is positive semi-definite. It is easily shown that $K_k$ is independent of $g_N, h_N$, and $\{c_j\}$, $1 \le j \le N$. If we reset the values of $g_N, h_N$, and $\{c_j\}$ all to zero, the values of $K_k$ will be unchanged, but $J_k(x(k))$ will have the form

$$J_k(x(k)) = x'(k)\,K_k\,x(k) \quad (5.50)$$

Since $J_k(x(k)) \ge 0$ by (5.35) and the positive semi-definiteness of all $Q_j$ and $R_j$, $K_k$ must also be positive semi-definite.

We now summarize the general result we have established for constrained state-space systems.

*Theorem 6.6:* Given a completely maintainable system characterized by (5.31) and (5.32) and a quadratic performance function (5.33), the optimal input is a linear function of the state, given by the feedback control law (5.49), and the optimal cost is given by (5.34) at k=0 for arbitrary initial conditions.

It is interesting to note that the feedback control law will, in general, have a constant vector term. This is similar to the result one derives in unconstrained systems where one tries to "track" a non-zero trajectory.

We will now extend this result to a more general case where (5.31) is replaced by any square, regular descriptor system. In Section 5.3 we derived the optimal control law for square, regular linear descriptor systems with quadratic performance criteria. In this section we will examine rectangular, regular systems and show how Theorem 5.6 can be used to determine an optimal control law for these systems. A rectangular linear descriptor system

$$\bar{E}_{k+1}\, x(k+1) = \bar{A}_k\, x(k) + \bar{B}_k u(k) \quad k=0,\ldots,N-1 \tag{5.51}$$

is *regular,* if by elementary row operations, the system can be transformed into a square, regular descriptor system plus a set of static relationships on x(k) and u(k) for k=0,...,N-1.

Clearly, rectangular systems must be such that $\bar{E}_{k+1}$ and $\bar{A}_k$ have at least as many rows as columns in order to be regular. Otherwise, solvability will be violated.

Suppose we have decomposed (5.51), as suggested by the definition of regularity, into a square, regular system

$$E_{k+1}\, x(k+1) = A_k\, x(k) + B_k u(k) \quad k=0,\ldots,N-1 \tag{5.52}$$

and a set of static relationships

$$\theta = \tilde{A}_k\, x(k) + \tilde{B}_k\, u(k) \quad k=0,\ldots,N-1 \tag{5.53}$$

where $z_N = \Lambda_N x(N)$ is given. We know from Section 5.3 that (5.52) can be reformulated using the forward condition vector:

$$y(k+1) = R_k y(k) + P_k u(k) \quad k=0,\ldots,N-1 \tag{5.54}$$

$$x(k) = L_k y(k) - M_k u(k) \quad k=0,\ldots,N-1 \tag{5.55}$$

$$x(N) = L_N y(N) + W_N z_N \qquad z_N \text{ given} \tag{5.56}$$

Substituting (5.55) into (5.53) yields

$$\theta = \tilde{A}_k L_k y(k) + (\tilde{B}_k - \tilde{A}_k M_k) u(k) \quad k=0,\ldots,N-1 \tag{5.57}$$

Suppose we wish to choose u(0),...,u(N-1) to minimize, as we did in Section 5.3,

$$\frac{1}{2}\sum_{k=0}^{N} x'(k) Q_k x(k) + \frac{1}{2}\sum_{k=0}^{N-1} u'(k)\, S_k u(k) \tag{5.58}$$

where $Q_k$ is positive semi-definite and $S_k$ is positive definite for all k. Substituting (5.55) and (5.56) into (5.58), yields a new objective function

$$\min_{u(0),\ldots,u(N-1)} \left\{ \frac{1}{2}\sum_{k=0}^{N-1} (L_k y(k) - M_k u(k))' Q_k (L_k y(k) - M_k u(k)) + \frac{1}{2}\,(L_N y(N) + W_N z_N)' Q_N (L_N y(N) + W_N z_N) + \frac{1}{2}\sum_{k=0}^{N-1} u'(k)\, S_k u(k) \right\} \tag{5.59}$$

Consider the problem described in (5.54), (5.57), and (5.59). It is clearly equivalent to the original minimization problem described in (5.52), (5.53), and (5.58). It is also very similar to the problem encountered in Theorem 5.6 since (5.54) is a state-space representation, (5.57) represents static constraints, and (5.59) is a quadratic function in y(k). This suggests the following theorem:

*Theorem 5.7:* For any properly conditioned, rectangular, regular descriptor system (5.51) that can be represented as a completely maintainable constrained state-space system (5.54) and (5.57), the optimal input u(k) corresponding to a quadratic objective function (5.58) is a linear function of the forward condition vector y(k).

*Proof:* First of all, we note there are only two differences between the problem addressed in Theorem 5.6 and the problem characterized by (5.54), (5.57) and (5.59) for y(k): (1) the constraints (5.57) are expressed in terms of y(k) and u(k) rather than y(k+1), and (2) the objective function is not separable in y(k) and u(k) and it has linear terms in y(N). Therefore, if we argue that these differences will not contradict Theorem 5.6, this theorem will be established.

While the constraints in Theorem 5.6 were originally stated in terms of x(k+1), the constraints were subsequently restated in terms of x(k) and u(k) when adjoined to the cost-to-go function. Therefore, there is no inherent problem in stating the constraints in terms of y(k) and u(k); in fact it is perhaps in a more convenient form for optimization. The definitions of complete maintainability and uniform complete maintainability apply and have the same significance in this problem as in Theorem 5.6. However, Theorem 5.2 does not apply, but we can find a similar condition:

*Lemma 5.8:* The constraint set described by (5.57) is uniformly completely maintainable if and only

if the space spanned by the columns of $\tilde{A}_k L_k$ is also spanned by the columns of $(\tilde{B}_k - \tilde{A}_k M_k)$.

*Proof:* This lemma follows from the definition of uniform complete maintainability since (5.59) at any k must have a solution for any y(k). Therefore using Lemma 5.8, one may characterize uniform complete maintainability for constraints given in terms of y(k) and u(k). While (5.59) may not immediately satisfy uniform complete maintainability, Theorem 5.5 assures us of an equivalent constraint system that does. ■

Returning to the proof of Theorem 5.7, it is fairly straightforward, by repeating the procedure (5.33) - (5.49), to show that the result of Theorem 5.6 is not altered by the existence in the objective function of cross-terms in y(k) and u(k), such as in (5.59). The existence of the term

$$\frac{1}{2}(L_N y(N) + W_N z_N)' Q_N (L_N y(N) + W_N z_N) \qquad (5.60)$$

in (5.59) also poses no major difficulty since the positive semidefiniteness of $Q_N$ assures the existence of a lower bound.

Therefore, Theorem 5.6 applies to this class of rectangular systems, as stated in Theorem 5.7. ■

The derivation of the results in this section was facilitated by the introduction of the concepts of *complete maintainability* and *uniform complete maintainability*. It is important to note that these properties are determined by the structure of the underlying state-space system *and* the constraint equations, not merely the latter. This underscores a conceptual advantage of using the descriptor approach in that it analyzes the composite of all equations characterizing the system to get the full benefit of system structure.

Theorem 5.7 considers "unconstrained" rectangular descriptor systems, although the procedure used to compute the optimal control treats some of the static relationships as constraints. It is a straightforward observation that the addition of static, linear constraints to a rectangular, regular system imposes no difficulties in computing a feedback control law as long as the added constraints do not destroy complete maintainability by the descriptor system. These constraints can be added to the separated static relationships to form a composite constraint set. Theorem 5.7 applies immediately and the procedure in Section 5.4 can be used to create uniformly completely maintainable constraints.

The above observation establishes an important link between constrained state-space systems and rectangular, regular descriptor systems with or without constraints. Such an observation was anticipated from the common approach used for both classes of systems. This result suggests that descriptor variable theory will be useful for the optimization of large-scale systems, specifically when the interacting subsystems are characterized by state-space models. The existence of feedback solutions to subsystem optimization problems fits nicely into the technique of spatial dynamic programming, discussed in Chapter VI.

## 5.6 MAINTAINABILITY OF CONSTRAINTS IN GENERAL SYSTEMS

A procedure for accommodating equality constraints on descriptor variables in regular systems was outlined in Sections 5.4 and 5.5. In this case it was possible to determine constraints, specified locally in terms of x(k) and u(k), that were necessary and sufficient for satisfying the original constraints. This approach will not extend to nonregular systems, however, since the solution cannot be generally recovered using a forward condition system and static constraints. Some remarks on the general case appear in this section.

If the constraints are imposed directly on linear combinations of descriptor variables (as was assumed previously), the constraints can be reformulated as constraints on the inputs using the sensitivity results of Section 2.5. Recall that the solution of any particular descriptor can be expressed as a linear function of the inputs and the initial and final condition vectors:

$$x(k) = \bar{D}_k y(0) + \sum_{j=0}^{N-1} D_{kj} u(j) + \tilde{D}_k z(N) \qquad (5.61)$$

Therefore any constraint on x(k) is equivalently a constraint on the right-hand side of (5.61), and since y(0) and z(N) are known, the constraint is reformulated in terms of u(0),...,u(N-1). The equivalence of these constraints means that selection of inputs that satisfy the reformulated constraints is necessary and sufficient to satisfy the constraints on descriptor variables. This result applies directly to inequality constraints on descriptor vectors as well.

While the above procedure creates an equivalent set of constraints in terms of the inputs, the fact that a single constraint in this set may involve every input vector can be inconvenient. Certainly it would be preferable to characterize the set of consistent inputs in terms of constraints on individual input vectors. Unfortunately, in general such a characterization will not be simultaneously necessary and sufficient to satisfy the original constraints on the descriptor variables.

Consider the descriptor variable relationships on x(k), x(k+1), and u(k):

$$E_{k+1} x(k+1) - A_k x(k) = B_k u(k) \qquad (5.62)$$

Suppose there are constraints on x(k) and x(k+1) of the form

$$F_k x(k) = c_k \qquad (5.63)$$

$$F_{k+1} x(k+1) = c_{k+1} \qquad (5.64)$$

Obviously if x(k) and x(k+1) are to satisfy (5.63) and (5.64), u(k) must be selected such that (5.62) can also be satisfied. The observation motivates

the following lemma:

*Lemma 5.9:* Suppose there exist vectors $\alpha$, $\beta_1$, and $\beta_0$ such that

$$\alpha' E_{k+1} = \beta_1' F_{k+1} \tag{5.65}$$

$$\alpha' A_k = \beta_0' F_k \tag{5.66}$$

then constraints (5.63) and (5.64) can be satisfied only if there exists u(k) such that

$$\alpha' B_k u(k) = \beta_1' c_{k+1} - \beta_0' c_k.$$

*Proof:* Suppose no such u(k) exists. By taking a linear combination of the equations in (5.62), weighted by $\alpha$, the left-hand side of (5.62), becomes

$$\beta_1' c_{k+1} - \beta_0' c_k$$

by (5.63) and (5.64). By assumption, there will be no u(k) satisfying the new right-hand side, and therefore (5.63) and (5.64) are inconsistent with the descriptor system. ■

For any linear descriptor system, the set of vectors $(\alpha, \beta_1, \beta_0)$ satisfying (5.65) and (5.66) form a subspace, namely the row vectors orthogonal to the column vectors of the matrix

$$\begin{bmatrix} E_{k+1} & A_k \\ -F_{k+1} & \\ & -F_k \end{bmatrix} \tag{5.67}$$

Any element in this space can be expressed as a linear combination of elements that form a basis for the subset. Therefore, to insure that the lemma is satisfied, it is only necessary to check the elements in the basis. This generates the following theorem:

*Theorem 5.10:* Given the descriptor equations (5.62), the equality constraints on x(k) and x(k+1) given by (5.63) and (5.64) can be satisfied only if

$$T_k B_k u(k) = V_k c_{k+1} - W_k c_k$$

where the rows of

$$[T_k \,\vdots\, V_k \,\vdots\, W_k]$$

form a basis for all null row vectors to (5.67).

Thus, Theorem 5.10 gives necessary but not sufficient conditions on u(k) to meet the constraints on x(k) and x(k+1). More importantly, these conditions are on specific input vectors, as opposed to the procedure yielding necessary and sufficient conditions at the beginning of the section. The theorem serves as a test of the feasibility of the constraints as well as a guideline for the selection of u(k). This result also applies, in a slightly more complex form, to inequality constraints on descriptor vectors.

The concept of maintainability has interesting associations with other theoretical research in dynamic systems. The notion of system invertibility [22] for an input-output system implies that the system is structured such that the roles of the inputs and outputs can be reversed. A weakened form of invertibility holds if for any trajectory of outputs there will exist some trajectory of inputs that achieves those outputs. Conceptualizing the given outputs as constraints, this latter property corresponds to complete maintainability.

Bryson and Ho [23] discuss the optimization of continuous-time systems with state equality constraints. Assuming differentiability, they indicate one method for computing the optimal control that involves alternately differentiating the constraints with respect to time and substituting the system equation, repeating this until all of the constraints have an explicit dependence on the inputs. This procedure is analogous to the discrete-time algorithm for transforming a completely maintainable system to a uniformly completely maintainable system in Section 5.3, where constraints are shifted backward in time until they explicitly depend on the inputs. From this study of linear, discrete-time systems it was also desirable that the combination of inputs on which the constraints depend constitute a linearly independent set. The same condition is desirable in the continuous-time case.

In a state-space system, a state from which a system is capable of satisfying feasibility constraints over a finite interval of time has been defined as a *mobile state* [24]. If any feasible state at any time instant is a mobile state, the system will be uniformly completely maintainable. If a sequence of mobile states can be achieved from any initial state, the system is completely maintainable. Thus, state mobility is in some sense a dual property of complete maintainability.

Schlueter and Levis [25] assumed a full row rank property similar to the sufficient condition given for uniform complete maintainability, in the context of sampled-data processes. The goal here was to derive an optimal adaptive regulator where one or more of the outputs was specified by the sampling criterion. Their full row rank property, called the strong sampling constraint criterion, was sufficient to guarantee the existence of a feasible control.

The study of maintainability in Sections 6.4 and 6.5, like the related notions above, applies primarily to the regular case. The nonregular case is also of importance, however, as evidenced in the growing literature on noncausal economic models. If the combination of a macroeconomic model and a set of long-range policy objectives does not constitute a maintainable system, the need for more realistic objectives is indicated. Furthermore, a systemmatic reformulation of systems that are maintainable can identify an equivalent, but more behaviorally consistent, set of objectives.

# VI. SPATIAL DYNAMIC PROGRAMMING

## 6.1 INTRODUCTION

Spatial dynamic programming (SDP) is a promising new method that is a hybrid of two well-established optimization techniques: dynamic programming and coordination of decomposed subsystems. The underlying idea of decomposition is that the problem is more tractable if one performs a set of smaller optimization problems, and then uses a coordination scheme to account for interactions between subsystems. A difficulty with many decomposition approaches is that they entail an iterative scheme for revising the coordination variables that may converge slowly. The theme of SDP is to optimize across subsystems in the same manner as one optimizes across time increments in classical dynamic programming. Thus, one can derive the optimal solution through a double sweep of the subsystems, while retaining the feature of solving a set of smaller optimization problems.

In order to motivate a clear understanding of SDP, consider briefly the classical state-space dynamic programming procedure. Initially one performs a backward sweep through time, determining the optimal input corresponding to a given state at a given time by minimizing the cost of the input (and state) at that instant plus the optimal cost-to-go from the state resulting from that input at the next time instant. Thus, one solves a series of optimization problems, each successive set corresponding to an incrementally longer time span, until the inputs for the entire time horizon have been optimized. This sequential process allows the embedding of previously solved and slightly smaller problems, vastly simplifying each individual problem. Once one has propagated the procedure back to the initial time instant, a forward sweep is used to recover the optimal input trajectory.

SDP proceeds in an analogous fashion for a group of interconnected subsystems. First, all of the subsystems are arranged in a sequence. The criterion for the first subsystem is optimized subject to a given set of total inputs from and individual outputs to the other subsystems that are specified as parameters in the optimization. Next, the second subsystem is joined with the first subsystem to create a new "composite" subsystem. The performance criterion for this composite subsystem is optimized subject to a given set of total inputs to each of the first two subsystems from outside the composite system, and a given set of the combined outputs from the composite system to each of the other subsystems. This set is specified as parameters. This optimization can be simplified by embedding the parameterized optimization for the first subsystem. The procedure is repeated in a similar fashion so that at the $k^{th}$ step, the $k^{th}$ subsystem in the sequence is added to the previous k-1 subsystems to again create a new composite subsystem. The performance criterion corresponding to this new composite subsystem is optimized subject to (1) given net inputs to each subsystem inside the composite from those outside the composite system, and (2) given net outputs from the composite system that affect each remaining subsystem. Again, this optimization is eased by embedding the parameterized optimization performed at the $(k-1)^{th}$ step. Once this procedure has been completed for all subsystems, a global optimum will be achieved, and the optimal control inputs can be recovered via a backward sweep through the sequence of subsystems.

The next section establishes the global optimality of SDP for a general class of optimization problems. The following section describes the more structured case of optimizing interconnected dynamic systems where the inputs to a subsystem from other subsystems are additive and the performance criteria are additive. The chapter concludes with a discussion of SDP characteristics and the application of SDP to the problem of optimal power flow.

## 6.2 OPTIMALITY OF SPATIAL DYNAMIC PROGRAMMING

If an optimization problem can be formulated as a mathematical programming problem that is weakly decomposable, a spatial dynamic programming algorithm can be applied if implemented in a manner consistent with the decomposition. A proof that spatial dynamic programming will yield the optimal solution in such cases is demonstrated below by showing that the spatial dynamic programming formulation is equivalent to the original mathematical programming problem. (For related disucssions, see [26]-[30].)

Suppose the problem is to minimize the function $F(\underline{u}_1,\ldots,\underline{u}_N)$, where $\underline{u}_1,\ldots,\underline{u}_N$ are each input vectors, subject to the constraint that $(\underline{u}_1,\ldots,\underline{u}_N)$ fall in some constraint set V. Assume this problem is *weakly decomposable,* meaning that for any integer k, $1 \leq k \leq N-1$, the problem

$$J_k(\hat{\underline{u}}_{k+1},\ldots,\hat{\underline{u}}_N) = \min_{(\underline{u}_1,\ldots,\underline{u}_N)\in V} F(\underline{u}_1,\ldots,\underline{u}_N) \qquad (6.1)$$

$$\text{Subject to } (\underline{u}_{k+1},\ldots,\underline{u}_N) = (\hat{\underline{u}}_{k+1},\ldots,\hat{\underline{u}}_N)$$

is well-defined for each $(\hat{\underline{u}}_{k+1},\ldots,\hat{\underline{u}}_N)$ in the projection of V on $U_{k+1}x\ldots xU_N$. For $k=N$, define $J_N(0)$ to be the original optimization problem. The decomposibility assumption implies that optimization of the original objective function while holding $\underline{u}_{k+1},\ldots,\underline{u}_N$ fixed is meaningful for all k. This feature provides the separability necessary to perform spatial dynamic programming.

Due to the weak decomposibility for all k, it is clear that problem (6.1) is equivalent to the problem

$$\min_{(\underline{u}_1,\ldots,\underline{u}_N)} F(\underline{u}_1,\ldots,\underline{u}_N) \qquad (6.2)$$

$$\text{Subject to } (\underline{u}_{k+1},\ldots,\underline{u}_N) = (\hat{\underline{u}}_{k+1},\ldots,\hat{\underline{u}}_N)$$

$$\text{and } (\underline{u}_1,\ldots,\underline{u}_{k-1}) = \arg\min_{(\underline{u}_k,\hat{\underline{u}}_{k+1},\ldots,\hat{\underline{u}}_N)} J_{k-1}$$

This equivalence follows from the observation that the solution to (6.1) must have components $\tilde{\underline{u}}_1,\ldots,\tilde{\underline{u}}_{k-1}$ that form a solution to $J_{k-1}(\tilde{\underline{u}}_k, \underline{u}_{k+1},\ldots,\underline{u}_N)$. Thus, by first solving $J_{k-1}$ over all feasible $(\underline{u}_k,\hat{\underline{u}}_{k+1},\ldots,\hat{\underline{u}}_N)$, the optimization in (6.1) is reduced to an optimization over the set of $\underline{u}_k \varepsilon U_k$ corresponding to the projection on $U_k$ of all elements in V for which $(\underline{u}_{k+1},\ldots,\underline{u}_N) = (\hat{\underline{u}}_{k+1},\ldots,\hat{\underline{u}}_N)$.

The subproblems defined by Formulation (6.2) form the basis for constructing a sequence of recursive optimizations to solve the original problem by spatial dynamic programming. First $J_1(\hat{\underline{u}}_2,\ldots,\hat{\underline{u}}_N)$ is determined for each $(\hat{\underline{u}}_2,\ldots,\hat{\underline{u}}_N)$ in the projection of V on $U_2 x \ldots x U_N$. Next, Problem (6.2) is solved for k=2 for each feasible $(\hat{\underline{u}}_3,\ldots,\hat{\underline{u}}_N)$, embedding the optimization results for $J_1$ and optimizing over $\underline{u}_2$. This procedure is repeated for k=3,...,N, recursively, embedding the solutions for $J_{k-1}$ and optimizing over $\underline{u}_k$. Since $J_N(0)$ corresponds to the original optimization problem, the solution to $J_N(0)$ produces the optimal solution $\underline{u}_1^*,\ldots,\underline{u}_N^*$ and the optimal objective function value $F(\underline{u}_1^*,\ldots,\underline{u}_N^*)$.

In addition to determining the optimal solution, the embedding principle of SDP will generally allow a very large optimization to be handled as a sequence of tractable optimizations. However, the SDP algorithm will vary in ease and efficiency, depending on the grouping and sequencing of the input variables. The subproblems defined by (6.2) will rarely need to be determined for each feasible $(\hat{\underline{u}}_{k+1},\ldots,\hat{\underline{u}}_N)$; usually an entire class of $(\hat{\underline{u}}_{k+1},\ldots,\hat{\underline{u}}_N)$ will have the same solution. In such cases the class of these configurations can be usually represented by a vector $\underline{z}_k$ which has far fewer components than $(\underline{u}_{k+1},\ldots,\underline{u}_N)$. This reduces substantially the complexity of each stage of SDP. Furthermore, these $\underline{z}_k$ vectors can be observed quite naturally in most large-scale systems: if each $\underline{u}_k$ vector corresponds to inputs associated with a particular subsystem, then the $\underline{z}_k$ vector will correspond to the "interactions" between the subsystems 1,...,k with subsystems k+1,...,N. A particularly convenient representation of the $\underline{z}_k$ vectors can be obtained when the interactions are additive, as is discussed in detail in the next section.

As a brief illustration, consider the three variable optimization problem:

$$\max_{u_1,u_2,u_3} u_1u_2u_3$$

$$\text{Subject to } u_1^2+u_2^2+u_3^2 \le 1$$

$$u_i \ge 0 \quad \text{all } i$$

This problem is weakly decomposable in $u_1$, $u_2$, and $u_3$. The solution via SDP is described by the following sequence of three subproblems:

*Subproblem 1:*

$$J_1(\hat{u}_2,\hat{u}_3) = \max_{u_1} u_1\hat{u}_2\hat{u}_3$$

$$\text{Subject to } u_1^2 \le 1 - \hat{u}_2^2 - \hat{u}_3^2$$

$$u_1 \ge 0$$

$$= (1-\hat{u}_2^2-\hat{u}_3^2)^{1/2}\,\hat{u}_2\,\hat{u}_3$$

$$\tilde{u}_1 = (1-\hat{u}_2^2-\hat{u}_3^2)^{1/2}$$

*Subproblem 2:*

$$J_2(\hat{u}_3) = \max_{u_2} (1-u_2^2-\hat{u}_3^2)\,u_2\hat{u}_3$$

$$\text{Subject to } u_2^2 \le 1-\hat{u}_3^2$$

$$u_2 \ge 0$$

$$= 1/2(1-\hat{u}_3^2)\,u_3$$

$$\tilde{u}_2 = \sqrt{1/2(1-\hat{u}_3^2)}$$

$$\tilde{u}_1 = \sqrt{1/2(1-\hat{u}_3^2)}$$

*Subproblem 3:*

$$J_3(0) = \max_{u_3} 1/2\,(1-u_3^2)u_3$$

$$\text{Subject to } u_3^2 \le 1$$

$$u_3 \ge 0$$

$$= 1/9\,\sqrt{3}$$

$$u_3^* = 1/3\,\sqrt{3}$$

$$u_2^* = 1/3\,\sqrt{3}$$

$$u_1^* = 1/3\,\sqrt{3}$$

This simple example could have been solved readily in a centralized manner, using the Kuhn-Tucker Theorem. However, note that the Kuhn-Tucker conditions for the overall problem would form seven equations with seven unknowns, while the Kuhn-Tucker conditions for each of the above subproblems consist of only three equations and

three unknowns. This difference suggests how SDP can perform a large-scale optimization in a computationally tractable manner by optimizing a sequence of smaller subproblems.

## 6.3 DESCRIPTION OF THE TECHNIQUE FOR DYNAMIC LARGE-SCALE SYSTEMS WITH ADDITIVE INTERCONNECTIONS

Interconnected dynamic subsystems possess a natural spatial structure that can often be exploited effectively with SDP. The purpose here is to illustrate how these special structural features affect the SDP procedure by considering the following minimization problem for a large system composed of m subsystems to be optimized over N time periods:

$$\bar{J} = \min_{\bar{u}_1,\ldots\bar{u}_m} \left\{ \sum_{i=1}^{m} L_i(\bar{x}_i,\bar{u}_i) \right\}$$

where

$$\bar{x}_i = (x_i(0),\ x_i(1),\ldots,x_i(N))$$

$$\bar{u}_i = (u_i(0),\ u_k(1),\ldots,u_i(N-1))$$

and

$$x_i(k+1) = f_i(x_i(k),u_i(k)) + \sum_{j\neq i} g_{ji}(x_j(k)),$$

$$k = 0,\ldots,N-1$$

The use of a discrete time-invariant system in this problem is employed merely for convenience and is not a requirement for SDP. Define

$$\bar{y}_{ji} = (g_{ji}(x_j(0)),g_{ji}(x_j(1)),\ldots,g_{ji}(x_j(N-1)))\ j\neq i$$

to be the trajectory of inputs to subsystem i resulting from outputs from subsystem j, and define

$$\bar{y}_{ii} = -\sum_{j\neq i} \bar{y}_{ji}$$

to be the negative of the trajectory of total inputs to subsystem i from other subsystems. Clearly,

$$\sum_{j=1}^{m} \bar{y}_{ji} = 0$$

for all i. Note that in practice $g_{ji}(x_j(k))$ is often equal to zero for many combinations of i,j, and k. This not only simplifies the interconnection structure, but also increases the efficiency of SDP, since SDP is best suited for systems with sparse (but not necessarily weak) interactions.

A diagram of the sequence of subsystems and the general interconnection structure appears in Figure 6.1. Note that the outputs from each subsystem (i.e., $\{\bar{y}_{ji} \mid i \neq j\}$ for subsystem j) leave verically (either upward or downward) and are summed into a "bus" with other outputs headed for the same subsystem. The buses appearing above the subsystem boxes run to the left and represent inputs to a subsystem resulting from outputs of subsystems that follow that particular subsystem in the sequence. The buses below the subsystem boxes run to the right and represent inputs to a particular subsystem from other subsystems preceding in the sequence.

Note that if an imaginary vertical line is drawn between any two subsystems you will cross exactly m buses. In particular, if the line is drawn between subsystem k and subsystem k+1 (see Figure 6.2) the line will cross k buses in the upper set and m-k buses in the lower set. In SDP the value of the outputs in each bus *at the point where the line intersects the bus* is used as a parameter ($z_i^k$ or $-z_i^k$) for the $k^{th}$ step optimization in the SDP procedure. This figure illustrates the physical interpretation of those parameters: If subsystems 1 to k are treated as a composite, then the upper buses running between subsystems k and k+1 will carry the value of the net inputs to each subsystem in the composite from subsystems not in the composite; the lower buses will carry the value of net output from the composite subsystem to each subsystem not included in the composite. As was observed, a k increases, the number of "input buses" increases and the number of "output buses" decreases, yet the sum of the two is always m.

Now we proceed with the algebraic description of SDP for this problem. Define

$$J_i(\bar{z}_1^i,\ \bar{z}_2^i,\ldots,\bar{z}_m^i)$$

to be equal tothe optimal cost of controlling the composite system created at the $i^{th}$ step of the procedure subject to parameters $\bar{z}_j^i$, where

$$\bar{z}_j^i = \begin{cases} -\sum_{k=i+1}^{m} \bar{y}_{kj} & j \leq i \\ \sum_{k=1}^{i} \bar{y}_{kj} & j > i \end{cases}$$

represents the negative of the net input to the $j^{th}$ subsystem in the composite from all subsystems outside the composite for $j \leq i$, and represents the net output of the composite subsystem to the $j^{th}$ subsystem for $j > i$. As will be seen below, inputs are assigned a negative value merely for algebraic convenience.

We now consider the m-step process which successively adds subsystems to a composite system optimization until all subsystems have been included.

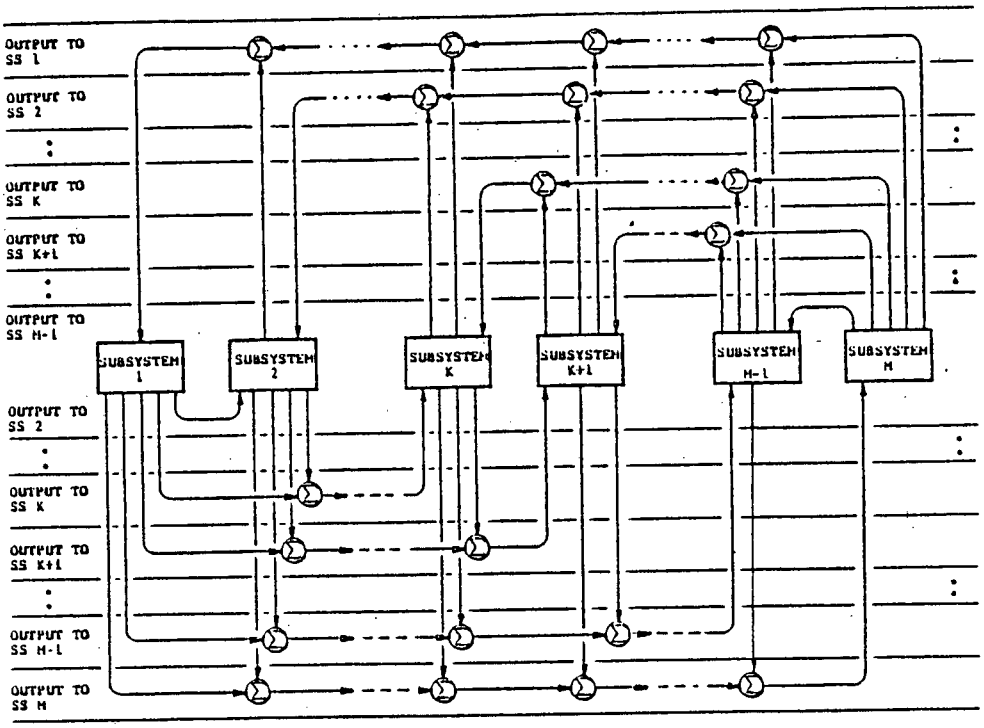



Figure 6.1. Large-Scale System with M Sequentially-Arranged Subsystems

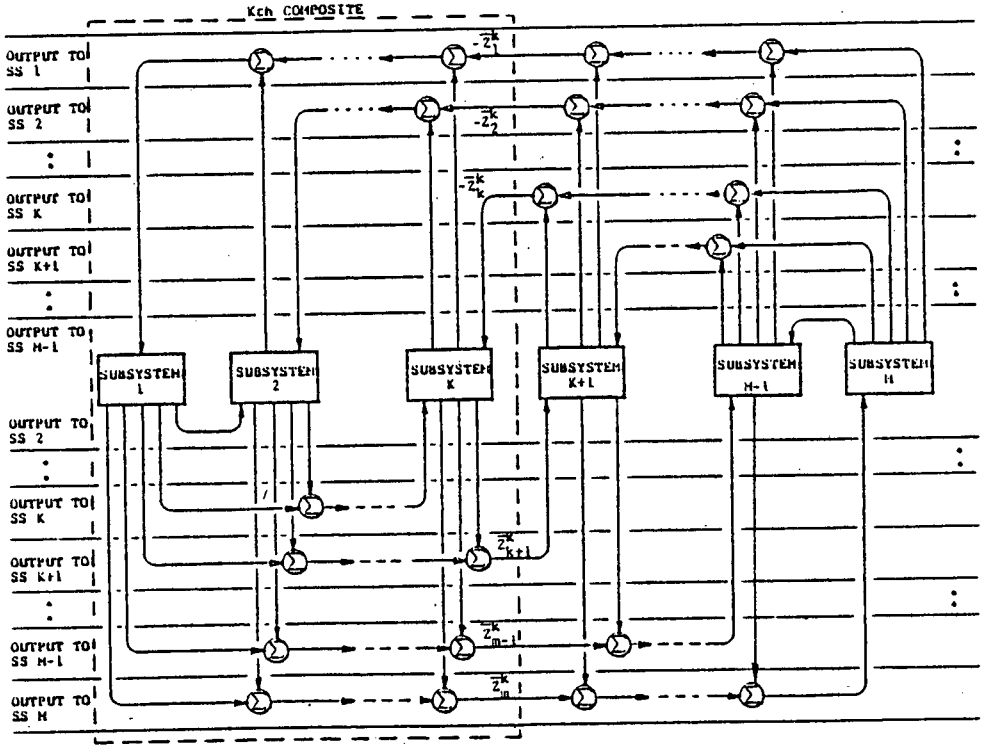



Figure 6.2. $K^{th}$ Composite System and Interaction Parameters

*Step 1*

In the first step, the first composite subsystem is equivalent to subsystem 1 (see Figure 6.3). The optimization problem is to determine

$$J_1(\bar{z}_1^1, \bar{z}_2^1, \ldots, \bar{z}_m^1) = \min_{\bar{u}_1} L_1(\bar{x}_1, \bar{u}_1)$$

$$\text{subject to } y_{ij} = \bar{z}_j^1, \text{ all } j.$$

Given the determination of a solution for any $\bar{z}_1^1, \bar{z}_2^1, \ldots, \bar{z}_m^1$, we proceed to Step 2.

*Step 2*

We add subsystem 2 to subsystem 1 to create a new composite (see Figure 6.4) and determine

$$J_2(\bar{z}_1^2, \bar{z}_2^2, \ldots, \bar{z}_m^2) = \min_{\bar{u}_1, \bar{u}_2} \{L_1(\bar{x}_1, \bar{u}_1) + L_2(\bar{x}_2, \bar{u}_2)\}$$

$$\text{subject to } \bar{y}_{1j} + \bar{y}_{2j} = \bar{z}_j^2, \text{ all } j$$

Using the definition of $\bar{z}_j^2$ and the results of Step 1, this can be expressed

$$J_2(\bar{z}_1^2, \bar{z}_2^2, \ldots, \bar{z}_m^2) = \min_{\bar{u}_2, \bar{z}_1^1, \bar{z}_2^1, \ldots, \bar{z}_m^1} \left\{ L_2(\bar{x}_2, \bar{u}_2) + J_1(\bar{z}_1^1, \bar{z}_2^1, \ldots, \bar{z}_m^1) \right\}$$

$$\text{subject to } \bar{y}_{2j} = \bar{z}_j^2 - \bar{z}_j^1$$

Given the solution for any specified $\bar{z}_1^2, \bar{z}_2^2, \ldots, \bar{z}_m^2$, we proceed to Step 3.

For all k, $3 \leq k \leq m-1$, we have

*Step k*

We add subsystem k to the previous k-1 subsystems to create the $k^{th}$ composite (see Figure 6.2) and determine

$$J_k(\bar{z}_1^k, \bar{z}_2^k, \ldots, \bar{z}_m^k) = \min_{\bar{u}_1, \ldots, \bar{u}_k} \left\{ \sum_{\ell=1}^{k} L_\ell(\bar{x}_\ell, \bar{u}_\ell) \right\}$$

$$\text{subject to } \sum_{\ell=1}^{k} \bar{y}_{\ell j} = \bar{z}_j^k, \text{ all } j$$

Using the definition of $\bar{z}_j^k$ and the results of Step k-1, this optimization problem can be expressed

$$J_k(\bar{z}_1^k, \bar{z}_2^k, \ldots, \bar{z}_m^k) = \min_{\bar{u}_k, \bar{z}_1^{k-1}, \bar{z}_2^{k-1}, \ldots, \bar{z}_m^{k-1}} \left\{ L_k((\bar{x}_k, \bar{u}_k) + J_{k-1}(\bar{z}_1^{k-1}, \ldots, \bar{z}_m^{k-1}) \right\}$$

$$\text{subject to } \bar{y}_{kj} = \bar{z}_j^k - \bar{z}_j^{k-1}, \text{ all } j$$

Given the solution for all $\bar{z}_1^k, \ldots \bar{z}_m^k$, we proceed to Step k+1.

*Step m*

All m systems are now conceptually combined into a composite. Determine

$$J_m(\bar{z}_1^m, \bar{z}_2^m, \ldots, \bar{z}_m^m) = \min_{\bar{u}_1, \ldots, \bar{u}_m} \left\{ \sum_{\ell=1}^{m} L_\ell(\bar{x}_\ell, \bar{u}_\ell) \right\}$$

$$\text{subject to } \sum_{\ell=1}^{m} \bar{y}_{\ell j} = \bar{z}_j^m, \text{ all } j$$

We note two things. First, since

$$\sum_{\ell=1}^{m} \bar{y}_{\ell j} = 0$$

we must have $\bar{z}_j^m = 0$ for all j. Second, by definition,

$$J_M(0,0,\ldots,0) = \bar{J}$$

So this is the global system optimization criterion:

$$\bar{J} = \min_{\bar{u}_1, \ldots, \bar{u}_m} \left\{ \sum_{\ell=1}^{m} L_\ell(\bar{x}_\ell, \bar{u}_\ell) \right\}$$

$$\text{subject to } \sum_{\ell=1}^{m} y_{\ell j} = 0, \text{ all } j$$

By definition of $\bar{z}_j^{m-1}$ and Step m-1 (see Figure 6.5), we can reformulate the problem:

$$\bar{J} = \min_{\bar{u}_m, \bar{z}_1^{m-1}, \bar{z}_2^{m-1}, \ldots, \bar{z}_m^{m-1}} \left\{ L_m(\bar{x}_m, \bar{u}_m) + J_{m-1}(\bar{z}_1^{m-1}, \bar{z}_2^{m-1}, \ldots, \bar{z}_m^{m-1}) \right\}$$

$$\text{subject to } \bar{y}_{mj} = -\bar{z}_j^{m-1}, \text{ all } j$$

The solution of Step m provides the optimal $\bar{u}_m, \bar{z}_1^{m-1}, \ldots, \bar{z}_m^{m-1}$, which by Step m-1 identifies the optimal $\bar{u}_{m-1}, \bar{z}_1^{m-2}, \ldots, \bar{z}_m^{m-2}$, etc. Be repeating this procedure backward through the successive steps the entire set of optimal input trajectories $\bar{u}_m, \bar{u}_{m-1}, \ldots, \bar{u}_1$ are obtained.

An important observation is that the above procedure exploits the fact that inputs to each subsystem from the other subsystems are additive. This permits the above definition of $\bar{z}_j^k$ parameters that efficiently summarize the net effect of a potentially large number of interactions between subsystems inside and outside the defined composite system. The embedding of composite subsystems into slightly larger composite subsystems serves to simplify the optimization relative to the complex network of interactions *inside* the

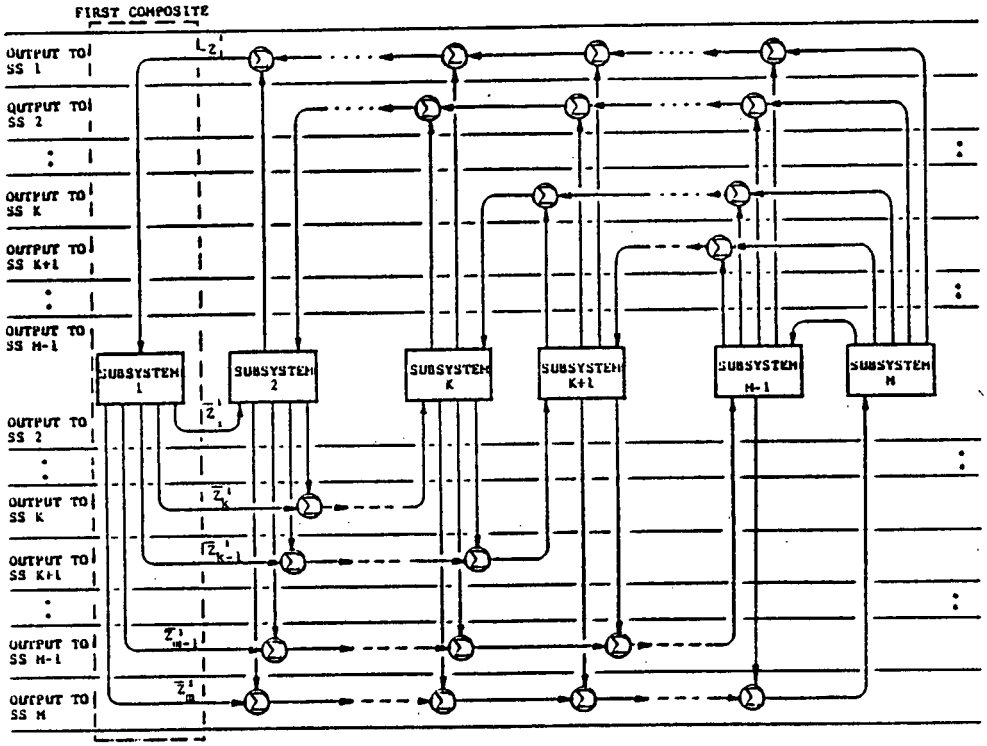



Figure 6.3. First Composite System and Interaction Parameters

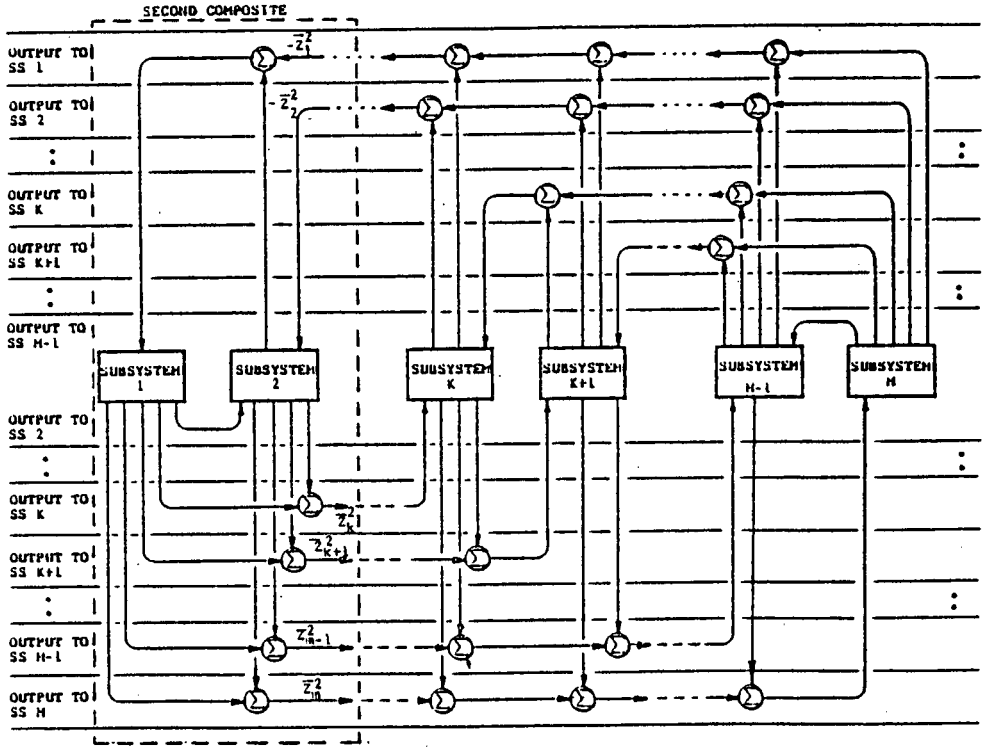



Figure 6.4. Second Composite System and Interaction Parameters.

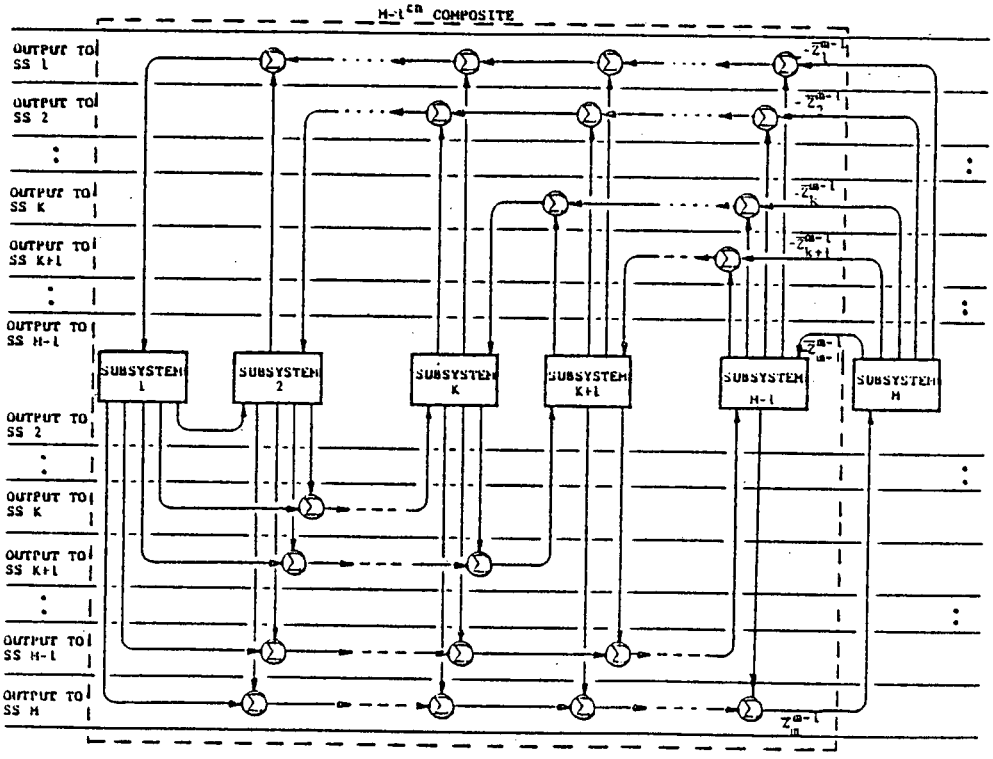



Figure 6.5. $(m-1)^{th}$ Composite System and Interaction Parameters.

composite. The procedure is most efficient when the interactions are sparse, but it does not require that the nonzero interactions be weak.

## 6.4 CHARACTERISTICS OF SPATIAL DYNAMIC PROGRAMMING

Spatial dynamic programming provides an extremely powerful approach to the optimization of a set of interconnected subsystems. Basically, the total optimization problem is decomposed into two types of problems - optimization of the subsystem itself and optimization of the interactions of the subsystem with the remainder of the overall system. In the subsystem optimization problem, it is necessary to find a solution for every possible sequence of interaction variables; thus, the effective dimensionality of the subsystem problem is the dimensionality of the subsystem itself plus the total number of interaction variables associated with the subsystem. If the performance criterion meets the weak decomposibility of Section 6.2, a global optimum is found by first solving each subsystem problem and then sequentially optimizing the interactions of each subsystem with the rest of the system.

The computational implications of this approach are quite impressive. Basically, if the number of interaction variables is small compared to the dimensionality of the subsystems, the solution of the total problem, which has a number of state variables equal to the sum of the dimensionalities of the individual subsystems, is reduced to solving a sequence of sub-problems, each of which has dimensionality on the order of the dimensionality of the subsystems. Because of the well-known exponential growth of computational requirements with respect to system dimensionality, this reduction is extremely significant. For example, consider a system consisting of 10 interconnected subsystems, each of which has 10 state variables and 3 interaction variables. If the problem were solved in a straightforward manner, there would be 100 state variables, a very formidable problem. On the other hand, using spatial dynamic programming, the solution can be obtained by solving 10 subsystem problems, each with an effective dimensionality of 13, and 10 interaction optimization problems, each with an effective dimensionality of 3; clearly, these problems are much more manageable. Note that this great reduction in dimensionality is obtained with no requirement for iteration and that a global optimum is guaranteed with no assumption on system equations, constraints, or performance criterion, other than the decomposability of the problem.

In addition to the great reduction in dimensionality with no loss of optimality, the technique is ideally structured for decentralized control and distributed data processing. Note that there is no requirement for accumulating information about the total state of the system at a central point. Thus, the optimal control of the total system can be implemented by a series of local controllers, one for each subsystem,

each of which communicates only with the other subsystems with which it interacts. Each local controller solves the subsystem problem parameterized on its interaction variables. The interaction variables are then optimized by proceeding through the subsystems in a fixed sequence; this sequence can be pre-determined, so that, in real-time, the local controllers can solve the interaction problem in the proper order. The only communication requirements are for sending the minimum cost function as a function of the interaction variables to interconnected subsystems. Because of these relatively modest computational and communication requirements, the technique is potentially suitable for implementation in a network of minicomputers or microprocessors connected by communication links.

The technique also provides an extremely attractive approach for responding to subsystem failures. If a subsystem becomes disabled for some reason, then it is only necessary to inform the interconnected subsystems, specify a new sequence of subsystems for computing the interaction variables (this could be pre-determined for all possible failure modes), and then proceed as before. In this manner the new system structure is automatically optimized, without any additional computations. This opens up a whole new approach to optimizing systems that are subject to subsystem failure.

The basic spatial dynamic programming procedure also has a number of important theoretical implications. Because it always obtains a global optimum, it provides a very useful tool for evaluating other decomposition techniques and for proving theorems about their properties. In particular, the dynamic programming successive approximation technique developed by Larson and Korsak [31] has a number of similarities to spatial dynamic programming; essentially, this technique fixes all the interaction variables according to an initial policy and iteractively optimizes one subsystem at a time. The successive approximation technique thus has lower computational requirements than spatial dynamic programming, but it is only globally optimal under certain conditions. By exploiting the similarity of the two techniques, it should be possible to develop algorithms that retain the computational simplicity of the former and the global optimality of the latter.

It should be noted that the theoretical value of spatial dynamic programming is not limited to techniques that use dynamic programming to solve the sub-problems. Basically, the ideas of dynamic programming are necessary only in the decomposition into subsystem problems and interaction variable problems. Any technique can be used to solve the subsystem problem and/or the interaction variable problem. This observation greatly increases the class of decomposition techniques that can be analyzed in terms of spatial dynamic programming.

Descriptor variable representation plays a critical role in spatial dynamic programming. The subsystem problems must be solved for all possible sequences of interaction variables. For any particular set of sequences of these variables, the problem has constraints on both inputs and outputs. These constraints introduce static relationships of the type that require descriptor variable representation. Thus, even if the subsystem itself is a state-space representation, the subsystem problem can be expressed via descriptor variables. Of course, even more generality can be allowed in spatial dynamic programming by permitting the subsystems themselves to be in descriptor variable form.

The results in Chapter V on the analysis of linear-quadratic descriptor variable systems are particularly relevant to spatial dynamic programming. If the individual subsystems are linear state-variable systems and if the performance criterion is quadratic, then the results developed there can be applied to show that the solutions to the sub-problems consist of a linear function of the state variables plus a linear function of the interaction variables. This greatly facilitates the computations for these systems, both for the subsystem problem and for the interaction variable problem.

Spatial dynamic programming is also an effective procedure for large-scale systems without explicitly defined subsystems. In this sense, SDP provides a technique for decomposing large mathematical programming problems. To use SDP, each variable must be assigned to one of a sequence of vectors to create weak decomposibility. In many cases, this decomposition can be effectively created by first decomposing the constraints into constraint subsets. After arranging the constraint sets into a sequence, a corresponding sequence of variable vectors is determined in the following manner: variable vector k will contain all variables that affect *only* constraint sets $1,\ldots,k$ and that are *not* in vectors $k-1,k-2,\ldots,1$. This decomposition leads to a natural definition of the constraint sets for each subproblem in the SDP algorithm.

The efficiency of SDP relies heavily on the decomposition of the variables into input vectors and on their sequencing. The existence of an efficient decomposition and sequencing obviously depends on the structure of the system. However, the efficiency will vary significantly for different decompositions of the same system. Some principles of effective decomposition and sequencing have been recognized. As stated earlier, the decomposition is most effective when the interconnections can be summarized by a relatively sparse set of variables. Also, if one subsystem affects a second subsystem but not vice-versa, the latter subsystem should precede the former in the sequence. The recognition of additional principles is an important research area for spatial dynamic programming.

## 6.5 OPTIMAL POWER FLOW USING SPATIAL DYNAMIC PROGRAMMING

An important problem in power system operation is the determination of control parameters that minimize the costs of generation, maintenance, load curtailment, etc.. The controls typically include torque angles and voltage magnitudes at network nodes, transformer tap settings, and the network topology. Since power networks are usually sparse and the fundamental laws governing power flow have the weak decomposability property, spatial dynamic programming is naturally suited for the optimal power flow problem. This section will formulate the optimal flow problem and describe an example network for which this problem was solved using SDP. Some promising extensions of this application of SDP to power system analysis are also discussed.

The problem formulation presented here assumes a network of N nodes, which can be load nodes and/or generation nodes. The system equations correspond to a balance of real and reactive power flows at each node. These equations are standard in power systems literature (see for example Elgerd [32]):

$$P_i - jQ_i = \sum_{k=1}^{N} Y_{ik} V_i V_k^* \qquad i=1,\ldots,N \qquad (6.3)$$

where

$j = \sqrt{-1}$

$*$ denotes complex conjugation

$P_i$ = the net real power flow at node i

$Q_i$ = the net reactive power flow at node i

$Y_{ik}$ = the admittance of the branch connecting nodes i and k

$V_i$ = (complex) voltage at node i
$= |V_i| e^{j\delta_i}$

$\delta_i$ = torque angle at node i

Upper and lower bounds are specified for each $P_i$, $Q_i$, $|V_i|$, and $\delta_i$. Constraints are also imposed on the power flows over individual branches or on the differences between the angles at the connected nodes.

A given set of real and reactive power loads may be satisfied by zero, one, or multiple configurations of the node angles and voltage magnitudes. When there are several possible configurations, a cost criterion will generally allow the determination of a best set of controls. For most applications it is reasonable to assume a cost criterion of the form

$$\sum_{i=1}^{N} C_i$$

where the function $C_i$ depends only on variables associated with node i or branches incident to node i.

The system equations (6.3) and the additive cost criterion possess the weak decomposability to guarantee that spatial dynamic programming will determine the optimal controls. SDP is efficient for this problem because most power system networks are sparse in the sense that the number of branches is small compared to the total number of branches possible. Since the network is sparse, the number of variables processed at each stage of the solution will be small relative to the total number of variables.

The SDP algorithm for solution of the optimal power flow problem has been translated into a computer program for power system models that:

(1) incorporates branch flow constraints

(2) allows arbitrary generation costs at each node

(3) solves the real power flow equations (with losses)

(4) assumes arbitrary, but fixed, node voltage magnitudes

(5) permits an arbitrary order for processing the nodes

(6) accepts upper and lower bounds on all torque angles and generations.

The cost criterion is the sum of all generation costs at the nodes where generation can occur. The SDP algorithm proceeds by adding a new node in each iteration to the subset of nodes serving as the composite in the previous iteration. The interaction variables for a subproblem are the torque angles corresponding to (1) nodes that are in the composite and directly linked to a node that is not in the composite or (2) nodes that are not in the composite and are directly linked to nodes in the composite. Due to the nonlinearity of the system equations, the interactions are not additive.

This SDP algorithm was successfully applied to the seven node example illustrated in Figure 6.6, which has four load nodes, two generation nodes, and one node with both generation and load. The generation costs were assumed to be quadratic functions of the real power generation:

$$C(G_1) = 10\, G_1^2 + 5\, G_1 + 2$$

$$C(G_5) = 13.32\, G_5^2 + 7\, G_5 + 5$$

$$C(G_7) = 8\, G_7^2 + 4\, G_7 + 3$$

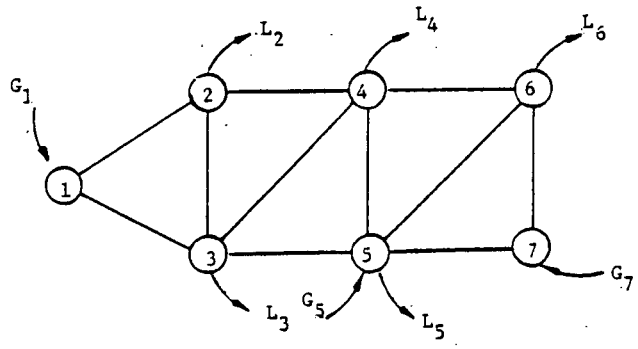



Figure 6.6. Seven Node Example for Optimal Power Flow Problem

The given real power loads at nodes 2 through 6 were

$L_2 = 70$

$L_3 = 30$

$L_4 = 100$

$L_5 = 160$

$L_6 = 40$

All node voltage magnitudes were set equal to one. The admittances were

$$Y_{ik} = \begin{cases} 1000 & \text{if } i \neq k \text{ and nodes } i \text{ and } k \text{ are directly linked} \\ 50 & \text{if } i = k \\ 0 & \text{otherwise} \end{cases}$$

Following the implementation of the SDP algorithm, the optimal torque angles were determined to be (in radians):

$\delta_1 = 0.08463$

$\delta_2 = 0.013$

$\delta_3 = 0.02406$

$\delta_4 = 0.0$

$\delta_5 = 0.02831$

$\delta_6 = 0.03437$

$\delta_7 = 0.1145$

The corresponding real generations were:

$G_1 = 132.1$

$G_5 = 100.4$

$G_7 = 166.1$

Note that there is a small difference between the sum of the real loads and the sum of the real generations. Of course, theoretically the difference does not exist; the numerical disparity is the result of the granularity of the grid used to discretize the continuous variables.

In addition to solving the optimal power flow problem, the SDP algorithm has other potential applications to power systems. For example, in the case of expansion planning, the dynamic programming procedure can be used to find the optimal combination of new generators lines that meets requirements and minimizes the overall costs. Also, since the SDP algorithm must account for feasibility constraints, the procedure is capable of determining all sets of loads that can be satisfied without violating some capacity constraint. Finally, by accommodating variation in the network topology due to potential outages, SDP can be applied in the evaluation of system reliability.

# VII. APPLICATION: NETWORKS OF ECONOMIC MARKETS

## 7.1 INTRODUCTION

Large economic systems are often analyzed as a coordinated set of supplier, producer, and consumer sectors. The classical approach has been to treat large economic networks as one large production system and determine the production flows that satisfy a given consumer demand at minimum cost. The drawback of this approach is that the demand behavior in responding to market conditions is ignored, so that these models do not determine a true market equilibrium.

The extensive use of the classical approach is certainly not due to the absence of a theoretical understanding of equilibrium. Neoclassical economics accommodates price-elastic demands and has produced significant achievements in characterizing the equilibrium of general closed economic systems (Debreu [33], Quirk and Saposnik [34], Arrow and Hahn [35]). However, the modeling and computation of general equilibrium is overwhelming, and usually the analysis concerns only a small piece of the general economy. Thus, practical considerations often suggest using *partial* equilibrium analysis as opposed to general equilibrium analysis. The distinguishing feature between partial and general analyses is that in a partial equilibrium analysis, a major portion of the flows in the economy are suppressed so that an "equilibrium" in the sectors of interest is more easily determined.

The recent focus, then, has been to develop approaches to partial equilibrium analysis, as evidenced by recent accomplishments in large-scale economic modeling (Brock and Nesbitt [36], Hoffman and Jorgenson [37], Hogan [38,39], Levis et. al. [40], Manne [41], Naill [42], Takayama and Judge [43]). Many of these efforts have represented the respective system as one large supplier-consumer market. While it can be argued that such a market exists conceptually, such representations generally are not convenient for describing the economic behavior of sectors within the system. Viewed at the level of sectors, an economic system is really a network of sectors, linked by markets between sectors where there are commodity flows.

It is common to conceptualize these markets as each having a supply function and a demand function, which together determine the equilibrium in that market. However, the existence of these constructs does not immediately follow from models of sector behavior. While supplier sectors may be characterized by supply functions and demand sectors may be characterized by demand functions, sectors producing intermediate goods in the system cannot be modeled by such constructs. Producers are simultaneously suppliers and consumers, with the quantities purchased being dependent on the amount of outputs that are sold.

Therefore, this intuitive notion of market supply and demand functions lacks analytical justification in complex networks of economically-based sectors. This chapter examines this justification for economic networks consisting of a chain of sectors, with good flows as illustrated in Figure 7.1.

In a network consisting merely of a single supply commodity undergoing a chain of transformation processes, under certain conditions it is straightforward to demonstrate that a supply function and demand function exist at each market between adjacent sectors. Since each supply function reflects cost of inputs plus cost of production, the supply function can essentially be determined from the supply function for the previous market in the chain. In this way, the supply function can be "propagated" from the supplier market to the consumer market. Similarly, one can "propagate" the demand function of consumers through the chain to the supply market. Thus, a supply and demand function exist at each market in the network.

This notion of propagating supply and demand functions has been employed in explaining more complex chain networks where sectors may use multiple inputs and have multiple outputs. In particular, a methodology called generalized equilibrium modeling (Cazalet [44]) has been described as conducting this propagation in a pointwise fashion. However, while this description is helpful in motivating an understanding of this algorithm, this argument has problems in explaining difficult cases where multiple types of inputs are used in a single output or an output can be used for multiple production and consumption purposes. Still, this intuitive idea of propagation is very appealing in its unification of the conceptual framework of supply and demand with a system description that is consistent with behavior at the sector level. This appeal encouraged the investigation of chain-structure networks, where by formulating the network model as a descriptor variable system, an analytical justification is established for certain classes of system models.

## 7.2 COMPOSITION AND STRUCTURE OF ECONOMIC NETWORKS

Economic systems are composed of a number of economic agents who either possess goods or are capable of performing services that either transform or transport these goods. These goods and services are exchanged, thereby creating a general market. The general market can be decomposed into markets for particular types of goods and services, for particular locations, or for particular times.

In making decisions or designing policies, one may be interested in assessing all the good flows and prices (or exchange rates) operating in the general market. Since these flows and prices

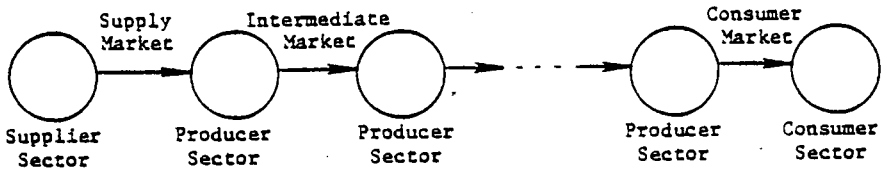



Figure 7.1. Economic Network Consisting of a Chain of Sectors

generally correspond to some balance of individual preferences, such an assessment is called a general equilibrium analysis. Often, however, one is interested in assessing flows and prices associated with only a set of particilar markets; this is a partial equilibrium analysis.

There is an important difference between general analysis and partial analysis models in the role of prices. In a well-defined general equilibrium model, prices reflect the substitution rate and are homogeneous of degree zero. This means that all prices could be multiplied by a uniform constant and the same flow of goods and services will result. Prices are generally not homogeneous of degree zero in partial equilibrium models because the prices must reflect the substitution between any goods or services endogenous to the model and goods and services outside the model. Since the exogenous flows and prices are assumed fixed in some sense, the homogeneity property is lost. However, the prices will still continue to reflect the exchange rate between goods endogenous to the model.

Any model intended to reflect actual markets assumes a considerable amount of aggregation, regardless of its size. This is an important point, because profit-maximization or utility-maximization by individual firms or consumers may not translate when the firms or consumers are aggregated. Therefore, the models may also reflect behavioral aspects that correspond to the inner dynamics of the aggregated set. This suggests the feasibility of models that make sense for aggregated sets without making sense at the level of the individual decision-maker.

Aggregation in these models is often done on the basis of activities rather than by individual persons. Therefore, one individual's behavior as a producer may be aggregated in one producer group, while his different types of consumption are reflected by separate aggregate consumption groups. We will refer to such groups as *sectors*. We distinguish three types of sectors:

*Suppliers* - Suppliers exchange goods for money. Supplier models relate supply flow to the price.

*Producers* - Producers purchase goods and sell goods. Producer models relate the inflow of goods, the price of inflowing goods, the outflow of goods, and the price of outflowing goods.

*Consumers* - Consumers exchange money for goods. Consumer models relate consumption flows to the price.

In general equilibrium models, the consumer sectors are also suppliers of some nature (e.g. providers of labor). In partial equilibrium models, there is no reason to assume consumers must supply some good that is endogenous to the model. Since we will be interested in the partial case, we will continue the discussion assuming suppliers and consumers to be separate.

If the model is sufficiently aggregated, one can examine the structure of the model, i.e., the pattern of the commodity flows between the sectors. It is useful to visually represent this structure as a *network*, and the convention is to use arrows to show the flow of goods along a *link*. It is implicitly understood that either money (or goods) must flow in the opposite direction since the link represents an exchange in this context.

For the purpose of this chapter, we distinguish *simple networks* from *looped networks*. A network is a looped network if it is possible to find a path following the directions of arrows from sector to sector, such that the path leaves and returns to the same sector. Otherwise, it is a simple network. Figure 7.2 shows a simple network, while Figure 7.3 displays a looped network.

Simple networks are an important case because the absence of loops accommodates a natural chain-structure corresponding to the goods markets, as illustrated in Figure 7.1. In some cases the chain-structure may not be immediate, but can be created readily by the following simple procedure:

1. Determine the longest chain of sectors, i.e., the path connecting the most sectors. Call the number of sectors N+2.

2. Assign each sector to an index i, $0 \leq i \leq N+1$, such that each sector providing it with goods is assigned to a lower index and each sector to which it provides goods is assigned to a higher index. Therefore, all supplier sectors will have index 0, all consumer sectors will have index N+1, and all producer sectors index j, where $1 \leq j \leq N$.

3. In general, the system grouped by index sets will still not have the chain-structure indicated on Figure 7.1, since a sector in sector set j could directly provide to a sector in set j+k, $k \geq 2$. To create the chain-structure one creates "dummy" sectors in the intervening sector sets: for any good that can flow between the sectors, the dummy sectors merely set the flow in equal to the flow out, and the unit price in equal to the unit price out.

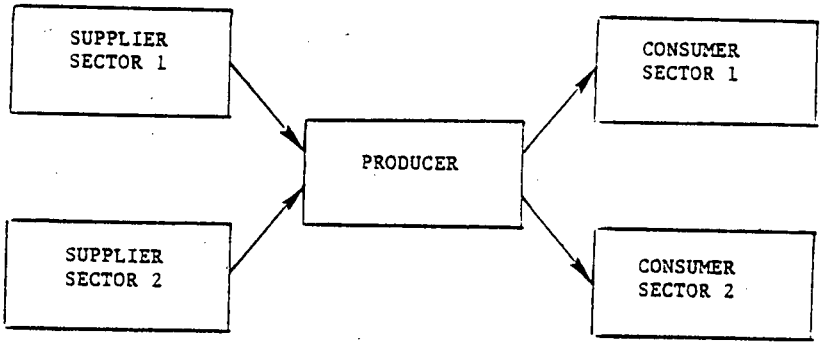



Figure 7.2. A Simple Network.

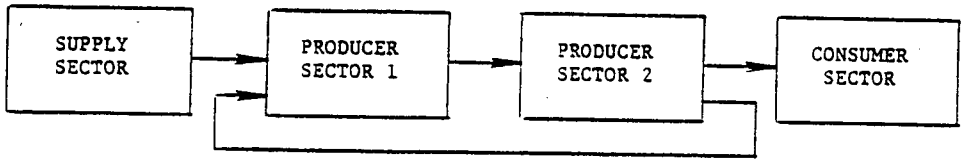



Figure 7.3. A Looped Network.

As a simple example, consider the simple energy network in Figure 7.4. Clearly the longest path connects three sectors, so we assign index values of 0, 1, and 2 to the fossil fuels, electricity, and consumer sectors, respectively. There are three markets in this network: a fossil fuels-electricity market, an electricity-consumer market, and a fossil fuels-consumer market. The third market includes sectors that are not in adjoining sector sets. Therefore we create a "dummy fuel" sector with index 1 and split the fossil fuels-consumer market into fossil fuels-dummy fuel and dummy fuel-consumer markets. The network of markets now has a chain-structure.

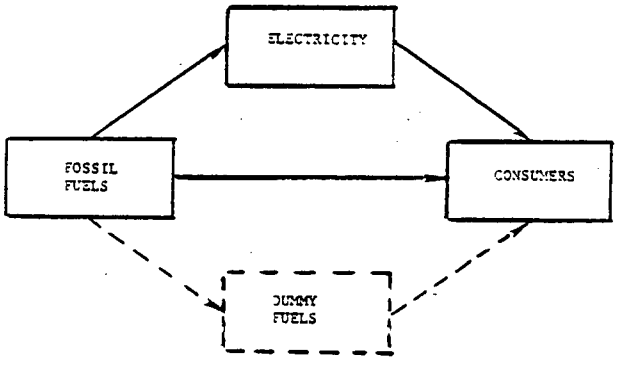



Figure 7.4. A Simple Energy Network.

The construction of this chain-structure itself is not immediately meaningful. However, as noted in descriptor variable systems, such structures are useful decomposition tools. The concepts developed for descriptor variable systems provide some fundamental insights into these equilibrium models. For example, the condition systems corresponding to this network often have an intuitive significance, since they can correspond to supply and demand functions, as will be demonstrated in Section 7.4.

## 7.3 THE DECENTRALIZED CONTROL MODEL FOR CHAIN-STRUCTURE ECONOMIC NETWORKS

In neoclassical microeconomic theory, all participants in the exchanges are assumed to be price-takers, i.e. they decide the quantities to be bought and sold based on given prices. The prices are "given" by a fictitious "auctioneer," whose task is to announce a set of prices for which all excess demands for quantities are zero. However alternative models can be formulated where a sector of participants may decide prices as well as commodities. One such modeling approach is discussed here, called the decentralized control approach.

The decentralized control approach is described by the following behavior for the three sector types: Suppliers are given the good flow rate demanded of them; they respond by announcing the price at which such flows will be provided. Producers are given the flow rate of output they are to produce and the unit prices of their inputs; they choose the composition of inputs they will use to produce the outputs and the price they will charge for each unit of output. Consumer sectors behave in the same manner as in the neoclassical approach - they are given prices and they choose their consumption of goods.

Under the decentralized approach, each price or quantity flow is determined by exactly one

sector model. The problem of determining equilibrium in these networks is to determine a set of quantities and prices such that each sector model will "choose" responses that are identical to the equilibrium signals. This notion of equilibrium corresponds to the game theoretic concept of Nash equilibrium, where no player (sector) has an incentive to change his response.

The sector models for suppliers and producers clearly assume that the sectors are not individual free enterprise firms, since an individual firm would clearly choose an infinite price for its outputs under this formulation. Therefore, these sector models embed some internal competition that results in the response of a finite price. However, these sectors may reflect oligopoly-type situations, such that while there is limited competition, the price can still exceed marginal cost.

This chapter will consider simple economic networks using the decentralized approach. The sector models will be assumed to be linear. The primary motivation for this assumption is that such networks are very tractable from the standpoint of descriptor analysis. However, the use of linear models requires some additional justification.

Other than perhaps being simplistic as a representation of reality, linear models admit infeasible possibilities such as negative prices or negative goods. While this may be conceptually disturbing, this is of practical consequence only if the equilibrium is infeasible, and this should happen only with grossly unrealistic sector models. When one expects a feasible solution, the addition of proper feasibility constraints is likely to produce only inoperative constraints at the expense of additional complexity.

Linear models are often justified by the argument that they are a good approximation in the region close to an operating point. This is valid in assessing the properties of an equilibrium point, such as stability. However, the purpose here is to study the underlying structure of the networks, and not merely the equilibrium point itself. Therefore, the use of linear models is not intended to only approximate local behavior, but also to reflect a topological structure of the sector models in the region of interest. The linearity of sector models assumes the following properties of the economic sector models:

- The models are continuous and well-defined in the region of interest. The models express responses as functions of the signals.
- Partial derivatives of the models are finite and remain either nonpositive or nonnegative in the region of interest, i.e. monotonicity.

We now proceed to study networks of these linear sector models using descriptor analysis. Let $q_i$ be a vector representing the flow of outputs from supply or producer sectors in sector set i, $0 \le i \le N$, where there are N+2 sector sets in the network chain. Let $p_i$ be the vector of corresponding prices per unit. The model for all supply sectors in sector set 0 is given by

$$p_0 = f_0 + S_0 q_0 \tag{7.1}$$

The model for all producer sectors in sector set j, $1 \le j \le N$, is given by

$$q_{j-1} = g_{j-1} + A_{j-1} p_{j-1} + B_j q_j \tag{7.2a}$$

$$p_j = f_j + C_{j-1} p_{j-1} + D_j q_j \ . \tag{7.2b}$$

Finally, the model for all consumer sectors, sector set N+1, is given by

$$q_N = g_N + R_N p_N \tag{7.3}$$

Note that all flows and prices are expressed as functions of other flows and prices. The system can be re-expressed:

$$\begin{bmatrix} I & -S_0 & & & & & \\ -A_0 & I & -B_1 & & & & \\ -C_0 & & I & -D_1 & & & \\ & & & \ddots & & & \\ & & & & -A_{N-1} & I & -B_N \\ & & & & -C_{N-1} & I & -D_N \\ & & & & & -R_N & I \end{bmatrix} \begin{bmatrix} p_0 \\ q_0 \\ p_1 \\ \vdots \\ q_{N-1} \\ p_N \\ q_N \end{bmatrix} = \begin{bmatrix} f_0 \\ g_0 \\ f_1 \\ \vdots \\ g_{N-1} \\ f_N \\ g_N \end{bmatrix} \tag{7.4}$$

This system has a unique solution if and only if the matrix in (7.4) is nonsingular. We would like to establish properties for the submatrices that insure the existence of a condition system.

Clearly (7.1) serves a set of initial conditions for the N+1 set chain, while (7.3) serves as end-conditions. Equations (7.1) correspond to an inverse supply function for the goods provided by the supply sectors. Intuitively, one expects the forward condition at any location to also represent an inverse supply function. Specifically, we hypothesize that the forward condition that summarizes the models for sector sets 0 through k is given by a matrix equation of the form

$$p_k = h_k + S_k q_k \tag{7.5}$$

Likewise, since the consumer sectors conditions correspond to a demand function, we hypothesize that the backward condition summarizing the models of sector sets k+1 through N+1 is given by a vector demand function:

$$q_k = \ell_k + R_k p_k \tag{7.6}$$

Since (7.1) corresponds to an inverse supply function, it is reasonable to assume some property of the matrix $S_0$. Occasionally in the economic literature, one finds that this matrix is assigned the property of positive definiteness. The economic justification for this assumption has been somewhat unclear. This does reflect that increasing the output of a supply good will increase the price of that good. Similarly, the matrix $R_N$ in the consumer demand function is assumed to be negative definite. This reflects that increasing the price of a consumer good will decrease the amount of that good consumed. (In this analysis the assumption of positive definite or negative definite will not imply symmetry of the matrix.) In addition, if we simply had a single market of suppliers and consumers, a unique equilibrium (solvability) would be assured:

*Theorem 7.1:* Given a market specified by inverse supply function

$$p = f + Sq \tag{7.7}$$

and demand function

$$q = g + Rp \tag{7.8}$$

where q and p are vectors and where S is positive definite and R is negative definite, a unique equilibrium exists.

*Proof:* Substituting (7.7) into (7.8) yields

$$q = g + Rf + RSq$$

Existence of a unique solution depends on the existence of $(I-RS)^{-1}$. Suppose it does not. Then I-RS is singular. By negative definiteness of R, $R^{-1}$ exists and is negative definite. Clearly

$$R^{-1}(I-RS) = R^{-1} - S$$

is also singular. However, since S is positive definite, $R^{-1} - S$ must be negative definite and nonsingular. This contradicts (I-RS) being singular. Hence $(I-RS)^{-1}$ exists and the equilibrium is given by

$$q = (I-RS)^{-1}[Rf + g]$$

$$p = f + S(I-RS)^{-1}[Rf + g] \qquad \blacksquare$$

Therefore, based on the desirable first derivative and solvability properties, we will assume (A1) $S_0$ is positive definite and (A2) $R_N$ is negative definite. We would like to assume properties on the producer sectors (7.2) such that $S_k$ in the forward condition (7.5) is positive definite and $R_k$ in the backward condition (7.6) is negative definite for all k.

Consider the matrix $B_j$ in (7.2a). This can be interpreted as the "technology matrix" of index set j since the $k^{th}$ element in the $i^{th}$ row of matrix indicates how much of the $i^{th}$ element in $q_{j-1}$ is required for each output unit of the $k^{th}$ element $q_j$. With this interpretation, it follows naturally that (A3) $B_j$ is assumed to be nonnegative.

If the sectors in sector set j are all fairly competitive, one would expect the price of an output to be determined by the marginal cost of the inputs. In this case, there would be duality between the input composition for a unit of output and the marginal cost of the output. For example, if the production of good z requires 2 units of good x and 3 units of good y, the marginal cost of z would be twice the marginal cost of x plus three times the marginal cost of y. In terms of (7.2b), the matrix $C_{j-1}$ would represent the transpose of the technology matrix. So we will assume (A4) $C_{j-1} = B_j'$ for all j, $1 \le j \le N$.

Now consider $A_{j-1}$ in (7.2a). This matrix can be interpreted as exogenous effects, like other consumers for the output that are outside the network. Therefore, to the extent that $A_{j-1}$ is of consequence, we may expect that it has behavior similar to a demand function. Therefore, we will assume (A5) $A_{j-1}$ is negative semi-definite. The remaining matrix in (7.2b), $D_j$, may reflect the effects of other inputs outside the network. In this sense, $D_j$ is similar to $S_0$ in (7.1), however, since there may be no external effects, we will assume (A6) a weaker condition of positive semi-definiteness for $D_j$. (As in the case of $S_0$ and $R_N$, these matrices may be asymmetric).

## 7.4 EQUILIBRIUM IN THE DECENTRALIZED CONTROL MODEL

This section presents the major result for networks modeled via the decentralized control approach. The following theorem simultaneously demonstrates the existence of a unique equilibrium and establishes the hypothesized condition systems of (inverse) supply functions and demand functions. The significance of this result is that it is consistent with the intuitive notions for these networks discussed in Section 7.1.

*Theorem 7.2:* Given an economic network described by (7.1), (7.2), and (7.3), with the accompanying assumptions (A1) - (A6), sufficient conditions for the completeness of the network model are that

$$B_{k+1}' B_{k+1} + D_{k+1} \tag{7.9}$$

is positive definite for all k and

$$-B_{k+1} B_{k+1}' + A_k \tag{7.10}$$

is negative definite for all k. Furthermore, there is a condition system described by (7.5) and (7.6).

*Proof:* We will demonstrate the existence of the condition system, and then using the forward and backward conditions at any market, observe that each market can be solved by Theorem 7.1. Therefore, the entire network of markets has a unique equilibrium.

The demonstration of the existence of the forward condition system can be established by induction. Suppose we have established the forward condition (7.5) for k with positive definite $S_k$ (this holds by assumption for k=0). Now consider (7.2) for j=k+1. This gives a system of

equations

$$p_k = h_k + S_k q_k \tag{7.11}$$

$$q_k = g_k + A_k p_k + B_{k+1} q_{k+1} \tag{7.12}$$

$$p_{k+1} = f_{k+1} + B'_{k+1} p_k + D_{k+1} q_{k+1}$$

Substituting (7.12) into (7.11) gives

$$(I - S_k A_k)\, p_k = h_k + S_k g_k + S_k B_{k+1} q_{k+1}$$

Now $(I-S_kA_k)$ is nonsingular by the same argument used in Theorem 7.1, since $S_k^{-1} - A_k$ is positive by assumption of negative semi-definite $A_k$. So we get

$$p_k = (I - S_k A_k)^{-1} [h_k + S_k g_k + S_k B_{k+1} q_{k+1}]$$

$$p_{k+1} = f_{k+1} + B'_{k+1} (I - S_k A_k)^{-1} [h_k + S_k g_k]$$

$$+ [B'_{k+1} (I - S_k A_k)^{-1} S_k B_{k+1} + D_{k+1}]\, q_{k+1} \tag{7.13}$$

Equation (7.13) gives the desired forward condition at k+1 where

$$h_{k+1} = f_{k+1} + B'_{k+1} (I - S_k A_k)^{-1} [h_k + S_k g_k]$$

$$S_{k+1} = B'_{k+1} (I - S_k A_k)^{-1} S_k B_{k+1} + D_{k+1}$$

Now the matrix $S_{k+1}$ is the sum of two positive semi-definite matrices, and therefore is positive definite unless there exists some vector $d \neq 0$ such that both

$$d' B_{k+1}' (I - S_k A_k)^{-1} S_k B_{k+1} d = 0 \tag{7.14}$$

and

$$d' D_{k+1} d = 0 \tag{7.15}$$

Now $(I-S_kA_k)^{-1}S_k$ is positive definite since its inverse

$$S_k^{-1} - A_k = S_k^{-1} (I - S_k A_k)$$

is positive definite. Hence (7.14) holds if and only if

$$d' B_{k+1}' B_{k+1} d = 0 \tag{7.16}$$

By theorem condition (7.9), both (7.15) and (7.16) cannot hold for any d. Therefore $S_k$ is positive definite for all k and we have a forward condition system of inverse supply functions with positive definite $S_k$.

The establishment of the backward condition system of demand functions for each market follows the same inductive argument, except the induction goes in the opposite direction in the network. The demand function for $q_N$ exists by assumption. Given the demand function for $q_k$, the demand function for $q_{k-1}$ summarizing (7.3) and (7.2) for $j \geq k$ is equivalent to the backward condition that summarizes

$$q_k = \ell_k + R_k p_k \tag{7.17}$$

$$p_k = f_k + B'_k p_{k-1} + D_k q_k \tag{7.18}$$

$$q_{k-1} = g_{k-1} + A_{k-1} p_{k-1} + B_k q_k$$

Substituting (7.18) into (7.17) gives

$$(I - R_k D_k)\, q_k = \ell_k + R_k f_k + R_k B'_k p_{k-1}$$

Now $(I-R_kD_k)^{-1}$ exists by previous argument, so

$$q_k = (I - R_k D_k)^{-1} [\ell_k + R_k f_k + R_k B'_k p_{k-1}]$$

$$q_{k-1} = g_{k-1} + B_k (I - R_k D_k)^{-1} [\ell_k + R_k f_k]$$

$$+ [B_k (I - R_k D_k)^{-1} R_k B'_k + A_{k-1}]\, p_{k-1} \tag{7.19}$$

Letting

$$\ell_{k-1} = g_{k-1} + B_k (I - R_k D_k)^{-1} [\ell_k + R_k f_k]$$

$$R_{k-1} = B_k (I - R_k D_k)^{-1} R_k B'_k + A_{k-1}$$

Equation (7.19) gives the desired backward condition in the form of a demand function for $q_{k-1}$. Matrix $R_{k-1}$ is negative definite by theorem condition (7.10), using the same type of argument as in demonstrating the positive definiteness of $S_k$.

Therefore, at each market j with good flows $q_j$ and prices $p_j$, we have an inverse supply function and demand function that determine a unique equilibrium. ■

A physical interpretation can be given to the conditions in Theorem 7.2. Suppose the technology matrix $B_{k+1}$ does not have full column rank. In this case there is no unique vector of outputs that the corresponds to a vector of inputs to index sector set k+1. Furthermore, there exists a vector of outputs $x_{k+1} + \Delta x_{k+1} \neq x_{k+1}$ which requires the same inputs technologically and for which $B_{k+1} \Delta x_{k+1} = 0$. Therefore,

$$(\Delta x_{k+1})' B_{k+1}' B_{k+1} \Delta x_{k+1} = 0$$

Hence (7.9) is positive definite only if

$$(\Delta x_{k+1})' D_{k+1} \Delta x_{k+1} > 0$$

In other words, in such cases where $B_{k+1}$ does not have full column rank, there need to be external effects, like consumption outside the network, in order that (7.9) be positive definite.

A similar conclusion can be made about condition (7.10). If $B_{k+1}$ does not have full row rank, the marginal cost of outputs determined on the basis of marginal cost of inputs in the model does not correspond to a unique vector of input prices. Therefore, there must exist externalities, such as unspecified inputs, reflected in the $A_{k-1}$ matrix to insure that (7.10) is negative definite.

The iterative process used to create the inverse supply function and demand function for each market and their significance as forward and backward conditions indicate that the initial inverse supply function "sweeps" forward through the network sector sets, while the consumer demand function "sweeps" backward through the network sector sets. This notion intuitively underlies much discussion of such networks and the design of large-scale equilibrium models. The preceding analysis justifies this intuition for the linear case.

Condition systems of supply and demand functions are likely to exist for other modeling approaches characterizing sector behavior. For example, the existence of the condition system can be established for simple networks where each sector is modeled as a price-taker, using assumptions similar to those in the decentralized control approach. The identification of these underlying condition systems follows directly from the fundamentals of descriptor variable systems. Since many economic network models can be expressed as chain-structure networks, descriptor variable systems appear to be a natural framework for network equilibrium analysis.

## 7.5 ANALYSIS OF ECONOMIC NETWORKS IN DESCRIPTOR FORM

The expression of a simple network model in descriptor form has benefits beyond the existence of supply and demand functions at each market in the chain-structure network. Often the purpose of creating these models is to aid policy analysis. Some of the results in this report, particularly in Chapter V, were developed to support such analyses.

For example, the objective of determining the best policy is often the problem of solving for the optimal controls. From Section 5.2, it is known that any system in descriptor form can be optimized using dynamic programming. If the policy variables are each directed at specific markets, (e.g., taxes or subsidies), dynamic programming will apply, despite the fact that system is nonregular and has varying numbers of quantities and prices associated with each market. One performance criterion that is often cited for evaluating policy is the social surplus. The social surplus consists of the sum of all profits to supplier and producer sectors plus the consumers' surplus, which is roughly the difference between the amount consumers are willing to pay and the amount they actually pay. At each iteration of a dynamic programming algorithm, the objective would be to choose policy inputs to markets k,...,N that maximize the surplus realized by sector sets k+1,...,N+1, given the output quantity and output price of sector set k. Each iteration is, of course, eased by the previous iteration. Thus, the policy for attaining the optimal social surplus can be determined by optimizing the surplus of successively longer subchains of the network.

In some cases the purpose of creating a network model is simply to ascertain the effects of a particular policy instrument. For example, the concern may be the effect of increasing the tax on a commodity in market i on the quantity of a commodity exchanged in market j. For a linear model, these effects can be determined using the sensitivity analysis procedure described in Section 2.5.

Sometimes a policy instrument behaves like a constraint on the system rather than as an input. Examples of this are price ceilings or output quotas. If constraints are imposed on individual markets, these constraints can be "propagated" forward and backward through a chain-structure network in a way similar to forward and backward condition functions. The result of such a constraint propagation is that the new constraints will reflect both the direct and indirect impacts created by the set of imposed levels. (For further discussion of policy analysis for networks in descriptor form, see [45]).

As described in Chapter IV, the propagation of forward and backward condition functions applies in the nonlinear case as well as in the linear case. However, actually performing numerical propagations of the entire nonlinear supply or demand function may be too complex. In such cases, iterative methods can be an effective means of determining the equilibrium. Most iterative techniques involve using the model to update "guesses" on some or all of the unknown quantities, iteratively repeating the updating procedure until the updated guesses are the same as the former guesses, indicating convergence. Although usually applied to nonlinear systems, iterative techniques are best examined on linear systems.

Two important results have been demonstrated involving iterative methods for chain-structure economic networks that are modeled via the decentralized control approach (Section 7.3). The first result concerns the comparison of the methods of simultaneous displacements and alternating displacements. In simultaneous displacements, each sector set simultaneously updates its responses based on the responses of the previous iteration. In alternating displacements, the sectors that update responses alternate between the odd-numbered sector sets and the even-numbered sector sets. One method will converge to an equilibrium if and only if the other method converges, however alternating displacements is twice as efficient as simultaneous displacements when there is convergence.

A second result concerns the method of successive displacements, where in each cycle price vectors are successively updated for market 0,

market 1, etc., to market N, and then quantity vectors are successively updated for market N, market N-1, etc., to market 0. The issue of interest is whether the convergence of this technique for the network depends on the convergence of this technique when applied to one of the chain of markets, i.e. holding fixed the prices and quantities of the other markets. With additional assumptions on the $A_{j-1}$ and $D_j$ matrices of the decentralized control model, the general convergence of successive displacements for any individual market is necessary for general convergence in a chain of markets. The proofs of these results appear in [45].

Looped networks of economic systems can be expressed in descriptor form, however the condition systems will not correspond to supply functions and demand functions. Looped networks can be expressed as the interconnection of two simple networks oriented in opposite directions. Thus, if the prices and quantities corresponding to flows in one of the networks are held fixed, the remaining flows in the other network will have *partial* condition systems of conditional supply and demand functions. This observation suggests an effective procedure for determining the equilibrium of a network where the reverse-oriented flows are sparse with respect to the forward-oriented flows [45].

# VIII. SYNTHESIS AND FUTURE RESEARCH

## 8.1 SYNTHESIS OF PROJECT THEMES

The discussion in Chapter VII describes how an economic network of markets can be converted to an equivalent network suitable for a model in descriptor form. This equivalent restructuring can be applied to many other types of large-scale networks. More importantly, however, the equivalence of many network models with a model in descriptor form provides a unifying link between the two themes of this project: descriptor variable theory and large-scale optimization.

In a wide sense, many models of large systems or dynamic systems can be expressed as a chain of interconnected subsystem models. For models of economic networks, the subsystems correspond to the sector sets. In the context of discrete dynamic system models, the subsystem model is the relationships governing the change of the system over a particular time interval. Suppose the variables describing the interaction between two adjacent subsystems are defined as a descriptor vector. With this interpretation, a chain of interconnected subsystem models is a model in descriptor form.

Consider the procedure of spatial dynamic programming, which applies to systems where a chain-structure is not immediate. In applying dynamic programming to a sequence of subsystems, interaction variables can be identified to make the optimization more efficient. The interaction variables at each iteration may be interpreted as forming the descriptor vectors characterizing the behavior of the system at particular links. When expressed in this manner, spatial dynamic programming is equivalent to dynamic programming for descriptor variable systems.

Thus, both descriptor variable theory and spatial dynamic programming operate on the same principle: exploitation of the chain-structure of the model. A system in descriptor form assumes this chain-structure, and therefore descriptor variable theory is the appropriate theoretical framework for any model in this form. Spatial dynamic programming relies on an efficient sequencing of subsystems and identification of corresponding interaction parameters, which evokes a chain-structure to accommodate the dynamic programming procedures. Therefore, together descriptor variable theory and spatial dynamic programming form a general approach to the wide class of large-scale systems that can be efficiently expressed as a chain of subsystems.

## 8.2 FUTURE RESEARCH

Descriptor variable theory provides a convenient framework for understanding many of the standard notions for analyzing dynamic systems, as illustrated in Section 2.6. The applications in Chapter II suggest that a descriptor representation often constitutes a more appropriate formulation for the study of dynamic systems, even when the system is originally in state-space form. Further research in this direction is likely to develop a comprehensive theory of dynamic systems and identify new concepts for the study of these systems.

The application of descriptor variable theory and spatial dynamic programming to different classes of large-scale systems will not only test the usefulness of this approach, but will encourage other theoretical developments and extension of the approach. The applications to economic networks and power systems have been very promising and are worthy of further development. Additional work on dynamic economic systems and defense systems has also indicated significant benefits from this approach. Other application areas, such as communication systems and transportation networks, are being explored.

The exploitation of natural structure in the analysis of a system model provided the major focus for the research in this project. The results underscore the importance of recognizing and understanding structure. The approach that was developed here used the paradigm of a chain of interconnected subsystems. Of course, this approach will best suit systems that have or nearly have this structure. Other paradigms can and should be explored. For example, systems could be viewed from the standpoint of a grid of subsystems. By developing approaches for other important structure paradigms, our ability to analyze large-scale systems will be enhanced.

## IX. REFERENCES


[1] Gardner, M.F. and J. L. Barnes, *Transients in Linear Systems*, Vol. 1, New York, Wiley, 1942.

[2] Kalman, R. E., "On the General Theory of Control Systems," *Proc. 1st IFAC Congr.*, 1960.

[3] Zadeh, L. A. and C. A. Desoer, *Linear System Theory: The State Space Approach*, New York: McGraw-Hill, 1963.

[4] Crandall, S. H., D. C. Karnopp, E. F. Kurtz, Jr., and D. C. Pridmore-Brown, *Dynamics of Mechanical and Electromechanical Systems*, New York: McGraw-Hill, 1968.

[5] Bailey, F. N., "The Application of Lyapunov's Second Method to Interconnected Systems," *SIAM J. Contr.*, Ser. A, Vol. 3, pp. 443-462, 1965.

[6] Sing, S. P., and R.-W. Liu, "Existence of State Equation Representation of Linear Large-Scale Dynamical Systems," *IEEE Trans. Circuit Theory*, Vol. CT-20, pp. 239-246, May 1973.

[7] Enns, M., J. E. Matheson, J. R. Greenwood, and F. T. Thompson, "Practical Aspects of State Space Methods," *Proc. 1964 Joint Automatic Control Conf.*, pp. 494-513.

[8] Anderson, J. H., "Matrix Methods for the Study of a Regulated Synchronous Machine," *Proc. IEEE*, Vol. 57, pp. 2122-2136, Dec. 1969.

[9] Kokotovic, P. V. and R. A. Yackel, "Singular Perturbation of Linear Regulators: Basic Theorems," *IEEE Trans. Automatic Control*, Vol. AC-17, pp. 29-37, Feb. 1972.

[10] Rosenbrock, H.H., *State-Space and Multivariable Theory*. New York: Wiley, 1970.

[11] Luenberger, D. G. and A. Arbel, "Singular Dynamic Leontief Systems," *Econometrica*, 1977.

[12] Silverman, L. M., "Discrete Riccati Equations: Alternative Algorithms, Asymptotic Properties, and System Theory Interpretations," in *Control and Dynamic Systems*, edited by C. T. Leondis, Vol. 12, Academic Press, New York, 1976.

[13] Grossman, S., "Rational Expectations and the Economic Modeling of Markets Subject to Uncertainty," *J. Econometrics*, No. 3, pp. 255-272, 1975.

[14] Rosenbrock, H. H., *State Space and Multivariable Theory*. John Wiley & Sons, New York, 1970.

[15] Campbell, S. L., C. D. Meyer, Jr., and N. J. Rose, "Applications of the Drazin Inverse to Linear Systems of Differential Equations with Singular Constant Coefficients," *SIAM J. Appl. Math.* 31,(3) pp. 411-425, 1976.

[16] Godbout, L. F. and D. Jordan, "On State Equation Descriptions of Linear Differential Systems, *ASME Trans. J. Dynamic Systems Meas. Control 97*, (Series G) pp. 333-344, 1975.

[17] Gantmacher, F. R., *The Theory of Matrices*, Vol. II. Chelsea, New York, 1959.

[18] Turnbull, H. W. and A. C. Aitken, *An Introduction to the Theory of Canonical Matrices*. Dover, New York, (First published by Blackie & Son, 1932), 1961.

[19] Guillemin, V. and A. Pollack, *Differential Topology*, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1974.

[20] Fikhtengol'ts G. M., *The Fundamentals of Mathematical Analysis*, Vol. 2, Pergamon Press, Oxford, 1965.

[21] Larson, R. E., *State Increment Dynamic Programming*, Elsevier, 1968.

[22] Silverman, L. M., and H. J. Payne, "Input-Output Structure of Linear Systems with Application to the Decoupling Problem," *SIAM Journal on Control*, Vol. 9, No. 2, pp. 199-233, May 1971.

[23] Bryson, A. E., and Y. C. Ho, *Applied Optimal Control*, Ginn, 1969, Chapter 3.

[24] Kahne, S. J., "On Mobility in Constrained Dynamical Systems," *IEEE Transactions on Automatic Control*, Vol. AC-9, No. 3, July 1964.

[25] Schlueter, R. A., and A. H. Levis, "The Optimal Adaptive Sampled-Data Regulator," *Transactions of the ASME*, pp. 334-340, September 1974.

[26] Cline, T. B., and R. E. Larson, "Decision and Control in Large-Scale Systems via Spatial Dynamic Programming," *Proceedings of Lawrence Symposium on Systems and Decision Sciences*, Western Periodicals Company, North Hollywood, California, 1977.

[27] Bertelè, U. and F. Brioschi, *Nonserial Dynamic Programming*, Academic Press, New York and London, 1972.

[28] Aris, R., *Discrete Dynamic Programming*, Braisdell Publishing Company, New York, 1964.

[29] Mine, H., and K. Ohno, "Decomposition of Mathematical Programming Problems by Dynamic Programming and its Applications to Block-Diagonal Geometric Programs," *J. Math. Anal. and Appl.*, Vol. 32, pp. 370-385, 1970.

[30] Mine, H., K. Ohno, and M. Fukishima, "Multi-level Decomposition of Nonlinear Programming Problems by Dynamic Programming," *J. Math. Anal. and Appl.*, Vol. 53, pp. 7-27, 1976.

[31] Larson, R. E., and A. J. Korsak, "A Dynamic Programming Successive Approximations Technique with Convergence Proofs - Parts I and II," *Automatica*, Vol. 6, pp. 245-260, March 1970.

[32] Elgerd, O., *Electric Energy Systems Theory: An Introduction*, McGraw-Hill, New York, 1971.

[33] Debreu, G., *Theory of Value*, New Haven: Yale University Press, 1959.

[34] Quirk, J. and R. Saposnik, *Introduction to General Equilibrium Theory and Welfare Economics*, McGraw-Hill, 1968.

[35] Arrow, K.J. and F. H. Hahn, *General Competitive Analysis*, San Francisco: Holden-Day, 1971.

[36] Brock, H. W., and D. M. Nesbitt, *Large-Scale Energy Planning Models: A Methodological Analysis*, NSF Contract No. C-915-1977.

[37] Hoffman, K. C., and D. W. Jorgenson, "Economic and Technological Models for Evaluation of Energy Policy," *The Bell Journal of Economics*, Vol. 8, No. 2, Autumn, pp. 444-466, 1977.

[38] Hogan, W. W., "Energy Policy Models for Project Independence," *Computation and Operations Research*, Vol. 2, pp. 251-271, 1975.

[39] Hogan, W. W., "Project Independence Evaluation System: Structure and Algorithm," Presented at American Mathematical Society Short Course on Mathematical Aspects of Production and Distribution of Energy, San Antonio, Texas, January 20-21, 1976.

[40] Levis, A. H., E. R. Ducot, K. M. Sidenblad, and I. S. Levis, *AGRIMOD: A Dynamic Simulation Model of the U.S. Food Production System, Part I: Crop Production*, Report No. PRA-75-22720/1/8, Systems Control, Inc., Palo Alto, Calif., March 1977.

[41] Manne, A. S., "ETA: A Model for Energy Technology Assessment," *The Bell Journal of Economics*, Vol. 7, No. 2, pp. 379-406.

[42] Naill, R. F., *Managing the Energy Transition*, Volumes I and II, Cambridge, Mass: Ballinger, 1977.

[43] Takayama, T., and G. G. Judge, *Spatial and Temporal Price and Allocation Models*. Amsterdam: North-Holland, 1971.

[44] Cazalet, E. G., *Generalized Equilibrium Modeling: The Methodology of the SRI-Gulf Energy Model*, Draft Final Report to Federal Energy Administration, 1977.

[45] Stengel, Donald N., *Descriptor Analysis of Economic Systems*, Ph.D. dissertation, Dept. of Engineering-Economic Systems, Stanford University, August 1978.

## X. PUBLICATIONS OF PROJECT RESEARCH

In the course of the research for this project, results have been documented in journal articles and conference papers. Although this report is a complete presentation of all major results of this research, in some cases the material has been presented with a different perspective than in the earlier publications. For this reason, these publications are listed below:


1. Luenberger, D. G., "Dynamic Equations in Descriptor Form," *IEEE Transactions on Automatic Control*, Vol. AC-22, pp. 312-322, June 1977.

2. Luenberger, D. G., "Time-Invariant Descriptor Systems," *Proceedings of 1977 Joint Automatic Control Conference*, June 1977.

3. Cline, T. B., and R. E. Larson, "Decision and Control in Large-Scale Systems Via Spatial Dynamic Programming," *Proceedings of Lawrence Symposium on Systems and Decision Sciences*, Berkeley, California, October 1977.

4. Stengel, D. N., "A Descriptor Approach to Economic Network Equilibrium," *Proceedings of Second International Symposium on Large Engineering Systems*, Waterloo, Ontario, Canada, May 15-16, 1978.

5. Stengel, D. N., *Descriptor Analysis of Economic Systems*, Ph.D. Dissertation, Stanford University, August 1978.

6. Luenberger, D. G., "Time-Invariant Descriptor Systems," *Automatica*, Vol. 14, pp. 473-480, September 1978.

7. Chong, C. Y., P. L. McEntire, and R. E. Larson, "Decomposition of Mathematical Programming by Dynamic Programming," *Proceedings of Second Lawrence Symposium on Systems and Decision Sciences*, Berkeley, California, October 1978.

8. McEntire, P. L., C. Y. Chong, and R. E. Larson, "A Global Optimality Theorem for Spatial Dynamic Programming," *Proceedings of Second Lawrence Symposium on Systems and Decision Sciences*, Berkeley, California, October 1978.

9. Stengel, D. N., "Optimal Control of Completely Maintainable Dynamic Systems," *Proceedings of 17th Conference on Decision and Control*, San Diego, CA., January 10-12, 1979.




*U.S. GOVERNMENT PRINTING OFFICE : 1979 0-620-097/2190