Decentralized Pole Placement Using Polynomial Matrix Fractions

by

Helal M. Al-Hamadi

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

APPROVED:

H. F. VanLandingham, Chairman

H. Sherali

A. G. Phadke

W. Baumann

J. Ball

May, 1988

Blacksburg, Virginia

Decentralized Pole Placement Using Polynomial Matrix Fractions

by

Helal M. Al-Hamadi

H. F. VanLandingham, Chairman

Electrical Engineering

(ABSTRACT)

As the dimension and the complexity of large interconnected systems grow, so does the necessity for decentralized control. One of the interesting challenges in the field of decentralized control is the arbitrary pole placement using output feedback. The feasibility of this problem depends solely on the identification of the decentralized fixed modes. As a matter of fact, if the system is free of fixed modes, then by increasing the controller's order, any arbitrary closed loop poles can always be assigned. Due to this fact, reducing the controller's order constitutes another interesting challenge when dealing with decentralization.

This research describes the decentralized pole placement of linear systems. It is assumed that the internal structure of the system is unknown. The only access to the system is from a number of control stations. The decentralized controller consists of output feedback controllers each built at a control station.

The research can be divided into two parts. In the first part, conditions for fixed modes existence as well as realization and stability of the overall system under decentralization are established using polynomial matrix algebra. The second part deals with the solution of decentralized pole placement problem, in particular, finding a decentralized controller which assigns some set of desired

poles. The solution strategy is to reduce the controller's order as much as possible using mathematical programming techniques. The idea behind this method is to start with a low order controller and then attempt to shift the poles of the closed loop system to the desired poles.

Table of Contents

List of Illustrations

List of Tables

Chapter 1: Introduction

1.1 Overview

Although there is no unified definition of large-scale systems, the consensus is to refer to large-scale systems as those which possess high dimensions and large number of parts which are interconnected in a complex way.

Large-scale systems arise in many fields of science and technology. Socio-economic systems, power generation systems, network flow systems, chemical processes, dynamic file assignment in computer networks and transportation systems are just few examples of large scale systems.

Centralized control theory is based on the concept of controlling the entire system by a single centrally located controller which has access to all available information about the system. The theory of designing such controllers for a linear system is well established [1],[2],[6].

As the large-scale systems of the application areas mentioned earlier grow in dimension and complexity, the classical centralized control methodology of analysis and design does not seem to be adequate [45].

The centralized scheme has a number of economical and technical disadvantages when applied to large-scale systems. First is the complexity of the controller which may not be able to process the large amount of data required to control the overall system. As a result, expensive computing time and space are required to process such data, and also a large number of communication links are required. Obviously, as the communication links increase in number and in length, the cost of constructing and maintaining them will also increase.

In recent years, researchers have tried to develop satisfactory methods in the field of modeling, analysis and design of large scale systems. The fundamental idea of these methods is to decompose the centralized problem into subproblems using techniques such as aggregation [47], [55]; perturbation [56], [57]; hierarchical decomposition [15], [57]; and decentralized control [19], [21].

Decentralized control is the subject of this research. Many control systems such as power systems, socio-economic systems and computer-automated manufacture systems are decentralized in nature. However, the decentralized control approach can be imposed whether the inherent structure of the system is decentralized or not.

The fundamental idea behind decentralization is to divide the large-scale system into a number of control stations. Each control station possesses a limited amount of information about the overall system. The control problem is then to use a set of decentralized local controllers, each at a different control station observing only the output at its control station and producing only the control signals that affect its local station.

There are many advantages of using a decentralized control strategy over the use of one central unit [15], [30]. Some of these advantages are:

Cost Reduction: the number and the length of the transmission links between the plant and the controller, in most applications, are reduced. Moreover, the cost of constructing a single complex controller unit is eliminated.

Simplicity: it is easier to construct, operate and maintain several simple controllers than a single complex one.

Reliability: the reduction of communication links not only reduces the cost, but also minimizes the possiblity of error when transmitting the information.

The remainder of this chapter surveys the decentralized control literature. Chapter 2 covers some mathematical results in polynomial matrices. These results will be frequently used in the subsequent chapters. Chapter 3 constitutes the main result of this research. Conditions for the existence of fixed modes as well as realization and stability of the plant using a decentralized controller are presented. The solution of the arbitrary pole placement problem, with the objective of reducing the controller's order, is given. Chapter 4 is dedicated to a vital and indispensable application of modern society. The techniques of chapter

3 are applied to a multi-area power generation system. In chapter 5, concluding remarks and future work are discussed.

1.2 Survey of Decentralized Control Methods

One of the active fields in control theory is the field of decentralized control. The literature on decentralized control covers three basic areas [17]. The first area is on stochastic decentralized control. It started with the work of Witsenhausen [16], [48], on non-classical information patterns, followed by the work of Aoki [49], and Sandell and Athans [50] on the solution of the linear stochastic control problem with quadratic cost. The second area is deterministic decentralized control using pole placement techniques. The fundamental work in this area was done by Wang and Davison [21] and Corfmat and Morse [18],[19]. Fessas [30] developed the same results of Corfmat and Morse using the polynomial methods of Wolovich 1974 [5]. The third area is concerned with decentralized controller design. The results of the second area form the first step in the design of the decentralized controllers. Issues of existence of fixed modes and controllability are discussed in the works on deterministic decentralized control. The design problem of decentralized controllers is studied by several authors Davison [24], [25], Hassan and Singh [51], Viswanadham [52], and others. To the author's knowledge all the work done in the design problem uses state feedback techniques. The remainder of this section is a brief summary of the work done in the field of deterministic decentralized pole placement.

Consider the linear system

$$\dot{x}(t) = A\,x(t) + \sum_{i=1}^{N} B_i\,u_i(t)$$

$$y_i = C_i\,x(t) \qquad i = 1, 2, \ \ldots\ , N \tag{1.2.1}$$

where N is the number of control stations, $x \in R^n$ is the state, $u_i \in R^{q_i}$, $y_i \in R^{p_i}$ are the input and output vectors of the i^{th} subsystem, and A, B_i and C_i are constant matrices with the proper dimensions.

The i^{th} controller is given by the following dynamic compensation:

$$\begin{aligned} \dot{z}_i(t) &= S_i\,z_i + R_i\,y_i \\ u_i(t) &= Q_i(t)\,z_i(t) + K_i(t)\,y_i(t) + v_i \qquad i = 1, 2, \ \ldots\ , N \end{aligned} \tag{1.2.2}$$

where $z \in R^{n_i}$ is the state of the i^{th} controller, $v_i \in R^{q_i}$ is the i^{th} local external input, and S_i, R_i, Q_i are constant matrices.

let K be a set of block diagonal matrices defined as follows:

$$\mathrm{K} = \{K \mid K = blockdiag\,(K_1, \ \ldots\ , K_N), \qquad i = 1, \ \ldots\ , N\} \tag{1.2.3}$$

Then the set of fixed modes of the triple (C,A,B) with respect to K is defined as follows:

$$\Lambda\,(C, A, B;\ \mathrm{K}) = \bigcap_{K \in \mathrm{K}} \sigma(A + BKC) \tag{1.2.4}$$

where $\sigma(A + BKC)$ denotes the eigenvalues of $(A + BKC)$, $B = [B_1 \mid \ \ldots\ \mid B_N]$ and $C = [C_1^T \mid \ \ldots\ \mid C_N^T]^T$.

Davison and Wang [21] have described a straightforward procedure to compute the fixed modes of a decentralized system. Furthermore, they have proved a theorem which states that necessary and sufficient conditions for the existence of a set of local feedback laws (1.2.2), such that the closed loop system is asymptotically stable, is that all fixed modes are stable.

In the proof of the above theorem, a constructive algorithm is given. It starts with a feedback matrix K such that all the poles of $(A + BKC)$ are distinct from those of A. Then, the poles which are controllable and observable from a given station are placed by using a successive application of dynamic feedback [22].

Corfmat and Morse[19] have studied the effect of decentralized feedback on the closed loop properties of the k-channel, jointly controllable and jointly observable, linear system. They have introduced the notion of strongly connected systems. A system is strongly connected if every node is connected to every other node in its graph. They have shown that strongly connected systems can be made controllable and observable through a single channel by application of static feedback to all other channels.

Furthermore, Corfmat and Morse have introduced the concept of completeness of a system [18],[19]. A system is complete if its transfer matrix is non zero and its remnant polynomial denoted by $\rho(C, A, B)$ equals unity. The remnant polynomial is defined as the product of the invariant polynomials of the system matrix:

$$\begin{bmatrix} sI - A & B \\ C & 0 \end{bmatrix} \tag{1.2.5}$$

For a strongly connected system, Corfmat and Morse [18] have shown that system completeness is a necessary and sufficient condition for the system to be controllable and observable from *any* channel is that the system must be complete.

It is impossible to make a nonstrongly connected system controllable and observable from a single channel. Corfmat and Morse have overcome this problem by decomposing nonstrongly connected systems to smaller strongly connected subsystems. Each subsystem can then be made controllable and observable from a single channel.

Corfmat and Morse's approach has some drawbacks [17]. Most importantly, the method concentrates the job of placing the poles of the system on a single controller. This has the effect of increasing the complexity of a single controller with most applications. Moreover, their method suffers a serious design problem if some of the modes are very weakly controllable or observable from a single channel. In effect high gains are required to assign those modes.

The first step of Davison and Wang's constructive algorithm has the same effect of placing the load on a single controller, even though it has not been stated explicitly.

Using the matrix polynomial techniques developed by Wolovich [5], Fessas [26], [27] developed the algebraic counterparts of the geometric results of Corfmat and Morse. In terms of polynomial matrix fractions he produced the concept of completeness, strong connectness and D-controllability of an interconnected system.

The notion of fixed modes of a decentralized system corresponds to the idea of uncontrollable and unobservable modes of the triple (A,B,C) of (1.2.1).

For decentralized systems a well known theorem [1], [2], [6] states that by the use of dynamic feedback the controllable and observable poles of the system (1.2.1) can be arbitrary assigned. On the other hand, the uncontrollable and the unobservable modes of (1.2.1) are uneffected by any dynamic feedback compression.

When decentralization is considered, the notion of fixed modes [21] and D-controllability and D-observability arises. It has been shown [29] that the controllability and observability of the original plant does not necessarily imply the controllability and observability of the decentralized plant. Moreover, it has been shown [18], [19] that the absence of uncontrollable and unobservable modes does not guarantee the ability of arbitrary pole assignment using decentralized output feedback. In summary, the decentralized system is free of fixed modes if and only if it is D-controllable.

An algebraic characterization of fixed modes is studied by Vidyasagar and Viswanadham [32]. They developed a formula for the decentralized fixed polynomial whose zeros are the fixed modes in terms of the greatest common divisor of some minors of the plant transfer matrix.

1.2.1 Survey of Fixed Modes Characterization

In this subsection the issue of fixed modes will be considered in more details. A survey of the methods for testing the existence of fixed modes and methods of computing them will be discussed. This interest in fixed modes is due to their strong relation to the problems of pole placement and overall system stability under decentralized structure. These modes may be thought of as being a generalization of the non-controllable and observable modes of the centralized system which are not both controllable and observable in the centralized sense [21].

Wang and Davison 1973 :

We first start with the definition and the fundamental result of Wang and Davison 1973, [21].

Definition 1.2.1 [21]: Consider a linear system and a decentralized controller as given by (1.2.1) and (1.2.2). And let $\mathrm{K} = \{K \mid K = \text{block-diag}\,[K_1, \ \ldots\ , K_N],$ $K_i \in R^{q_i \times p_i},\ i = 1, \ \ldots\ ,N\}$, $B = [B_1, \ \ldots\ ,B_N]$ and $C = [C_1^T, \ \ldots\ ,C_N^T]^T$. Then the set of fixed modes of the system (1.2.1) with respect to K is defined by (1.2.4):

$$\Lambda(C,A,B,\mathrm{K}) = \bigcap_{K \in \mathrm{K}} \sigma\,(A + BKC)$$

where $\sigma(A + BKC)$ is the set of eigenvalues of $(A + BKC)$.

Thus, the fixed modes of the system are those poles of the open loop plant which are invariant under any decentralized feedback matrix K. In other words,

the fixed modes are dependent only on the control structure and cannot be shifted by applying any decentralized control whether it is static or dynamic.

The basic Theorem of Wang and Davison 1973 is:

Theorem 1.2.1 [21]:

(i) The closed loop system, as given by (1.2.1) and (1.2.2), is stabilizable by an appropriate choice of matrices S_i, R_i, Q_i, K_i, $i = 1, \ldots, N$, if and only if $\Lambda(C,A,B,\mathrm{K}) \subset C^-$, where C^- is the complex left plane and K is the set of block diagonal output feedback matrices.

(ii) All the poles of the closed loop system can be placed arbitrarily (subject to the conjugate-pair condition) by an appropriate choice of the matrices S_i, R_i, Q_i, K_i, $i = 1, \ldots, N$, if and only if $\Lambda(C,A,B,\mathrm{K}) = \phi$.

The following algorithm shows that the decentralized fixed modes of the system may be calculated in a very simple way.

Algorithm 1.2.1 [21]:

1. Compute the eigenvalues of A.
2. Select 'arbitrary' matrices K_i, $i = 1, \ldots, N$.
3. Compute the eigenvalues of $A + \sum_{i=1}^{N} B_i K_i C_i$.
4. Then the decentralized fixed modes are contained in those eigenvalues of $A + \sum_{i=1}^{N} B_i K_i C_i$ which are common with the eigenvalues of A. Moreover, for almost all K_i, $i = 1, \ldots, N$ chosen, the decentralized fixed modes are equal to the eigenvalues of $A + \sum_{i=1}^{N} B_i K_i C_i$ which are common with the eigenvalues of A.

5. If in doubt as to which the decentralized fixed modes are, choose new 'arbitrary' matrices K_i, $i = 1, \ldots, N$ in step 2 and repeat steps 3 and 4.

After this fundamental work of Wang and Davison a great deal of research work was devoted for characterization of decentralized fixed modes. Although the result of Wang and Davison is fundamental in providing an algorithm for computing the fixed modes, it lacks the ability to provide a test for the existance of the fixed modes [58].

Anderson and Clements 1981 :

An algebraic characterization of fixed modes was provided by Anderson and Clements 1981 :

Theorem 1.2.2 [58]: λ^0 is a fixed mode of system (1.2.1) for a decentralized control if and only if there exist disjoint sets of indices $\alpha = \{i_1, \ldots, i_k\}$, $\beta = \{i_{k+1}, \ldots, i_N\}$, $\alpha \cup \beta = \{i_1, \ldots, i_N\}$ such that

$$rank \begin{bmatrix} \lambda^0 I - A & B_\alpha \\ C_\beta & 0 \end{bmatrix} < n \qquad (1.2.6)$$

where $B_\alpha = [B_{i_1}, \ldots, B_{i_k}]$, and $C_\beta = [C_{i_{k+1}}^T, \ldots, C_{i_N}^T]^T$.

For two control stations the above result has a useful interpretation. If λ^0 is a fixed mode of (1.2.1), then λ^0 is simultaneously uncontrollable from one of the two stations and unobservable from the other station. We can conclude from the

above result that if an eigenvalue of A is controllable and observable from one station then it cannot be a fixed mode. Anderson and Clements [58] have also provided a result for characterization of fixed modes using transfer function matrix description of the system (1.2.1). Their result is:

Suppose the open-loop system transfer function matrix has a left matrix fraction description $A^{-1}(s)\,B(s)$, so that in Laplace transform notation $A(s)Y(s) = B(s)U(s)$, where $U(s) = [u_1^T, \ \ldots \ , u_N^T]^T$ and $Y(s) = [y_1^T, \ \ldots \ , y_N^T]^T$, the dimensions of u_i and y_i are q_i and p_i, $i = 1,2, \ \ldots , N$. Partition $A(s)$ and $B(s)$ as:

$$\begin{aligned} A(s) &= [A_1(s) \ \ldots \ A_N(s)] \\ B(s) &= [B_1(s) \ \ldots \ B_N(s)] \end{aligned} \tag{1.2.7}$$

The closed-loop system transfer function matrix can be written in the left matrix fraction description $\overline{A}^{-1}(s)\,B(s)$ where

$$\overline{A}^{-1}(s) = [A_1(s) + B_1(s)K_1 \ \ldots \ A_{m-1}(s) + B_{m-1}(s)K_{m-1} \ A_m(s)] \tag{1.2.8}$$

where K_i, $i = 1,2, \ \ldots \ ,N-1$ are the static feedback matrices, such that

$$u_i = -K_i y_i + v_i , \qquad i = 1,2, \ \ldots \ N-1 \tag{1.2.9}$$

The left matrix fraction $A^{-1}(s)\,B(s)$ with the feedback structure of (1.2.3) has a fixed mode at λ^0 if $\overline{A}(\lambda^0)$ is singular for all K_i , $i = 1, \ \ldots \ , N-1$.

Theorem 1.2.3 [58] The left matrix fraction $A^{-1}(s)\,B(s)$ with the feedback pattern of (1.2.9) defined by the positive integers $q_1, \ldots, q_{N-1}, p_1, \ldots, p_{N-1}$ and non-negative q_N, p_N has a fixed mode at λ^0 if and only if there exist a nonempty subset $\mathrm{I} = \{i_1, i_2, \ldots, i_j\}$ of $\{1,2, \ldots N\}$ for which

$$rank\ [A_{i_1}(\lambda^0)\ B_{i_1}(\lambda^0)\ \ldots\ A_{i_j}(\lambda^0)\ B_{i_j}(\lambda^0)\ ] < \sum_{i \in \mathrm{I}} p_i \qquad (1.2.10)$$

except that if $N \in \mathrm{I}$, $B_N(\lambda^0)$ is omitted in the matrix.

Condition (1.2.10) can be tested by finding zeros of det $A(s)$ and then checking the ranks of a number of complex matrices.

Anderson 1982 [59]:

Another useful test for the existence of decentralized fixed modes using transfer function matrix was provided by Anderson 1982 [59]. Anderson has also defined the concept of the degree of a fixed mode. The result of Anderson 1982 may be useful in cases when it is required to test for decentralized fixed modes for systems which are of fixed structure but with variable parameters. The following is the main result of the work of Anderson 1982.

Write $\mathrm{P}_1 = [A_1\ \ldots\ A_j]$, $\mathrm{P}_2 = [A_{j+1}\ \ldots\ A_N]$, $\mathrm{B}_1 = [B_1\ \ldots\ B_j]$ and $\mathrm{B}_2 = [B_{j+1}\ \ldots\ B_N]$, where j is the index used in (1.2.10).

Definition 1.2.2 [59]: With the above notation, suppose that

$$rank\,[P_1(\lambda^0) \quad B_1(\lambda^0)] < \beta = \text{number of columns in } P_1 \tag{1.2.11}$$

Then the ***degree*** of the associated fixed mode is the largest positive integer k such that all $\beta x \beta$ minors of $[P_1(s) \quad B_1(s)]$ has a zero at λ^0 of order at least k.

Theorem 1.2.4 [59]: Suppose that the transfer function matrix description is given by:

$$W(s) = \begin{bmatrix} W_{11}(s) & W_{12}(s) \\ W_{21}(s) & W_{22}(s) \end{bmatrix},$$

$$Y_i(s) = \sum_{j=1,2} W_{ij}(s)\, U_j(s)$$

where Y_i and U_i, $i = 1,2$ are the outputs and inputs of the system corresponding to the partition of (1.2.11). And consider control structures of the form $U_j = K_j Y_j + V_j$, $j = 1,2$. Suppose that $W_{11}(s)$ has β rows and $W_{21}(s)$ has $\bar{\beta}$ rows and let $a(s)$ be the characteristic polynomial of $W(s)$. The following two conditions are equivalent.

(i) With $[P_1(s) \quad P_2(s)]^{-1} [B_1(s) \quad B_2(s)]$ a left coprime matrix fraction description of $W(s)$, $rank\,[\,P_1(\lambda^0) \quad B_1(\lambda^0)\,] < \beta$ and the fixed mode λ^0 has degree k.

(ii) Suppose $a(\lambda^0)$ has a zero of order κ. Let

$$\#\begin{pmatrix} i_1 & \cdots & i_p \\ l_1 & \cdots & l_p \end{pmatrix} = \text{number of zeros at } \lambda^0 \text{ of } W\begin{pmatrix} i_1 & \cdots & i_p \\ l_1 & \cdots & l_p \end{pmatrix}$$

where

$$W\begin{pmatrix} i_1 & \cdots & i_p \\ l_1 & \cdots & l_p \end{pmatrix} = \text{minor formed from rows } i_1, \ \ldots \ , i_p$$

and columns $l_1, \ \ldots \ , l_p$ of W

Let $N = \beta + \bar{\beta}$ and

$$\delta_r\begin{pmatrix} i_1 & \cdots & i_p \\ l_1 & \cdots & l_p \end{pmatrix} = |\{i'_1, \ \ldots \ , i'_{N-p}\} \cap \{1, \ \ldots \ , \beta\}|$$

where

$$\{i_1, i_2, \ \ldots \ , i_p\} \cup \{i'_1, \ \ldots \ , i'_{N-p}\} = \{1,2, \ \ldots \ ,N\}$$

and

$$\delta_c\begin{pmatrix} i_1 & \cdots & i_p \\ l_1 & \cdots & l_p \end{pmatrix} = |\{l_1, \ \ldots \ , l_p\} \cap \{1, \ \ldots \ , \gamma\}|$$

where γ is the number of columns of $B_1(s)$. Then there exists k, $0 < k \leq \kappa$ such that whenever $\delta_r + \delta_c \geq \beta$,

$$\#\begin{pmatrix} i_1 & \cdots & i_p \\ l_1 & \cdots & l_p \end{pmatrix} \geq (k-\kappa) + (\delta_r + \delta_c - \beta) \qquad (1.2.12)$$

for all minors of $W(s)$.

The above theorem in its general form is complicated. A special case is when λ^0 is restricted to be a simple pole, which implies $\kappa = k = 1$.

Proposition 1.2.1 [59] Assume the same hypothesis of the above theorem, and suppose that λ^0 is a simple pole of $a(s)$. Then the following two conditions are equivalent.

(i) With $[P_1(s) \quad P_2(s)]^{-1}[B_1(s) \quad B_2(s)]$ a left coprime matrix fraction description of $W(s)$, $rank\,[P_1(\lambda^0) \quad B_1(\lambda^0)] < \beta$.

(ii) The following holds:

W_{11} : no entry has a pole at λ^0

W_{12} : λ^0 is simple zero of characteristic polynomial of this block

W_{21} : every entry has a zero at λ^0

W_{22} : no entry has a pole at λ^0

Davison and Ozguner [60]:

A recursive characterization of decentralized fixed modes of an N control agent systems in terms of an $N-1$ control agent system is provided by Davison and Ozguner 1983. Also, they gave an interpretation of the fixed modes of diagonal systems in terms of the system's transfer function matrix. The following is a summary of their development.

Theorem 1.2.5 [60]:

(i) Given the N-control agent decentralized system (1.2.1) with $N \geq 3$, then $\lambda \in \sigma(A)$ is not a decentralized fixed mode of (1.2.1) if and only if λ is not a decentralized fixed mode of any of the following N control agent systems of (1.2.1)

$$
\begin{aligned}
&(1)\left\{ \begin{bmatrix} \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \\ C_3 \\ \vdots \\ C_N \end{bmatrix}, A, [(B_1, B_2), B_3, \ \dots\ , B_N] \right\} \\
&(2)\left\{ \begin{bmatrix} C_1 \\ \begin{pmatrix} C_2 \\ C_3 \end{pmatrix} \\ \vdots \\ C_N \end{bmatrix}, A, [B_1, (B_2, B_3), \ \dots\ , B_N] \right\} \\
&\qquad\vdots
\end{aligned}
\tag{1.2.13}
$$

$$(N-2)\left\{\begin{bmatrix} C_1 \\ \vdots \\ \begin{pmatrix} C_{N-2} \\ C_{N-1} \end{pmatrix} \\ C_N \end{bmatrix},\ A,\ [B_1,\ \ldots,\ (B_{N-2},B_{N-1}),\ B_N]\right\}$$

$$(N-1)\left\{\begin{bmatrix} C_1 \\ \vdots \\ C_{N-2} \\ \begin{pmatrix} C_{N-1} \\ C_N \end{pmatrix} \end{bmatrix},\ A,\ [B_1,\ \ldots,\ B_{N-2},\ (B_{N-1},\ B_N)]\right\}$$

$$(N)\left\{\begin{bmatrix} C_1 \\ \vdots \\ C_{N-3} \\ \begin{pmatrix} C_{N-2} \\ C_N \end{pmatrix} \\ C_{N-1} \end{bmatrix},\ A,\ [B_1,\ \ldots,\ B_{N-3},\ (B_{N-2},\ B_N),\ B_{n-1}]\right\}$$

(ii) Given the N-control agent decentralized system (1.2.1) with $N=2$, then $\lambda \in \sigma(A)$ is not a decentralized fixed mode of (1.2.1) if and only if the following three conditions all hold:

1. λ is not a centralized fixed mode of $\left\{\begin{pmatrix} C_1 \\ C_2 \end{pmatrix},\ A,\ (B_1,\ B_2)\right\}$, i.e.

$$\begin{aligned} &rank\ (A-\lambda I,\ B_1,\ B_2) = n, \\ &rank \begin{bmatrix} A-\lambda I \\ C_1 \\ C_2 \end{bmatrix} = n \end{aligned} \tag{1.2.14}$$

2. $\operatorname{rank} \begin{pmatrix} A-\lambda I & B_1 \\ C_2 & 0 \end{pmatrix} \geq n$

3. $\text{rank} \begin{pmatrix} A - \lambda I & B_2 \\ C_1 & 0 \end{pmatrix} \geq n$

Davison and Ozguner have also developed a result for characterizing decentralized fixed modes using a transfer function matrix. Consider an N-control agent system with transfer function matrix $W(s)$.

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \\ \vdots \\ Y_N(s) \end{bmatrix} = W(s) \begin{bmatrix} U_1(s) \\ U_2(s) \\ \vdots \\ U_N(s) \end{bmatrix}$$

Assume that $W(s)$ can be factored as

$$W(s) = \frac{A_1}{s - \lambda_1} + \sum_{i=2}^{n} \frac{A_i}{s - \lambda_i} \qquad (1.2.15)$$

where $A_1 \neq 0$ and $\lambda_i \neq \lambda_1$, $i = 2,3, \ldots ,n$. The following result is obtained for cases $N = 2,3$. The case $N = 4$ is presented in [60].

Theorem 1.2.6 [60]: λ_1 is not a decentralized fixed mode of (1.2.15) if and only if none of the following conditions occur with respect to the matrix A_1 :

$$\left\{ W(s) - \frac{A_1}{s - \lambda_1} \right\}_{s = \lambda_1}$$

nor with the transpose

$$\left\{ W(s) - \frac{A_1}{s-\lambda_1} \right\}^T_{s=\lambda_1}$$

Case 1: $(N = 2)$

$$(i)\ A_1 = \begin{bmatrix} 0 & X \\ 0 & 0 \end{bmatrix} \text{ and } \left\{ W(s) - \frac{A_1}{s-\lambda_1} \right\}_{s=\lambda_1} = \begin{bmatrix} X & X \\ X & 0 \end{bmatrix} \quad (1.2.16)$$

Case 2: $(N = 3)$

$$(i)\ A_1 = \begin{bmatrix} 0 & 0 & X \\ 0 & 0 & X \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \left\{ W(s) - \frac{A_1}{s-\lambda_1} \right\}_{s=\lambda_1} = \begin{bmatrix} X & X & X \\ X & X & X \\ 0 & 0 & X \end{bmatrix}$$

$$(ii)\ A_1 = \begin{bmatrix} 0 & X & X \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \left\{ W(s) - \frac{A_1}{s-\lambda_1} \right\}_{s=\lambda_1} = \begin{bmatrix} X & X & X \\ 0 & X & X \\ 0 & X & X \end{bmatrix}$$

$$(iii)\ A_1 = \begin{bmatrix} 0 & X & 0 \\ 0 & 0 & 0 \\ 0 & X & 0 \end{bmatrix} \text{ and } \left\{ W(s) - \frac{A_1}{s-\lambda_1} \right\}_{s=\lambda_1} = \begin{bmatrix} X & X & X \\ 0 & X & 0 \\ X & X & X \end{bmatrix}$$

where X denotes elements whose values are not necessarily zero.

Condition (i) for the case $N = 2$ has the following interpretation. After cancellation, $W_{11}(s)$, $W_{21}(s)$ and $W_{22}(s)$ have no elements with a pole λ_1 and that $W_{21}(s)$ has a zero at λ_1 for all elements of $W_{21}(s)$.

Furthermore, Davison and Ozguner showed that the conjecture made by Fessas 1979 is true only for two-control agents systems, they state the following conclusions.

(i) The 'if' part of Fessas's conjecture [26] is true for $N = 2$ and is false for $N \geq 3$.

(ii) The 'only if ' part of Fessas's conjecture [26] is false for $N \geq 2$.

Seraji [61]:

Seraji 1982 has established conditions for existence and characterization of fixed modes which employs a transfer function matrix description. It is shown that for scalar local controllers in the diagonal control structure a necessary and sufficient condition for a system pole at $s = \lambda$ to be a fixed mode is that λ be a transmission zero of all subsystems formed by selecting the same inputs and outputs of the system. In the following is a presentation of Seraji's major result.

Consider the m-input m-output linear multivariable system $Y(s) = G(s)\,U(s)$, where $G(s)$ is the mxm transfer function matrix, $U(s)$ is the m-input vector and $Y(s)$ is the m-output vector. Then $\det G(s) = N(s)/F(s)$ where $N(s)$ is the ***zero polynomial*** with maximum order n whose roots are defined to be the ***transmission zeros*** of the system and $F(s)$ is the n^{th} order open loop characteristic polynomial of the system. It is assumed that the decentralized controller has diagonal structure with scalar local controllers. Accordingly, the controller has the following structure

$$u_i = k_i [v_i - y_i], \qquad i = 1, \ \ldots \ , m$$

where k_i is the i^{th} constant controller and v_i is the i^{th} reference input. The closed-loop characteristic polynomial is then

$$H(s) = F(s) \det[I + G(s) K]$$

By defining $N_{i_1, \_ , i_j}(s)$ as the zero polynomial of the j-dimensional subsystem formed by selecting the j inputs and outputs of the system as $\{(u_{i1}, \ldots , u_{ij})\}$; $\{(y_{i1}, \ldots , y_{ij})\}$ and noting that $N_{i_1, \_ , i_j}(s) = F(s) \det G_{i_1, \_ , i_j}(s)$, where $G_{i_1, \_ , i_j}(s)$, is the jxj matrix formed by selecting the j columns $i_1, \ldots , i_j$ of $G(s)$, we obtain

$$H(s) = F(s)\left(1 + (tr\, G(s) K) + \frac{\phi(s,K)}{F(s)} + \det (G(s) K)\right)$$

$$F(s) + \sum_{i=1}^{m} k_i N_i(s) + \phi(s,K) + \left(\prod_{i=1}^{m} k_i \right) N(s) \qquad (1.2.17)$$

$$= F(s) + \sum_{j=1}^{m} \sum_{[i_1, \ldots , i_j]} k_{i1} \times \ldots \times k_{ij} \times N_{i_1, \ldots , i_j}(s)$$

where the term $\phi(s,K)$ is the sum of all possible partial products involving $j = 2,3, \ldots , m-1$ elements of the set $\{(k_1, \ldots , k_j)\}$ as $k_{i1} \times \ldots \times k_{ij}$ multiplied by the corresponding zero polynomial $N_{i_1, \_ , i_j}(s)$.

Now we can state the following result:

Theorem 1.2.7 [61]: A necessary and sufficient condition for a pole at $s = \lambda$ of the open loop system to be a fixed mode under scalar local controllers in the

diagonal structure is that $s = \lambda$ be a common transmission zero of all subsystems of dimensions $j = 1,2, \ldots ,m$ formed by selecting the same inputs and outputs of the system. In other words, it is required that $N_{i_1, \ldots ,i_j}(\lambda) = 0$.

An extension to non-diagonal controller structure has also been provided by Seraji [61]. For this general structure the non-diagonal controller matrix K whose entries are 0 or the m gains $k_1, \ldots , k_m$ in the columns $i = 1,2, \ldots m$ is converted to a diagonal matrix $\hat{K}$ such that $K = P\hat{K}$, where $\hat{K} = diag\{k_i\}$ and P is an mxm matrix whose elements are 0 or 1. Thus the pair $\{G(s),K\}$ is replaced by $\{\hat{G}(s),K\}$ where $\hat{G} = G(s)\,P$, and the previous theorem can be applied.

Example 1.2.1 [61]: Consider a two station system with

$$G(s) = \frac{1}{(s+1)(s-1)(s-2)} \begin{bmatrix} (s+1)(s-2) & (s+1)(s-1) \\ 0 & (s-1)(s-2) \end{bmatrix}, \quad K = \begin{bmatrix} 0 & k_2 \\ k_1 & 0 \end{bmatrix}$$

Thus we have

$$P = K\,diag\left\{\frac{1}{k_i}\right\} = \begin{bmatrix} 0 & k_2 \\ k_1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{k_1} & 0 \\ 0 & \frac{1}{k_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

then

$$\hat{G}(s) = G(s)\,P = \frac{1}{(s+1)(s-1)(s-2)} \begin{bmatrix} (s+1)(s-1) & (s+1)(s-2) \\ (s-1)(s-2) & 0 \end{bmatrix}$$

To obtain the fixed polynomial, we evaluate the following zero polynomials:

$$
\begin{aligned}
&(i)\ \hat{N}_1(s) = F(s)\,\hat{g}_{11}(s) = (s+1)(s-1) \\
&(ii)\ \hat{N}_2(s) = F(s)\,\hat{g}_{22}(s) = 0 \\
&(iii)\ \hat{N}(s) = F(s)\ \det\ \hat{G}(s) = -(s-2)
\end{aligned}
\tag{1.2.18}
$$

It is apparent that $\hat{N}_1(s)$, $\hat{N}_2$, $\hat{N}(s)$ and $F(s)$ do not have a common root, therefore the system has no decentralized fixed modes.

Vidyasagar and Viswanadham [32]:

Another algebraic decentralized fixed mode characterization using transfer function matrix description is provided by Vidyasagar and Viswanadham [32]. Their work is based on determinantal expansions and the Binet-Cauchy formula. They provided a formula for computing the fixed polynomial of a system under decentralized feedback as the greatest common divisor of certain minors of the system transfer function matrix and its characteristic polynomial.

Consider the linear system (1.2.1) and let $G(s) = C(sI - A)^{-1}B$ denote the transfer function matrix of the system. The characteristic polynomial $G(s)$ can be defined as the monic least common multiple of the denominators of all minors of $G(s)$. And let $\phi(s)$ denote the characteristic polynomial.

By applying the decentralized feedback (1.2.2), the closed-loop system state representation is: $\dot{x} = (A - BKC)\,x$, where $K = \mathit{Block\ Diag}\,\{K_1, \ldots, K_N\}$.

Theorem 1.2.8 [32]: The decentralized fixed polynomial of (1.2.1) and (1.2.2) is

$$\alpha = g.c.d.\left\{\phi; P\begin{pmatrix} I_{i_1} \cup I_{i_2} \cup \ \ldots \ \cup I_{i_j} \\ J_{i_1} \cup J_{i_2} \cup \ \ldots \ \cup J_{i_j} \end{pmatrix}\right\}, \qquad (1.2.19)$$

where $\{i_1, \ \ldots \ , i_j\}$ is a subset of $\{1,2, \ \ldots \ ,N\}$; $I_{i_s} \subset P_{i_s}$ and $J_{i_s} \subset Q_{i_s}$, where $P_1 = \{1, 2, \ \ldots \ , p_1\}$, $P_2 = \{p_1 + 1, \ \ldots \ , p_1 + p_2\}$, ... etc. And Q_{i_s} is defined in the same manner.

Example 1.2.2 [32],[19]: Consider a system controlled by two stations with $p_1 = 1$, $q_1 = 3$ and $p_2 = 2$, $q_2 = 1$. Thus the controller is a block diagonal $3x4$ matrix with the following structure:

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} & 0 \\ 0 & 0 & 0 & k_{24} \\ 0 & 0 & 0 & k_{34} \end{bmatrix}$$

Let the $4x3$ transfer function matrix be:

$$G(s) = \begin{bmatrix} \frac{1}{s(s-1)} & 0 & 0 \\ \frac{1}{s-1} & 0 & 0 \\ 0 & 0 & \frac{1}{s-1} \\ \frac{1}{s-1} & \frac{1}{s-1} & 0 \end{bmatrix}$$

The fixed polynomial is calculated by

$$\alpha = g.c.d. \left\{ \begin{matrix} \phi; p\begin{pmatrix}1\\1\end{pmatrix}, p\begin{pmatrix}2\\1\end{pmatrix}, p\begin{pmatrix}3\\1\end{pmatrix}; p\begin{pmatrix}4\\2\end{pmatrix}, p\begin{pmatrix}4\\3\end{pmatrix}; \\ p\begin{pmatrix}14\\12\end{pmatrix}, p\begin{pmatrix}24\\12\end{pmatrix}, p\begin{pmatrix}34\\12\end{pmatrix}, p\begin{pmatrix}14\\13\end{pmatrix}, p\begin{pmatrix}24\\13\end{pmatrix}, p\begin{pmatrix}34\\13\end{pmatrix} \end{matrix} \right\} \qquad (1.2.20)$$

The greatest common divisor of the $\phi = s(s-1)^3$ and all the above minors is equal to $(s-1)$. Therefore this system has a fixed mode at $s = 1$.

1.2.2 Survey of Decentralized Control Design

This subsection surveys the status of decentralized control design. The decentralized control problem is to synthesize N controllers for a plant such that the closed loop system is stable and satisfies certain design constraints. Before we start listing the existing decentralized control design methods, it is necessary to emphasize that in spite of the vast progress decentralized control has established in industrial and management systems, the designs used are based on ad hoc methods [62]. The researchers in this field still feel that there is a lot of work that needs to be done in the area of decentralized control theory ingeneral, and the design problem in particular. And it is necessary to establish a unified theory of decentralized control that solves problems of stability, performance index issues, cost of communications ... and others [17],[62].

In the previous subsection issues of stability and existence of fixed modes have been surveyed. Here we will survey results related to existence of decentralized robust controllers and their synthesis.

The decentralized control design problem was studied by several researchers. Davison [22],[24],[66] discussed regulation and the general decentralized servomechanism problem; Davison [64] treated the decentralized robust control of unknown systems; Davison [65] studied sequential stable robust controllers. Davison and Chang [25] developed a method of decentralized control design which is an extension of the centralized method of Davison and Ferguson [66]. Bernussou and Titli [67] outlined a design technique based on parametric optimization. Hassan and Singh [68] considered decentralized control design using model followers. Viswanadham and Ramakrishna [52] have formulated a decentralized servomechanism problem in the same setting as Corfmat and Morse [19]. First we consider the approach outlined by Davison 1976 [24].

Davison 1976 :

Davison [24],[63] [64] developed necessary and sufficient conditions for a solution of the 'robust decentralized servomechanism problem' to exist, together with a characterization of all robust decentralized controllers which solves the servomechanism problem. Assume the open loop system is described by the following linear time invariant model:

$$\begin{aligned} \dot{x} &= A\,x + \sum_{i=1}^{N} B_i\,u_i + E\,\omega \\ y_i &= C_i\,x + D_i\,u_i + F_i\,w\,, \\ e_i &= y_i - y_i^{ref} \qquad\qquad i = 1,\ \ldots\ ,N \end{aligned} \tag{1.2.21}$$

where $\omega \in R^{\Omega}$ is a disturbance vector, e_i, $i = 1, \ldots, N$ is the error in the system and y_i^{ref}, $i = 1, \ldots, N$ is the reference input signal. Moreover, the disturbance vector ω is assumed to satisfy the following model:

$$\begin{aligned} \dot{\sigma}_1 &= \hat{A}_1 \sigma_1 \\ \omega &= \hat{C}_1 \sigma_1 \end{aligned} \tag{1.2.22}$$

where $\sigma_1 \in R^{\hat{n}_1}$. The desired output is assumed to satisfy

$$\begin{aligned} \dot{\sigma}_2 &= \hat{A}_2 \sigma_2 \\ \gamma &= \hat{C}_2 \sigma_2 \\ y^d &= R\gamma \end{aligned} \tag{1.2.23}$$

The decentralized controller is to be assumed of the following structure:

$$\begin{aligned} \dot{\eta} &= \tilde{C}\eta + \tilde{B}e \\ u &= K_0 \hat{x} + K_1 \eta \end{aligned} \tag{1.2.24}$$

where η is the output of the general servo-compensator, $\hat{x}$ is the output of the stabilizing compensator $\tilde{S}$ with inputs y_m, u and η. The matrices $\tilde{S}, K_0, K_1$ are found to stabilize the composite system of states $[x, \eta]^T$.

Definition 1.2.3 [24]: Given the system (1.2.21), suppose there exists a decentralized controller so that the resultant system is stable and $e \to 0$ as $t \to \infty$, for every $X(0) \in R^n$. Suppose that the plant is perturbed such that:
$A \to A + \delta A$, $B \to B + \delta B$,and $C \to C + \delta C$, where $\delta A \in \Omega_\varepsilon$, $\delta B \in \Omega_\varepsilon$,

$\delta C \in \Omega_\varepsilon$,and $\Omega_\varepsilon = \{a|, \ |a| < \varepsilon, \varepsilon > 0\}$. Then if the system is still stable and asymptotic regulation still occurs, then the controller is said to be a robust controller.

Before introducing the conditions for existence of the robust decentralized controllers the following preliminaries are required. Let α_i, $i = 1, \ \ldots \ , \rho$ denote the coefficients of input system's characteristic polynomial

$$\prod_{i=1}^{\hat{\rho}} \lambda - \lambda_i^{\rho_i} = \lambda^\rho + \alpha_\rho \lambda^{\rho-1} + \ \ldots \ , \alpha_2 \lambda + \alpha_1 \tag{1.2.25}$$

where $\rho = \sum_{i=1}^{\hat{\rho}} \rho_i$. Define $\tilde{C}$ as

$$\tilde{C}^T = (\tilde{C}_1^T, \tilde{C}_2^T, \ \ldots \ , \tilde{C}_N^T) \tag{1.2.26}$$

where

$$\tilde{C}_1 = \begin{bmatrix} C_1 & 0 & 0 & \ldots & 0 \\ 0 & I_{p_1} & 0 & \ldots & 0 \end{bmatrix}$$

$$\tilde{C}_2 = \begin{bmatrix} C_2 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & I_{p_2} & 0 & \ldots & 0 \end{bmatrix}$$

$$\tilde{C}_N = \begin{bmatrix} C_N & 0 & \ldots & 0 \\ 0 & 0 & \ldots & I_{p_N} \end{bmatrix}$$

With the above notation together with the definition of the fixed modes (Definition 1.2.1), Davison [24] proved the following result.

Theorem 1.2.9 [24]: For a stable linear system (1.2.21) a necessary and sufficient condition for the existence of a robust decentralized controller for all disturbances ω of (1.2.22), all desired reference inputs y^d of (1.2.23), and stability of the closed loop system is that the fixed modes of

$$\left(\tilde{C}, \begin{bmatrix} A & 0 \\ C & \lambda_j I \end{bmatrix}, \begin{bmatrix} B \\ D \end{bmatrix}\right), \qquad j = 1, \ \ldots \ \rho \tag{1.2.27}$$

with respect to K of (1.2.3) do not contain any λ_j, $j = 1, \ \ldots \ , \rho$.

A constructive algorithm has been provided by Davison [63] for synthesizing the robust feedback controllers for desired reference inputs.

Algorithm 1.2.1 [63]:

Step 1: Form the augmented state

$$\dot{\eta}_1 = e \ , \qquad u_1 = \alpha_1 K \eta_1 \tag{1.2.28}$$

where k is the pseudo-inverse of the 'steady state tracking gain matrix' $T_1(1,1)$, (steady state tracking matrices are defined in [63], they are considered as a measure of the steady state output of the system for different decentralized controllers). The scalar α_1 is chosen such that the closed loop system poles i.e. the eigenvalues of the matrix

$$A_{c_1} = \begin{bmatrix} A & \alpha_1 B K \\ C & \alpha_1 D K \end{bmatrix} \tag{1.2.29}$$

are in the left-hand s-plane satisfying some design requirements.

Step 2: Introduce a second state

$$\dot{\eta}_2 = \eta_1 \ , \qquad u_2 = \alpha_2 K \eta_2 \tag{1.2.30}$$

as in *Step* 1 , α_2 is found such that the eigenvalues of the closed loop matrix

$$\begin{bmatrix} A & \tilde{\alpha}_1 B K & \alpha_2 B K \\ C & \tilde{\alpha}_1 D K & \alpha_2 D K \\ 0 & I & 0 \end{bmatrix} \tag{1.2.31}$$

has some desired eigenvalues. Where $\tilde{\alpha}_1$ is the optimum value of *Step* 1 .

. .

. .

. .

Step k: Let the k^{th} augmented state be

$$\dot{\eta}_k = \eta_{k-1} , \qquad u_k = \alpha_k K \eta_k \tag{1.2.31}$$

Similarly, α_k is obtained such that the closed loop augmented matrix has a desired set of eigenvalues.

Example 1.2.3 [47]: Consider a linear system described by (1.2.21) with $N = 2$, $q_1 = q_2 = 1$, $p_1 = p_2 = 1$ and $n = 2$. Let

$$A = \begin{bmatrix} -1 & .1 \\ 0 & -2 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 1 \\ .5 \end{bmatrix}$$

$$C_1 = [1 \quad 0] , \qquad C_2 = [0 \quad 1]$$

$$D_1 = [1] , \quad D_2 = [0]$$

$$F_1 = [0] , \quad F_2 = [0]$$

It is required to find a robust decentralized controller which achieves stability and asymptotic regulation with $y_1^d = .5$ and $y_2^d = -.5$. The steady state tracking gains are evaluated using Algorithm 1.2.1:

Let $u_1 = 1, \quad u_2 = 0$ then

$$T_1(1,1) = y_1^1 = \lim_{t\to\infty} y_1(t) = 1.05$$

$$T_1(1,2) = y_2^1 = \lim_{t\to\infty} y_2(t) = .5$$

and $u_1 = 0, \quad u_2 = 1$ then

$$T_1(2,1) = y_1^1 = \lim_{t\to\infty} y_1(t) = 1.025$$

$$T_1(2,2) = y_2^1 = \lim_{t\to\infty} y_2(t) = -.25$$

The first augmented state is

$$\begin{aligned} \dot{\eta}_1 &= e_1 = x_1 + u_1 - .5 \\ u_1 &= -\alpha_1 [\tilde{T}_1(1,1)]^{-1} \eta_1 = -(1/1.05)\alpha_1 \eta_1 \end{aligned} \tag{1.2.32}$$

By applying the feedback control (1.2.32) the closed loop system is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\eta}_1 \end{bmatrix} = \begin{bmatrix} -1 & .1 & 0 \\ 0 & -2 & -\alpha_1/1.05 \\ 1 & 0 & -\alpha_1/1.05 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \eta_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -.5 \end{bmatrix}$$

The optimum value of α_1 was $\tilde{\alpha} = .65$ with closed loop poles $(-2.04, -.788 \pm j.1214)$. Now let us introduce the second state:

$$\begin{aligned} \dot{\eta}_2 &= e_2 = x_2 + .5 \\ u_2 &= -\alpha_2 [\tilde{T}_1(2,2)]^{-1} \eta_2 = 8.4\,\alpha_2 \eta_2 \end{aligned} \tag{1.2.33}$$

The augmented feedback system is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\eta}_1 \\ \dot{\eta}_2 \end{bmatrix} = \begin{bmatrix} -1 & .1 & 0 & 8.4\alpha_2 \\ 0 & -2 & -\tilde{\alpha}_1/1.05 & 4.2\alpha_2 \\ 1 & 0 & -\tilde{\alpha}_1/1.05 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \eta_1 \\ \eta_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -.5 \\ .5 \end{bmatrix}$$

For $\tilde{\alpha}_1 = .65$ the optimal value of α_2 was $\tilde{\alpha}_2 = .046$ with the following closed loop poles $(-2.05, -1.23, -.166 \pm j.138)$. With this pole configuration and the controller given by (1.2.32) and (1.2.33), the system is asymptotically regulated with settling times of 20.8 and 21.0 seconds for $y_1(t)$ and $y_2(t)$, respectively.

Bernussou and Titli 1982 :

The design method of Bernussou and Titli [67] is based on parametric optimization. The problem is formulated such that a quadratic performance index is minimized subject to certain constraints. Consider the linear time invariant system

$$\begin{aligned} \dot{x}_i(t) &= \sum_{j=1}^{N} A_{ij}\, x_j(t) + B_{ii}\, u_i(t), \qquad x_i(0) = x_{i0}, \\ y_i(t) &= C_{ii}\, x_i(t), \qquad i = 1, \ \ldots \ , N \end{aligned} \tag{1.2.34}$$

The decentralized feedback controller is assumed to be static satisfying the following relation:

$$u_i = -K_i y_i, \qquad i = 1, \ \ldots \ , N \tag{1.2.35}$$

where $K_i \in R^{q_i \times p_i}$, $i = 1, \ \ldots \ , N$ are the constant gain matrices. The design problem is to find a decentralized controller (1.2.12) such that the following performance index is minimized.

$$J = \sum_{i=1}^{N} \int_0^T (x_i^T Q_i x_i + u_i^T R_i u_i)\, dt + x_i^T(\Gamma) S_i x_i(\Gamma)$$

To eliminate the effect of initial conditions x_{i0} and the integration limit T the following optimization model is suggested by Bernussou and Titli [67]

$$\begin{aligned} &\min_{K \in \mathrm{K}_s} \hat{J} = tr\, P \\ &\textit{subject to} \\ &(A - BKC)^T P = P(A - BKC) + Q + C^T K^T R K C = 0 \end{aligned} \tag{1.2.36}$$

where $\mathrm{K}_s \subset \mathrm{K}$ is the set of block diagonal matrices which makes the eigenvalues of the the matrix $(A - BKC)$, and P is a solution of(1.2.36). More constraints

can be added such that the feedback matrices K_i are to satisfy additional constraints [67].

Hassan and Singh 1980 :

A brief discussion of the approach of Hassan and Singh [68] will be presented. Their approach is based on synthesizing model followers where a 'crude' model of the interconnection between the subsystems is introduced, then the controller gain matrices are obtained by solving appropriate optimization problems [68]. Consider another form of (1.2.34)

$$\begin{aligned} \dot{x}_i &= A_{ii}\, x_i + B_i\, u_i + G_i\, z_i, \qquad i = 1, \ \ldots \ ,N \\ z_i &= \sum_{j=1}^{N} L_{ij}\, x_j \end{aligned} \tag{1.2.37}$$

where it is assumed that $y_i = x_i$, $i = 1, \ \ldots \ ,N$. Let the interconnection model be

$$\hat{z}_i = A_{z_i}\, \hat{z}_i$$

The feedback decentralized dynamic controller has the following structure:

$$u_i = -K_i^1\, x_i - K_i^2\, \tilde{z}_i \tag{1.2.38}$$

where $\tilde{z}_i$ is an estimate of the interconnection z_i . Furthermore, $\tilde{z}_i = \hat{z}_i - F_i\,(x_i - \hat{x}_i)$, where F_i is obtained such that the error between subsystem i and the subsystem reference model is minimized.

Viswanadham and Ramakrishna 1981 :

Viswanadham and Ramakrishna [52] have obtained conditions under which a large linear multivariable system can be regulated using local decentralized controllers. The conditions are in terms of ranks of certain polynomial matrices. Part of these conditions are equivalent to the fixed modes of Davison in Theorem (1.2.1). The approach used in obtaining the conditions follows the geometric methods of Corfmat and Morse [19]. Consider the linear system of (1.2.21)-(1.2.23) with $D = F = 0$. Let the decentralized compensator has the following structure:

$$\begin{aligned} \dot{z}_i(t) &= M_i z_i(t) + N_i e_i(t) \\ u_i(t) &= K_{i1} y_i(t) + K_{i2} z_i(t) , \qquad i = 1, \ldots, N \end{aligned} \tag{1.2.39}$$

where K_{i1} and K_{i2} are chosen such that the overall system maintains stability and asymptotic regulation. The main result of Viswanadham and Ramakrishna is

Theorem 1.2.10 [52]: Let the linear system be described by (1.2.38) and the servo-controller be given by (1.2.39) then

(a) There exists a decentralized controller such that the poles of the overall system are assigned arbitrarily if and only if the following three conditions hold.

$$
\begin{aligned}
&(i) \quad C_{\bar{k}-h}(sI-A)^{-1}B_h \neq 0, && h \in \tilde{k} \\
&(ii) \quad rank\begin{bmatrix} sI-A & B_h \\ C_{\bar{k}-h} & 0 \end{bmatrix} \geq n, && \forall\ s,h \in \tilde{k} \\
&(iii) \quad rank\begin{bmatrix} \lambda_j I-A & B_i \\ C & 0 \end{bmatrix} \geq n+p_i, && i=1,\ \ldots\ ,N
\end{aligned}
\tag{1.2.40}
$$

where λ_j is an eigenvalue of the input system, $\bar{k} = \{1,2,\ \ldots\ ,k\}$ and $\tilde{k}$ is the set of all nonempty proper subsets of $\bar{k}$.

(b) Such cotrollers achieve asymptotic tracking and disturbance rejection.

The first two conditions of (a) are equivalent to the fixed mode conditions of Davison. The third condition requires that the modes of the input system (disturbance and reference inputs) should not coincide with the transmission zeros of (C,A,B).

Chapter 2: Polynomial Matrices

2.1 Introduction

This chapter reviews a number of mathematical concepts which are essential for understanding the subsequent chapters. The material is intended rather for a quick review than for a detailed reading. Only relevant topics are introduced. All proofs are omitted since they can be found in the cited references.

In Section 2.2 the algebraic concept of polynomial matrices as a ***ring*** is briefly introduced. The concepts of elementary operations, equivalence and the Smith form of polynomial matrices are introduced in Section 2.3. In Section 2.4 the division and coprimeness of polynomial matrices are studied. Section 2.5 covers row and column degrees of polynomial matrices. The last section is devoted to properness and polynomial matrix fractions of rational matrices. The material discussed in this chapter can be found in [10], [9], [3], [5].

Appendix D includes detailed algorithms for operations on polynomial matrices. Appendix E contains the developed program listings for these algorithms.

2.2 *The Ring of Polynomial Matrices*

The class of polynomial matrices plays an important role in linear system theory. They differ from scalar matrices in one main aspect. In particular, the elements of polynomial matrices are polynomials which belong to a *commutative ring*. Unlike the set of real numbers which forms a field, the set of polynomials forms a commutative ring. The difference between a field and a ring is that the elements of a ring do not have a multiplicative inverse while the elements of the field do.

The set of polynomial matrices forms a ring because not every polynomial matrix has a polynomial matrix inverse. Moreover, because matrix multiplication is not commutative, the ring of polynomial matrices is a noncommutative ring. The units of this ring are a class of polynomial matrices called *unimodular* matrices defined as:

Definition 2.2.1 : A unimodular matrix $A(s)$ is defined as any square matrix with $\det A(s)$ a nonzero real number.

Because the determinant of a unimodular matrix $A(s)$ is a nonzero real number then the inverse of $A(s)$ is also a polynomial matrix.

2.3 Equivalence and the Smith Canonical Form

In this section the concept of equivalence of polynomial matrices is introduced which naturally leads to the Smith canonical form. The process of reducing a polynomial matrix $A(s)$ to its Smith canonical form requires only elementary row and column operations on polynomial matrices.

Row and Column Operations:

Similar to scalar matrices, the following three elementary row (column) operations are defined for a polynomial matrix $A(s)$ [5], [6]:

1. Interchanging any two rows (columns).

2. Multiplication of a row (column) by a nonzero real number.

3. Addition of $p(s)$ times one row (column) to another row (column), where $p(s)$ is a nonzero polynomial.

Performing the above row operations is equivalent to premultiplying (for column operations postmultiplying) $A(s)$ by elementary matrices. Taking $3x3$ elementary matrices for simplicity we have:

$$E_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \textit{for interchanging the first and second rows}$$

$$E_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \textit{for multiplying the second rows by } \lambda \neq 0$$

$$E_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & p(s) & 1 \end{bmatrix} \qquad \textit{for adding } p(s) \textit{ times the second rows to the third row}$$

Because each $\det E_i$, $i = 1,2,3$ is a nonzero real number, it follows that performing a sequence of elementary row (column) operations on $A(s)$ is equivalent to premultiplying (postmultiplying) $A(s)$ by a unimodular matrix $U(s)$, where $U(s)$ is the product of the elementary matrices corresponding to the sequence of row (column) operations performed on $A(s)$.

Equivalence of polynomial matrices

Definition 2.3.1 : Let A and B be an mxn polynomial matrices then:

- if $A = U_1 B$ where U_1 is an mxm unimodular polynomial matrix, then A and B are ***left equivalent.***

- if $A = B U_2$ where U_2 is an nxn unimodular polynomial matrix, then A and B are ***right equivalent.***

- if $A = U_1 B U_2$ where U_1 and U_2 are mxm and nxn unimodular polynomial matrix, then A and B are ***equivalent.***

From the above definitions it is clear that two polynomial matrices $A(s)$ and $B(s)$ are left equivalent, right equivalent and equivalent if and only if one of them can be obtained from the other by a sequence of row,column, and row and column operations, respectively.

The Smith Canonical Form:

Each polynomial matrix has unique characteristics determined by certain polynomials which characterize the polynomial matrix uniquely up to a nonzero real number. Every mxn polynomial matrix A of rank r can be reduced by a sequence of elementary row and column operations to the Smith form:

$$S = diag\,[f_1, f_2, \ldots, f_r, 0, 0, \ldots, 0]$$

where $A = U_1 S U_2$ and U_1 and U_2 are unimodular matrices representing the sequence of elementary row and column operations performed on A. Since the elementary row and column operations are are not unique, it follows that U_1 and U_2 are also not unique. The polynomials f_i, $i = 1, \ldots r$ are called the ***invariant factors*** of A. The invariant factors have the following properties:

1. The invariant factors of a matrix A are determined by A uniquely up to the multiplication of a nonzero real number. In other words, if two different sequences of elementary row and column operations are performed on A, and the following Smith forms are obtained:

$$A = U_1 S U_2 \qquad with \quad S = diag\,[f_1, \ldots, f_r, 0, \ldots, 0]$$

and

$$A = \tilde{U}_1 \tilde{S} \tilde{U}_2 \qquad with \quad \tilde{S} = diag\,[\tilde{f}_1, \ \ldots \ , \tilde{f}_r, 0, \ \ldots \ , 0]$$

then

$$f_i = c_i \tilde{f}_i, \qquad i = 1, \ \ldots \ , r$$

where c_i, $i = 1, \ \ldots \ , r$ are nonzero real, numbers.

2. The second property of the invariant factors is

$$f_i \ \ divides \ \ f_{i+1}, \qquad i = 1, \ \ldots \ , r-1$$

And we have k trivial invariant factors if $f_1 = f_2 = \cdots = f_k = 1$ where $k \leq r$

3. The greatest common divisor of minors of order k of A, and equivalently of S is given by:

$$g_k = f_1 f_2 \ \ldots \ f_k, \qquad k = 1,2, \ \ldots \ , r$$

4. The determinant of an mxm nonsingular polynomial matrix A is proportional to the product of the invariant factors:

$$\begin{aligned} \det A &= \det (U_1 S U_2) \\ &= \det U_1 \ \det S \ \det U_2 \\ &= c_1 (f_1 f_2 \cdots f_m) c_2 \\ &\sim f_1 f_2 \cdots f_m \end{aligned}$$

where c_1 and c_2 are nonzero real numbers.

2.4 Division and Coprimeness

Definition 2.4.1 : Let A, L and, R be an nxm, nxq and qxm polynomial matrices, respectively. Then if $A = L R$ then:

L is a ***left divisor*** of A
R is a ***right divisor*** of A
A is a ***right multiple*** of L
A is a ***left multiple*** of R

Greatest Common Left/Right Divisor:

Definition 2.4.2 : Let A, B and G_1 be an mxm , nxq and nxn polynomial matrices, respectively. If G_1 is a left divisor of both A and B , then G_1 is called a ***common left divisor*** of A and B . Moreover, if G_1 is a right multiple of every common left divisor of A and B , then it is called ***the greatest common left divisor*** of A and B .

Similarly, let A, B and G_2 be an nxm, qxm and mxm polynomial matrices, respectively. If G_2 is a right divisor of both A and B, then G_2 is called a ***common right divisor*** of A and B. Moreover, if G_2 is a left multiple of every common right divisor of A and B, then it is called ***the greatest common right divisor*** of A and B.

Left and Right Coprimeness

Definition 2.4.3 : Let A, and B be an nxm and nxq polynomial matrices. Then A and B are ***left coprime*** if their greatest common left divisors are unimodular matrices.

Similarly, let A, and B be an nxm and qxm polynomial matrices. Then A and B are ***right coprime*** if their greatest common right divisors are unimodular matrices.

As a corollary of the Smith form reduction theorem, it can be shown [4] that an nxm and nxq polynomial matrices A and B are left coprime if and only if the pair of matrices $([A \;\; B], [I_n \;\; 0])$ are left equivalent, where I_n is an nxn identity matrix. Similarly, an nxm and qxm polynomial matrices A and B are right coprime if and only if the pair of matrices $([A \;\; B]^T, [I_m \;\; 0]^T)$ are right equivalent, where I_m is an mxm identity matrix.

An algorithm is given in [3], for finding the greatest common left divisor of two polynomial matrices A and B. The algorithm is based on reducing the matrix $[A \;\; B]$ to a generalized lower triangular matrix $[\overline{G}_1 \;\; 0]$ by a sequence of

elementary column operations. This amounts to post-multiplying the matrix $[A \quad B]$ by a unimodular matrix U_1

$$U_1 = \begin{bmatrix} \overline{P}_1 & R_1 \\ \overline{Q}_1 & S_1 \end{bmatrix}$$

such that the following matrix equations are satisfied

$$A P_1 + B Q_1 = G_1$$

$$A R_1 + B S_1 = 0$$

where P_1, Q_1 and G_1 are obtained by appending $(n - r)$ zero columns to $\overline{P}_1$, $\overline{Q}_1$ and $\overline{G}_1$, where r is the rank of $[A \quad B]$. And G_1 is the greatest common left divisor of A and B.

A dual algorithm to the above one is used for for finding the greatest common right divisor of two polynomial matrices A and B. The algorithm is based on reducing the matrix $[A \quad B]^T$ to a generalized lower triangular matrix $[\overline{G}_2 \quad 0]^T$ by a sequence of elementary row operations. This amounts to pre-multiplying the matrix $[A \quad B]^T$ by a unimodular matrix U_2

$$U_2 = \begin{bmatrix} \overline{P}_2 & \overline{Q}_2 \\ R_2 & S_2 \end{bmatrix}$$

such that the following matrix equations are satisfied

$$P_2 A + Q_2 B = G_2$$

$$R_2 A + S_2 B = 0$$

where P_2, Q_2 and G_2 are obtained by appending $(m-r)$ zero rows to $\overline{P}_2$, $\overline{Q}_2$ and $\overline{G}_2$, where r is the rank of $[A \quad B]^T$. And G_2 is the greatest common right divisor of A and B.

Clearly if G_1 is unimodular matrix then A and B are left coprime (but not necessarily right coprime). Similarly, if G_2 is unimodular matrix then A and B are right coprime (but not necessarily left coprime).

2.5 Row and Column Properness

Consider the *nxn* polynomial matrix $A(s)$. Let the maximum degree of elements of $A(s)$ be p, then $A(s)$ can be written as:

$$A(s) = A_p s^p + A_{p-1} s^{p-1} + \ldots + A_1 s + A_0 \qquad (2.5.1)$$

where the A_i, $i = 0,1, \ldots, p$ are the coefficient matrices of $A(s)$. We have the following results [3]:

A(s) is singular if det $A(s) = 0$ for every s, otherwise it is nonsingular

A(s) is stable if det $A(s)$ is a stable polynomial, otherwise it is unstable

A(s) is proper if det $A_p \neq 0$, otherwise it is improper

Definition 2.5.1 : The ***row degree*** of the i^{th} row of $A(s)$, denoted by $\delta_{ri} A(s)$, is defined as the maximum degree of all polynomials in that row. Similarly, the

column degree of the j^{th} column of $A(s)$, denoted by $\delta_{cj} A(s)$, is defined as the maximum degree of all polynomials in that column.

Definition 2.5.2 : A nonsingular nxn matrix $A(s)$ is called ***row proper*** if

$$deg \ \det \ A(s) = \sum_{i=1}^{n} \delta_{ri} A(s)$$

and $A(s)$ is called ***column proper*** if

$$deg \ \det \ A(s) = \sum_{j=1}^{n} \delta_{cj} A(s)$$

The above definitions show that a polynomial matrix $A(s)$ can be row proper, but not necessarily column proper, and vice versa. Before introducing the next theorem, we need to define the following terms. An nxn polynomial matrix $A(s)$ can be written as

$$A(s) = A_{hc} H_c(s) + A_{lc}(s) \tag{2.5.2}$$

where $H_c(s) = diag\{ s^{kj}, j = 1,2, \ ... \ ,n ; \ k_j = \delta_{cj} A(s) \}$ and $A_{lc}(s)$ is a polynomial matrix in which the j^{th} column has a degree smaller than k_j. The constant matrix A_{hc} is called the ***column-degree coefficient matrix***. Similarly, an nxn polynomial matrix $A(s)$ can be written as

$$A(s) = H_r(s) A_{hr} + A_{lr}(s) \tag{2.5.3}$$

where $H_r(s) = diag\{ s^{k_i},\ i = 1,2,\ \ldots\ ,n;\ k_i = \delta_{ri}\, A(s) \}$ and $A_{lr}(s)$ is a polynomial matrix in which the i^{th} row has a degree smaller than k_i. The constant matrix A_{hr} is called the ***row-degree coefficient matrix.***

Theorem 2.5.1 : An *nxn* polynomial matrix $A(s)$ is row proper (column proper) if and only if the row-degree (column-degree) coefficient matrix A_{hr} (A_{hc}) is nonsingular.

Theorem 2.5.2 [6]: For every polynomial matrix $A(s)$, there exist unimodular matrices $U(s)$ and $V(s)$ such that $U(s)\,A(s)$ and $A(s)\,V(s)$ are row proper and column proper, respectively.

Using elementary operations on polynomial matrices, an algorithm is presented in [8] to convert a matrix into a row proper or a column proper one.

2.6 Properness and Stability of Rational Matrices

An *nxm* matrix $T(s)$ with its elements rational polynomials is called rational matrix. As in the case of rational functions, rational matrices can be fractioned into numerator and denominator polynomial matrices.

Matrix Fractions:

Definition 2.6.1 : A ***left matrix fraction*** of an nxm rational matrix $T(s)$ is a pair of polynomial matrices $(N_l(s), D_l(s))$, such that $T(s) = D_l^{-1}(s)\, N_l(s)$, where $N_l(s)$ and $D_l(s)$ are nxm and nxn polynomial matrices, respectively. In a similar way, a ***right matrix fraction*** of $T(s)$ is a pair of polynomial matrices $(N_2(s), D_2(s))$, such that $T(s) = N_2(s)\, D_2^{-1}(s)$, where $N_2(s)$ and $D_2(s)$ are nxm and mxm polynomial matrices, respectively.

If the pair $(N_l(s), D_l(s))$ are left coprime then the fraction is called a left coprime matrix fraction. If the pair $(N_2(s), D_2(s))$ are right coprime then the fraction is called a right coprime matrix fraction. The matrices $N_l(s)$ and $N_2(s)$ are called numerator matrices, while the matrices $D_l(s)$ and $D_2(s)$ are called denominator matrices.

Simple computational algorithms are presented in [3] to produce either left or right matrix fractions of a rational matrix. Moreover, the algorithms can convert any left matrix fraction to a right coprime matrix fraction and vice versa.

Properness of Rational Matrices:

A rational matrix $T(s)$ is proper if every element in it is a proper rational function. (i.e. the degree of the denominator polynomial is greater than or equal the degree of the numerator polynomial). The following is a useful Theorem

proved in [6] stating necessary and sufficient condition for properness in terms of polynomial matrix fractions:

Theorem 2.6.1 Let $T(s) = N(s)\,D(s)^{-1}$ where $N(s)$ and $D(s)$ are nxm and nxn polynomial matrices, respectively. And let $D(s)$ be column proper. Then $T(s)$ is proper if and only if

$$\delta_{cj}\,N(s) \leq \delta_{cj}\,D(s) \qquad \text{for } j = 1,2, \ \ldots\ , n$$

Stability of Rational Matrices:

Before discussing stability we need to introduce the concept of the characteristic polynomial of rational matrices [6].

Definition 2.6.2 : Let $T(s) = N_1(s)\,D_1^{-1}(s) = D_2^{-1}(s)\,N_2(s)$ where $N_1(s)$, $D_1(s)$, $N_2(s)$ and $D_2(s)$ are nxm , nxn , nxm and mxm polynomial matrices, respectively. It is assumed that N_1 and D_1 are left coprime, and N_2 and D_2 are right coprime. Then the ***characteristic polynomial*** of $T(s)$ is defined as

$$\det\ D_1(s) \qquad \text{or} \qquad \det\ D_2(s)$$

and the ***degree*** of $T(s)$ is defined as

$$\delta T(s) = deg\ \det\ D_1(s) = deg\ \det\ D_2(s)$$

Theorem 2.6.2 Let $T(s)$ be defined as in Definition 2.6.2. Then the $T(s)$ is asymptotically stable and BIBO stable if and only if all the roots of det $D_1(s)$ or det $D_2(s)$ have negative real parts.

Chapter 3: Decentralized Control Using a Polynomial Approach

3.1 Introduction

In this chapter a new approach is introduced for solving the problem of decentralized pole placement using the methods of polynomial matrix algebra [3], [6]. The mathematical model of the overall system (Plant and controllers) is formed using polynomial matrices. Conditions for stability, decentralization and realization are established in terms of polynomial matrices of the overall system and the decentralized controller.

The method of solution is based on the use of optimization theory, where the above conditions for stability, decentralization and realization are included as constraints to be satisfied while minimizing a certain objective function. The objective function is chosen such that, when minimized, the resulting poles of the closed loop system are as close as possible to a prespecified set.

We believe that the approach is applicable to a wide range of practical systems, and it has many advantages over existing methods. It is flexible enough to adopt and meet many design objectives; and when the technique does not

provide *the* minimum order controller it does an intensive search for minimizing the order.

Section 3.2 describes the general structure of the plant in relation with the decentralized controller. Section 3.3 presents the mathematical model of the plant and controller, in both transfer matrix and polynomial matrix fractions forms. Section 3.4 covers issues concerning fixed modes, realization and stability of the decentralized closed loop system. Section 3.5 presents the solution procedure of the pole placement problem. Section 3.6 concerns with a feedforward and feedback decentralized controller. In Section 3.7 the method used in minimizing the controller's order is discussed, and an upper bound of the controller's order is provided too. Three examples are presented in Section 3.8.

3.2 Controlling the Plant from Several Control Stations

The plant to be controlled, i.e. the open loop system, in general, consists of many interconnected subsystems. Although the internal interconnections of the subsystems are known in some applications, in general, they are unknown. Infact, in some cases it is difficult or impossible to identify the subsystems and their interconnections.

Therefore, in developing the mathematical description of the open loop system the input-output description is used. The input-output description considers the system as a "black box" with multi- input-output terminals or ports. The only access to the system and its properties and the only way to study the system for

analysis and design is by its input-output ports. Therefore, when using this description, it is not required to know the models of the individual subsystems nor the interconnections between them.

Several input-output ports can be grouped together to form a ***control area*** or a ***control station*** . The selection of a collection of ports to form a control station depends on the individual system to be controlled. In some systems the ports selection is influenced by the physical geographical separation such that ports which are in one geographical area and close to each other in distance are grouped together to form a control station. In other systems the ports selection is made to achieve mathematical simplification. In other cases the control stations are prespecified to the designer due to the physical structure of the system.

Let us consider a plant with N control stations. The i^{th} control station excites the system with q_i inputs, and measures p_i outputs. Figure 1 describes such a plant.

In Figure 1 the plant structure is assumed unknown. The input ports for the i^{th} control station are denoted by $u_1, u_2, \ \dots \ , u_{q_i}$ or by the column vector $U_i = [\, u_1, u_2, \ \dots \ , u_{q_i}\,]^T$. The output ports for the i^{th} control station are denoted by $y_1, y_2, \ \dots \ , y_{p_i}$ or by the column vector $Y_i = [\, y_1, y_2, \ \dots \ , y_{p_i}\,]^T$.

The plant is controlled from the available N control stations, where a controller is built at each control station. The collective effort of all N controllers is directed to achieve a desired global system performance.

Figure 2 illustrates the plant and the N decentralized controllers. The i^{th} controller transfer matrix is denoted by C_i . The inputs and outputs of the i^{th}

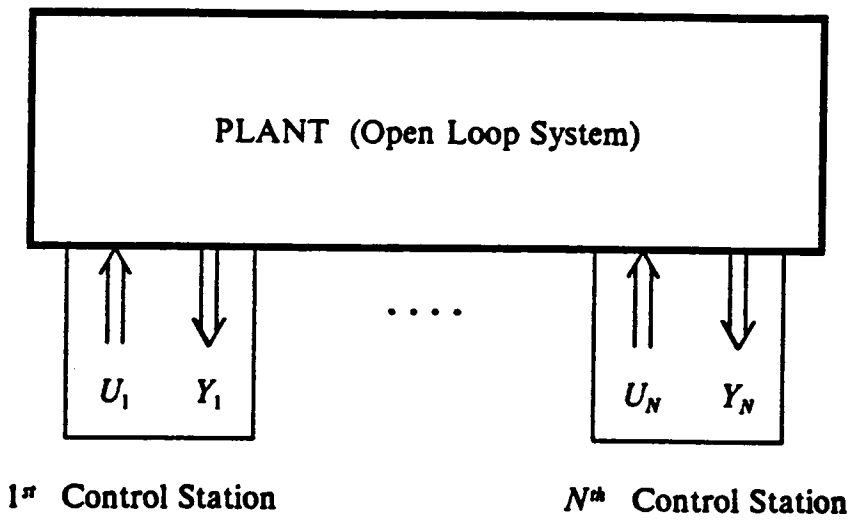


Figure 1. A Plant with N Control Stations.

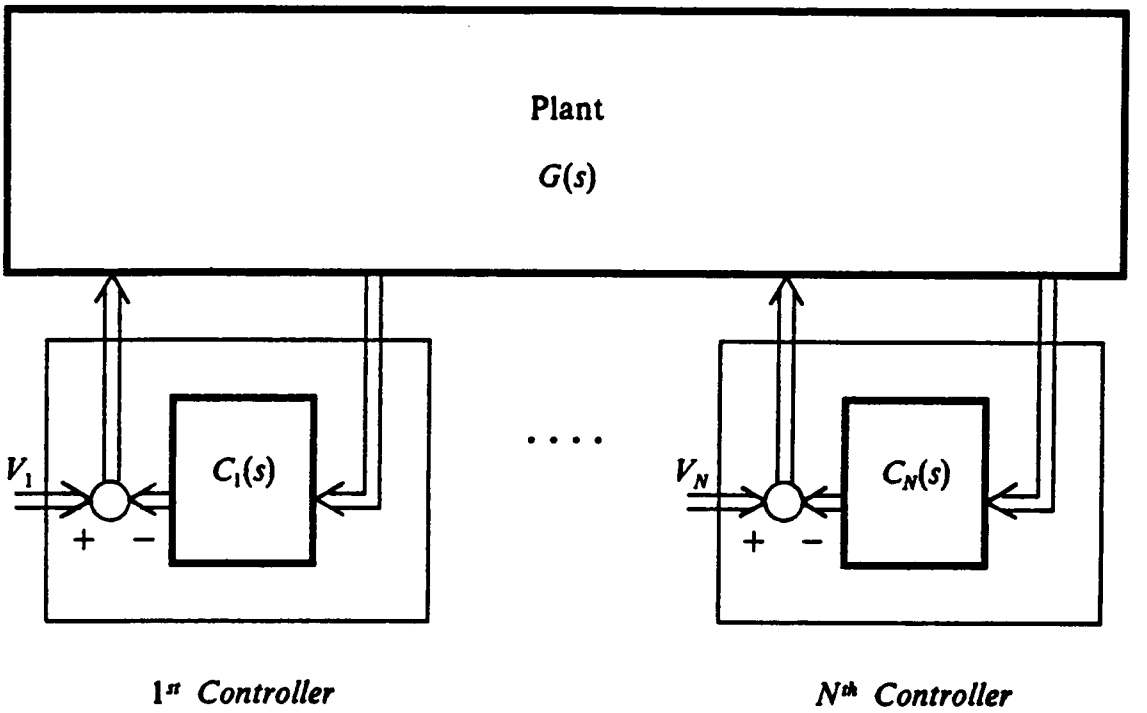


Figure 2. A Plant with Decentralized Controllers.

controller are Y_i and U_i , respectively, which are the output and input of the i^{th} control station. The external reference inputs of the system at the i^{th} control station are collected into the column vector $V_i = [\, v_1, v_2, \ \ldots \ , v_{q_i} \,]^T$.

3.3 Description of the Mathematical Model.

By assumption, the only available information about the system is obtained from the control stations. And it is also assumed that the system is linear, time invariant, and initially relaxed. With these assumptions the best mathematical description which can be used to analyze and control the overall system is the transfer matrix description.

3.3.1 Transfer Matrix of the Overall Feedback System

Let $G(s)$ denote the $p \times q$ transfer matrix of the plant:

$$G(s) = \begin{bmatrix} g_{11}(s) & \ldots & g_{1q}(s) \\ \vdots & & \vdots \\ g_{p1}(s) & \ldots & g_{pq}(s) \end{bmatrix} \tag{3.3.1}$$

where

$$q = \sum_{i=1}^{N} q_i \ , \qquad p = \sum_{i=1}^{N} p_i \ ,$$

and $g_{ij}(s)$, $(i = 1, \ldots, q\ ;\ j = 1, \ldots, p)$ is the transfer function obtained by activating the input u_j and measuring the output y_i. And let $C(s)$ denotes the $q \times p$ transfer matrix of the decentralized controller where:

$$C(s) = \textit{blockdiag}\ [\, C_1(s), \ \ldots\ , C_N(s)\,] \tag{3.3.2}$$

where $C_i(s)$, $(i = 1, \ldots, N)$ is a $q_i \times p_i$ transfer matrix of the i^{th} control station. The controller $C(s)$ is block diagonal due to the restriction of decentralization i.e. the i^{th} controller is using the inputs and outputs of the i^{th} control station. If we are dealing with centralized control, $C(s)$ will be a full matrix.

To derive the overall feedback system transfer matrix, we write the relation between inputs U_j and outputs Y_i, $(i,j = 1, \ldots, N)$:

$$\begin{bmatrix} Y_1(s) \\ \vdots \\ Y_N(s) \end{bmatrix} = \begin{bmatrix} G_{11}(s) & \cdots & G_{1N}(s) \\ \vdots & & \vdots \\ G_{N1}(s) & \ldots & G_{NN}(s) \end{bmatrix} \begin{bmatrix} U_1(s) \\ \vdots \\ U_N(s) \end{bmatrix} \tag{3.3.3}$$

or

$$Y(s) = G(s)\ U(s)$$

And the relation between U_i, Y_i and V_i, $(i = 1, \ldots, N)$ is given by :

$$\begin{bmatrix} U_1(s) \\ \vdots \\ U_N(s) \end{bmatrix} = \begin{bmatrix} V_1(s) \\ \vdots \\ V_N(s) \end{bmatrix} - \begin{bmatrix} C_1(s) & & \\ & \ddots & \\ & & C_N(s) \end{bmatrix} \begin{bmatrix} Y_1(s) \\ \vdots \\ Y_N(s) \end{bmatrix} \tag{3.3.4}$$

or

$$U(s) = V(s) - C(s)\, Y(s)$$

where:

$$Y(s) = [\, Y_1(s), \ \ldots\ , Y_N(s)\,]^T,$$
$$U(s) = [\, U_1(s), \ \ldots\ , U_N(s)\,]^T,$$
$$V(s) = [\, V_1(s), \ \ldots\ , V_N(s)\,]^T.$$

Combining (3.3.3) and (3.3.4), we get:

$$Y(s) = G(s)\, V(s) - G(s)\, C(s)\ Y(s)$$
$$\textit{or} \qquad [\, I_p + G(s)\, C(s)\,]\, Y(s) = G(s)\, V(s) \tag{3.3.5}$$
$$\textit{then} \qquad Y(s) = [I_p + G(s)\, C(s)]^{-1}\ G(s)\, V(s)$$

and using a matrix identity, we obtain another relation:

$$Y(s) = G(s)\, [\, I_q + C(s)\, G(s)]^{-1}\ V(s)$$

The above two relations hold provided that $\det\, [I_p + G(s)\, C(s)\,] \neq 0$ and $\det\, [I_q + C(s)\, G(s)\,] \neq 0$. Without those assumptions the feedback system has no physical meaning, and the matrix equation given in (3.3.5) will not be consistent. In other words for some input $V(s)$, (3.3.5) will not be satisfied for any output $Y(s)$ [6].

The overall transfer matrix of the feedback system is given by one of the following two relations:

$$T(s) = [\, I_p + G(s)\, C(s)\,]^{-1}\ G(s) \tag{3.3.6a}$$

or

$$T(s) = G(s)\ [\ I_q + C(s)\ G(s)\ ]^{-1} \qquad (3.3.6b)$$

Equations (3.3.6a) and (3.3.6b) are identical representations of $T(s)$; either one can be used in the analysis and design of the feedback system.

3.3.2 Matrix Fraction Representation

Let us write the transfer matrix of the plant in the form of the right and left coprime polynomial matrix fractions:

$$G(s) = N_r(s)\ D_r^{-1}(s) \qquad (3.3.7a)$$

and

$$G(s) = D_l^{-1}(s)\ N_l(s) \qquad (3.3.7b)$$

and write the transfer matrix of the controller in the left and right coprime matrix fractions:

$$C(s) = U_l^{-1}(s)\ V_l(s) \qquad (3.3.8a)$$

and

$$C(s) = V_r(s)\ U_r^{-1}(s) \qquad (3.3.8b)$$

where $N_r(s)$ and $D_r(s)$ are $p \times q$ and $q \times q$ right coprime (relatively prime) polynomial matrices, $N_l(s)$ and $D_l(s)$ are $p \times q$ and $p \times p$ left coprime polynomial matrices, $U_l(s)$ and $V_l(s)$ are $q \times q$ and $q \times p$ right coprime block

diagonal polynomial matrices, and $U_r(s)$ and $V_r(s)$ are $p \times p$ and $q \times p$ right coprime block diagonal polynomial matrices.

Substituting (3.3.7a) and (3.3.8a) into (3.3.6b), and (3.3.7b) and (3.3.8b) into (3.3.6a) we get the polynomial matrix representations of $T(s)$:

$$T(s) = N_r(s) \left[U_l(s) D_r(s) + V_l(s) N_r(s) \right]^{-1} U_l(s) \qquad (3.3.9a)$$

or

$$T(s) = U_r(s) \left[D_l(s) U_r(s) + N_l(s) V_r(s) \right]^{-1} N_l(s) \qquad (3.3.9b)$$

Once again the polynomial matrix representations of $T(s)$ given in (3.3.9a) and (3.3.9b) are equivalent, and their use can be interchanged in analysis and design of the feedback system of Figure 2 . The representation in (3.3.9a) is chosen arbitrarily, for the analysis and design throughout the remaining chapters.

We assume that the plant and the controllers are minimum realizations of their transfer matrices i.e. they are free of hidden modes. The above assumption is essential to ensure that the feedback system transfer matrix $T(s)$ completely characterizes the system. The following Theorem [54], provides sufficient and necessary conditions for the composite feedback system to be completely characterized by its transfer matrix $T(s)$.

Theorem 3.1 [54]:

Given two systems S_1 and S_2 which are completely characterized by $G(s)$ and $C(s)$ then the feedback connection of S_1 and S_2 is completely characterized by its transfer matrix $T(s)$ if and only if

$$\delta T(s) = \delta G(s) + \delta C(s) \qquad (3.3.10)$$

where δ is the degree of a proper transfer matrix and it is equal the degree of the characteristic polynomial of the transfer matrix ■ .

The application of the above theorem necessitates that the degree of $T(s)$ be kept equal to the degree of $G(s)$ plus the degree of $C(s)$ so that the feedback system is completely characterized by its transfer matrix $T(s)$.

3.4 Decentralized Pole Placement

The problem of decentralized pole placement is mainly to find a set of N controllers with transfer matrices C_1, ... , C_N , or equivalently a set of N pairs of left coprime polynomial matrices (U_1, V_1), ... , (U_N, V_N) satisfying the following conditions:

1. The poles of the overall closed loop system of Figure 2 are assigned arbitrarily if possible.

2. The overall closed loop system transfer matrix $T(s)$ given in (3.3.9a) is proper.

3. The decentralized controller transfer matrices $C_i(s)$, $(i = 1, \ldots, N)$ are proper.

The above three conditions are discussed in the following subsections in detail.

3.4.1 Solution Existence of the Pole Placement Problem

The existence of a solution of the pole placement problem depends basically on the concept of fixed modes first introduced by Wang and Davison [21]. As discussed in Section 1.2, the fixed modes are those modes of the plant which are not affected by the decentralized feedback. Table 3.1 summarizes the relation between plant fixed modes and closed loop system pole placement.

Therefore, it is essential to have a method to examine the existence of fixed modes of the plant transfer matrix. Below we describe a method to find the fixed modes of the plant transfer matrix in the polynomial matrix fraction form.

Since we will mainly deal with polynomial matrix fractions, we need to develop a method of finding the fixed modes of the system in terms of the polynomial matrix fractions of $G(s)$ and $C(s)$ given in (3.3.7a) and (3.3.8a). To do so, let us write $D(s)$ and $N(s)$ in the following form:

$$D(s) = \begin{bmatrix} D_1^1(s) \\ \vdots \\ D_{q_1}^1(s) \\ D_1^2(s) \\ \vdots \\ D_{q_2}^2(s) \\ \vdots \\ D_1^N(s) \\ \vdots \\ D_{q_N}^N(s) \end{bmatrix}, \qquad N(s) = \begin{bmatrix} N_1^1(s) \\ \vdots \\ N_{p_1}^1(s) \\ N_1^2(s) \\ \vdots \\ N_{p_2}^2(s) \\ \vdots \\ N_1^N(s) \\ \vdots \\ N_{p_N}^N(s) \end{bmatrix}$$

where $D_i^j(s)$ $(i = 1, \ \ldots \ , q_i)$ is the $(q_1 + \ \ldots \ + q_j + i)^{th}$ row of $D(s)$ and $N_i^j(s)$ $(i = 1, \ \ldots \ , p_i)$ is the $(p_1 + \ \ldots \ + p_j + i)^{th}$ row of $N(s)$.

Theorem 3.2

The fixed modes of the decentralized system of Figure 2, are the zeros of the fixed polynomial $\Phi(s)$ defined by:

$$\Phi(s) = g.c.d. \ \Psi_{J_1 \ \ldots \ J_N}(s) \tag{3.4.1}$$

where

$$J_j = \{ D_1^j(s), \ \ldots \ , D_{q_j}^j(s); \ N_1^j(s), \ \ldots \ , N_{p_j}^j(s) \} \tag{3.4.2}$$

and

Table 1. Effect of Fixed Modes on Stability.

Plant	Plant + Decentralized feedback
Has no fixed modes	All poles can be assigned arbitrarily
Has stable fixed modes	Overall system can be stabilized
Has unstable fixed modes	Overall system can not be stabilized

$$\Psi_{J_1 \ldots J_N}(s) = \det \begin{bmatrix} R_1^1(s) \\ \vdots \\ R_{q_1}^1(s) \\ R_1^2(s) \\ \vdots \\ R_{q_2}^2(s) \\ \vdots \\ R_1^N(s) \\ \vdots \\ R_{q_N}^N(s) \end{bmatrix} \tag{3.4.3}$$

where

$$R_i^j(s) \in J_j \qquad i = 1, \ \ldots \ , q_j \ , \qquad j = 1, \ \ldots \ , N$$

and

$$R_i^j \neq R_{i'}^j \qquad i, i' = 1, \ \ldots \ , q_j$$

Proof

The characteristic polynomial of the transfer matrix $T(s)$, using the notation of (3.3.9a) is given by:

$$\det \ (U(s) \ D(s) \ + \ V(s) \ N(s))$$

as stated before. Since the fixed modes will not be affected by any decentralized controller, they will be zeros of the above characteristic polynomial if $U(s)$ and $V(s)$ are taken constant matrices. To save unnecessarily complicated notation, let us ,without loss of generality, prove the case $N = 2$; the case $N > 2$ can be proved in exactly the same way. Let U and V have the following form:

$$U = \begin{bmatrix} u_{11}^1 & \cdots & u_{1q_1}^1 & & \\ \vdots & & \vdots & 0 & \\ u_{q_1 1}^1 & \cdots & u_{q_1 q_1}^1 & & \\ & & & u_{11}^2 & \cdots & u_{1q_2}^2 \\ & 0 & & \vdots & & \vdots \\ & & & u_{q_2 1}^2 & \cdots & u_{q_2 q_2}^2 \end{bmatrix}, \quad V = \begin{bmatrix} v_{11}^1 & \cdots & v_{1p_1}^1 & & \\ \vdots & & \vdots & 0 & \\ v_{q_1 1}^1 & \cdots & v_{q_1 p_1}^1 & & \\ & & & v_{11}^2 & \cdots & v_{1p_2}^2 \\ & 0 & & \vdots & & \vdots \\ & & & v_{q_2 1}^2 & \cdots & v_{q_2 p_2}^2 \end{bmatrix}$$

then $[\, U\ D(s) + V\ N(s)\,] =$

$$\begin{bmatrix} u_{11}^1 D_1^1 + \ldots + u_{1q_1}^1 D_{q_1}^1 + v_{11}^1 N_1^1 + \ldots + v_{1p_1}^1 N_{p_1}^1 \\ u_{q_1 1}^1 D_1^1 + \ldots + u_{q_1 q_1}^1 D_{q_1}^1 + v_{q_1 1}^1 N_1^1 + \ldots + v_{q_1 p_1}^1 N_{p_1}^1 \\ u_{11}^2 D_1^2 + \ldots + u_{1q_2}^2 D_{q_2}^2 + v_{11}^2 N_1^2 + \ldots + v_{1p_2}^2 N_{p_2}^2 \\ \vdots \\ u_{q_2 1}^2 D_1^2 + \ldots + u_{q_2 q_2}^2 D_{q_2}^2 + v_{q_2 1}^2 N_1^2 + \ldots + v_{q_2 p_2}^2 N_{p_2}^2 \end{bmatrix}$$

Expanding the determinant of the above matrix using a linear algebra result discussed in Appendix A, we have :

$$\det\ [\, U\ D(s) + V\ N(s)\,] =$$

$$k_1 \det \begin{bmatrix} D_1^1 \\ D_2^1 \\ \vdots \\ D_{q_1}^1 \\ D_1^2 \\ D_2^2 \\ \vdots \\ D_{q_2}^2 \end{bmatrix} + k_2 \det \begin{bmatrix} D_1^1 \\ D_2^1 \\ \vdots \\ D_{q_1}^1 \\ D_1^2 \\ D_2^2 \\ \vdots \\ N_1^2 \end{bmatrix} + k_3 \det \begin{bmatrix} D_1^1 \\ D_2^1 \\ \vdots \\ D_{q_1}^1 \\ D_1^2 \\ D_2^2 \\ \vdots \\ N_2^2 \end{bmatrix} + \ldots$$

where the $k's$ are constants resulting from multiplying some $u's$ and $v's$. Using the notation used in the statement of the theorem, the above determinant can be written as follows:

$$\det\ [\,U\ D(s)\ +\ V\ N(s)\,]\ =\ \sum_{i=1}^{\mu} k_i\ \Psi_{J_1\ \ldots\ J_N}\ (s)$$

where

$$\mu\ =\ \prod_{i=1}^{N} \binom{q_i}{q_i + p_i} \tag{3.4.4}$$

Hence the fixed polynomial is the *g.c.d.* $\Psi_{J_1\ \ldots\ J_N}(s)$ ■ .

The above theorem enables us to evaluate the fixed modes of the decentralized system. In some cases, it is only required to test for the existence of the fixed modes. This is equivalent to testing the coprimeness of the set of polynomials $\Psi_{J_1\ \ldots\ J_N}\ (s)$; the system is free of fixed modes if and only if that set of polynomials is coprime, or equivalently, the *g.c.d. of* $\Psi_{J_1\ \ldots\ J_N}\ (s)$ is a nonzero real number. For this purpose, it is most convenient to use the method of Sylvester's resultant matrix, introduced in Appendix C, to test for the coprimeness of a set of polynomials.

Example 3.4.1 [61] Consider the two station system of Example 1.2.1. The transfer function matrix $G(s)$ in the right coprime matrix fraction together with the decentralized controller are as follows:

$$G(s) = \frac{1}{(s+1)(s-1)(s-2)} \begin{bmatrix} (s+1)(s-2) & (s+1)(s-1) \\ 0 & (s-1)(s-2) \end{bmatrix} \tag{3.4.5}$$

$$= \begin{bmatrix} s+1 & 1 \\ -(s-2) & 0 \end{bmatrix} \begin{bmatrix} 2(s+1)(s-1) & s-1 \\ -(s+1)(s-2) & 0 \end{bmatrix}^{-1}$$

$$C(s) = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix} \tag{3.4.6}$$

To find the fixed modes of the system (3.4.5) we form J_1 and J_2 of (3.4.2). From the structure of $C(s)$ it is clear that $q_1 = q_2 = p_1 = p_2 = 1$.

$$J_1 = \{D_1^1; N_1^1\} \quad , \quad J_2 = \{D_1^2; N_1^2\}$$

$$\Psi_{J_1, J_2} = \left\{ \begin{array}{l} \det \begin{bmatrix} s+1 & 1 \\ -(s-2) & 0 \end{bmatrix}, \ \det \begin{bmatrix} s+1 & 1 \\ -(s+1)(s-2) & 0 \end{bmatrix}, \\ \det \begin{bmatrix} 2(s+1)(s-1) & s-1 \\ -(s-2) & 0 \end{bmatrix}, \ \det \begin{bmatrix} 2(s+1)(s-1) & s-1 \\ -(s+1)(s-2) & 0 \end{bmatrix} \end{array} \right\}$$

$$= \{ (s-2), (s+1)(s-2), (s-1)(s-2), (s+1)(s-1)(s-2) \}$$

Since the *g.c.d.* of the polynomials of Ψ_{J_1,J_2} is $(s+2)$, we conclude that the system has a fixed mode at $s=2$. Thus the system cannot be stabilized using the controller (3.4.6).

Now suppose that we use a controller of the following structure:

$$C(s) = \begin{bmatrix} 0 & c_2 \\ c_1 & 0 \end{bmatrix} \tag{3.4.7}$$

This is equivalent to applying a controller (3.4.6) to the following transfer function matrix:

$$G(s) = \frac{1}{(s+1)(s-1)(s-2)} \begin{bmatrix} (s+1)(s-1) & (s+1)(s-2) \\ (s-1)(s-2) & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -(s-1) & 1 \\ s-2 & 0 \end{bmatrix} \begin{bmatrix} (s+1)(s-2) & 0 \\ -2(s-1)(s+1) & s-1 \end{bmatrix}^{-1}$$

$$\Psi_{J_1,J_2} = \{ -(s-2), (s+1)(s-1), 0, (s-1)(s+1)(s-2) \}$$

The polynomials of Ψ_{J_1,J_2} has no common divisor. Thus with the controller (3.4.7) the system (3.4.5) has no fixed modes.

Example 3.4.2 [19],[61]: Consider the linear system of Example (1.2.2). The transfer function matrix has the following right coprime matrix fraction description:

$$G(s) = \begin{bmatrix} \frac{1}{s(s-1)} & 0 & 0 \\ \frac{1}{s-1} & 0 & 0 \\ 0 & 0 & \frac{1}{s-1} \\ \frac{1}{s-1} & \frac{1}{s-1} & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ s & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s(s-1) & 0 & 0 \\ -s(s-1) & s-1 & 0 \\ 0 & 0 & s-1 \end{bmatrix}^{-1}$$

And let $q_1 = 1, p_1 = 3, q_2 = 2$, and $p_2 = 1$. Thus the decentralized controller has the following structure:

$$C(s) = \begin{bmatrix} c_1 & c_2 & c_3 & 0 \\ 0 & 0 & 0 & c_4 \\ 0 & 0 & 0 & c_5 \end{bmatrix}$$

Then $J_1 = \{D_1^1 ; N_1^1, N_2^1, N_3^1\}$, $J_2 = \{D_1^2, D_2^2 ; N_1^2\}$

$$\Psi_{J_1, J_2} = \left\{ \begin{bmatrix} D_1^1 \\ D_1^2 \\ D_2^2 \end{bmatrix}, \begin{bmatrix} D_1^1 \\ D_1^2 \\ N_1^2 \end{bmatrix}, \begin{bmatrix} D_1^1 \\ D_2^2 \\ N_1^2 \end{bmatrix} ; \begin{bmatrix} N_1^1 \\ D_1^2 \\ D_2^2 \end{bmatrix}, \begin{bmatrix} N_1^1 \\ D_1^2 \\ N_1^2 \end{bmatrix}, \begin{bmatrix} N_1^1 \\ D_2^2 \\ N_1^2 \end{bmatrix} ; \right.$$
$$\left. \begin{bmatrix} N_2^1 \\ D_1^2 \\ D_2^2 \end{bmatrix}, \begin{bmatrix} N_2^1 \\ D_1^2 \\ N_1^2 \end{bmatrix}, \begin{bmatrix} N_2^1 \\ D_2^2 \\ N_1^2 \end{bmatrix} ; \begin{bmatrix} N_3^1 \\ D_1^2 \\ D_2^2 \end{bmatrix}, \begin{bmatrix} N_3^1 \\ D_1^2 \\ N_1^2 \end{bmatrix}, \begin{bmatrix} N_3^1 \\ D_2^2 \\ N_1^2 \end{bmatrix} \right\}$$

$$= \left\{ \begin{matrix} s(s-1)^3, 0\,, & -s(s-1)^2, (s-1)^2, 0\,, & -(s-1)\,; \\ s(s-1)^2, 0\,, & -s(s-1), 0\,, & -s(s-1), 0 \end{matrix} \right\}$$

Since Ψ_{J_1, J_2} has a *g.c.d.* $(s-1)$, then the system has a fixed mode at $s = 1$.

3.4.2 Realization of Overall System and Controllers

Due to the presence of noise in practical systems, the controller transfer matrices $C_i(s)$, $(i = 1, \ \ldots \ , N)$ and the overall system transfer matrix $T(s)$ are required to be proper rational matrices. Otherwise, noise at the input level will be amplified at the output level due to the improperness of the transfer matrices.

Using the result in Chapter 2, the decentralized controller transfer matrices are proper if and only if the following two conditions are satisfied:

$$\begin{aligned} &\det U_{jh} \neq 0 \\ &\delta_{r_i} U_j(s) \geq \delta_{r_i} V_j(s) \qquad \begin{matrix} i = 1, \ \ldots \ , q \\ j = 1, \ \ldots \ , n \end{matrix} \end{aligned} \tag{3.4.8}$$

where U_{jh} is the row degree matrix of $U_j(s)$ as defined in Section 2.5.

For studying the properness of the closed loop system transfer matrix we will use the same notation as in Section 2.5, in writing the matrices $D(s)$, $N(s)$, $U(s)$ and $V(s)$. Let

$$R(s) = U(s)D(s) + V(s)N(s)$$

$$= H_r(s)\,[\,U_h\,D_h + V_h\,N_h\,]\,H_c(s) + R_l(s) \qquad (3.4.9)$$

$$= H_r(s)\,R_h\,H_c(s) + R_l(s)$$

where

$$R_h = [U_h\,D_h + V_h\,N_h] \qquad (3.4.10)$$

is the leading coefficient matrix of $R(s)$, and $R_l(s)$ has determinant degree less than $(\sum_{i=1}^{q} \delta r_i\; H_r(s) + \sum_{j=1}^{p} \delta c_j\; H_c(s))$.

Theorem 3.3

Let $G(s) = N(s)\,D^{-1}(s)$ and $C(s) = U^{-1}(s)\,V(s)$ be proper transfer matrices of the open loop system and the decentralized controller, respectively. And let $T(s) = N(s)\,R^{-1}(s)\,U(s)$ be the transfer matrix of the closed loop system as defined in (3.3.9a), and let the leading coefficient matrix of $R(s)$,

$$R_h = (U_h\,D_h + V_h\,N_h)$$

be nonsingular, then $T(s)$ is proper if and only if

$$\begin{aligned} \delta_{c_j} H_c(s) &\geq \delta_{c_j} D(s) \\ \delta_{r_i} H_r(s) &\geq \delta_{r_i} U(s) \end{aligned} \tag{3.4.11}$$

Proof

Sufficiency : Suppose that (3.4.11) is satisfied; we want to proof that $T(s)$ is proper. Since G(s) and $C(s)$ are proper,

$$\begin{aligned} \delta_{c_j} N_{(s)} &\leq \delta_{c_j} H_c(s) \\ \delta_{r_i} U(s) &\leq \delta_{r_i} H_r(s) \end{aligned}$$

Now

$$\begin{aligned} T(s) &= N(s)\ R^{-1}(s)\ U(s) \\ &= [N_h\ H_c(s) + N_l(s)]\ [H_r(s)\ R_h\ H_c(s) + R_l(s)]^{-1}\ [H_r(s)\ U_h + U_l(s)] \\ &= [N_h + N_l(s)\, H_c(s)^{-1}]\, [R_h + H_r^{-1}(s)\ R_l(s)\ H_c^{-1}(s)]^{-1}\ [U_h + H_r^{-1}(s)\ U_l(s)] \end{aligned}$$

Taking the limit as $s \to \infty$:

$$\begin{aligned} T(\infty) &= N_h\ R_h^{-1}\ U_h \\ &= N_h\ (U_h\ D_h + V_h\ N_h)^{-1}\ U_h \end{aligned}$$

and since $(U_h\ D_h + V_h\ N_h)$ is nonsingular, the above limit is a constant matrix and $T(s)$ is proper.

Necessity: Suppose that $T(s)$ is proper we want to prove that (3.4.11) is satisfied. By contradiction: If

$$\delta_{c_j} H_c(s) < \delta_{c_j} D(s)$$

then both of D_h and N_h will have a zero column. And if

$$\delta_{r_i} H_r(s) < \delta_{r_i} U(s)$$

then both U_h and V_h will have a zero row. In either case the matrix $(U_h D_h + V_h N_h)$ will be nonsingular and the hypothesis is violated ■ .

From the above Theorem it is worth noting that the maximum degree that det $R(s)$ can take is n, where $n = \delta T(s)$ Any other choice will increase either the column degrees of $H_c(s)$ or the row degrees of $H_r(s)$, which will lead to a non-singular R_h. Moreover, the condition that R_h is not only essential for the properness of the overall system transfer matrix, but also it is necessary for the coprimeness of $U(s)$ and $V(s)$ as illustrated by the following theorem.

Theorem 3.4

The condition that the matrix $(U_h D_h + V_h N_h)$ is nonsingular implies the coprimeness of $U(s)$ and $V(s)$.

Proof

By contradiction. Suppose that $U(s)$ and $V(s)$ are not right coprime, then there exists a block diagonal polynomial matrix $E(s)$ such that

$$U(s) = E(s)\ \overline{U}(s)$$
$$V(s) = E(s)\ \overline{V}(s)$$

where $\overline{U}(s)$ and $\overline{V}(s)$ are coprime polynomial matrices, and $E(s)$ is not a unimodular polynomial matrix. And since

$$C(s) = U^{-1}(s)\ V(s) = \overline{U}^{-1}(s)\ \overline{V}(s)$$

then

$$\begin{aligned} \delta\, C(s) &= \delta\ \det\ U(s) \\ &= \delta\ \det\ \overline{U}(s) + \delta\ \det\ E(s) \end{aligned}$$

which implies that

$$\delta\ \det\ \overline{U}(s) < \delta\ \det\ U(s)$$

but

$$\begin{aligned} \delta\, T(s) &= \delta\, C(s) + \delta\, G(s) \\ &= \delta\ \det\ \overline{U}(s) + \delta\ \det\ D(s) \\ &\neq \delta\ \det\ U(s) + \delta\ \det\ D(s) \end{aligned}$$

This implies that the condition $\det\ (U_h\ D_h + V_h\ N_h) \neq 0$ is violated, because this condition means that

$$\begin{aligned} \delta\, T(s) &= \sum_{i=1}^{p} H_r(s) + \sum_{j=1}^{q} H_c(s) \\ &= \delta\ \det\ U(s) + \delta\ \det\ D(s) \end{aligned}$$ ■.

In summary, for properness of the controllers and overall transfer matrices the following conditions are required:

$$
\begin{aligned}
&\det\ U_{jh} \neq 0 \\
&\delta_{r_i}\, U_j(s) = \delta_{r_i}\, V_j(s) \qquad i = 1, \ \ldots\ , q \\
&\qquad\qquad\qquad\qquad\quad j = 1, \ \ldots\ , N \\
&\det\ (U_h\ D_h + V_h\ N_h) \neq 0
\end{aligned}
\tag{3.4.12}
$$

3.4.3 Stability of the Closed Loop Feedback System

In Chapter 2 the concept of the characteristic polynomial and the degree of a proper transfer matrix were introduced. In this section we shall use these concepts for the closed loop transfer matrix $T(s)$. Since for properness of $T(s)$ we require that $(U_h\ D_h + V_h\ N_h)$ be nonsingular, then

$$\delta\ \det\ R(s) = \delta\ G(s) + \delta\ C(s)$$

and for $T(s)$ to completely characterize the feedback system we require that

$$\delta\ T(s) = \delta\ G(s) + \delta\ C(s)$$

hence

$$\delta\ T(s) = \delta\ \det\ R(s) \tag{3.4.13}$$

Moreover, since

$$T(s) = N(s)\ R^{-1}(s)\ U(s)$$

then

$$\textit{Char. Poly. of } T(s) = \det R(s) \tag{3.4.14}$$

Accordingly, the feedback system is asymptotically stable and BIBO stable if and only if the roots of the polynomial $\det [U_l(s)\ D_r(s) + V_l(s)\ N_r(s)]$ have negative real parts.

From the above we conclude that for the pole assignment problem the overall system stability is satisfied if and only if the roots of

$$\det R(s) = \det [U_l(s)\ D_r(s) + V_l(s)\ N_r(s)] \tag{3.4.15}$$

are assigned to have negative real parts.

3.5 Solution of the Decentralized Pole Placement

Without loss of generality we can assume that the decentralized feedback system is free of fixed modes. The case when they exist is discussed later in this chapter. The method of solution is based on mathematical programming, where the problem of pole placement is set up as an optimization problem. The roots of the overall system characteristic polynomial are considered as complex variables. The distance norm between the variable roots and the desired roots is minimized subject to certain constraints. The constraints include decentralization constraints i.e. the controller matrix fraction $U(s)$ and $V(s)$ are block diagonal polynomial matrices, stability constraints i.e. determinant of $R(s)$ must equal the

variable roots of the overall system characteristic polynomial, and properness constraints of the controllers and the overall feedback system transfer matrices. Furthermore, other type of constraints may be included, constraints due to physical and practical requirements possed by the system considered. In other cases additional design constraints are required which are essential to that specific control problem.

The problem of pole assignment is to solve for the decentralized controllers such that (3.4.5) is satisfied and the polynomial in (3.4.12) has roots with negative real parts. In other words, it is required to solve the equation

$$\det\,[U(s)\,D(s) + V(s)\,N(s)] = \bar{x}_0\,(s - \bar{s}_1)\,\ldots\,(s - \bar{s}_n) \qquad (3.5.1)$$

for every s, where $\bar{s}_1, \ldots, \bar{s}_n$ are the desired poles, which are complex in general, $\bar{x}_0$ is a real number, and n is the order of the system i.e. $n = \delta T(s)$. In the following development we will replace the constant desired poles $\bar{s}_1, \ldots, \bar{s}_n$ by the variable poles $s_1, \ldots, s_n$. Accordingly (3.5.1) will be replaced by

$$\det\,[U(s)\,D(s) + V(s)\,N(s)] = x_0\,(s - s_1)\,\ldots\,(s - s_n) \qquad (3.5.2)$$

The reason for this replacement is to allow relative pole assignment. This is fulfilled by minimizing the difference between s_i and $\bar{s}_i$, $i = 1, \ldots, n$ and satisfying (3.5.2). Solving equation (3.5.1) will give either a solution to the desired pole placement problem or no solution at all. But by using equation (3.5.2) we are not only finding the solution of the desired pole placement but also investi-

gating any other solutions with poles different than the desired ones and finding how far those resulting poles are from the desired ones.

3.5.1 Modeling of the Objective Function

As mentioned earlier, the objective function to be minimized is a certain norm. For convenience the norm is taken to be the Euclidean distance norm. Now suppose that the roots of the characteristic polynomial, which are the poles of the overall system are of the following form:

$$\begin{aligned} s_i &= x_i && \textit{real pole} \\ s_i &= (x_i + j y_i), \\ s_{i+1} &= (x_{i+1} - j y_i) && \textit{complex pair} \end{aligned} \tag{3.5.3}$$

$$\textit{with} \quad y_i (x_i - x_{i+1}) = 0 \tag{3.5.4}$$

where x_i, x_{i+1} and y_i are variable real numbers.

The reason for assuming two different real parts in the case of the complex poles form is to give the optimization algorithm the freedom to choose the poles as a complex pair or as two real poles. The condition (3.5.4) insures that the coefficients of the polynomial formed by the poles (3.5.3) are real. In other words, only two real poles or a pair of conjugate complex poles are allowed.

Let the desired poles, which are considered constants, be denoted by:

$$\begin{aligned} \bar{s}_i &= \bar{x}_i && \textit{real pole} \\ \bar{s}_i &= (\bar{x}_{i+1} + j\bar{y}_i), && \textit{complex pair} \\ \bar{s}_{i+1} &= (\bar{x}_i - j\bar{y}_i) \end{aligned} \tag{3.5.5}$$

where $\bar{x}_i$, $\bar{x}_{i+1}$, and $\bar{y}_i$ are constant real numbers.

For the purpose of clarity let us consider the case $n = 3$, where we have one real pole and a pair of complex poles. The general case is treated in a similar fashion except that the number of poles is greater. Let the three constant poles to be assigned be: $\bar{s}_1 = \bar{x}_1$, $\bar{s}_2 = (\bar{x}_2 + j\bar{y}_2)$, and $\bar{s}_3 = (\bar{x}_2 - j\bar{y}_2)$, And the three corresponding variable poles be: $s_1 = x_1$, $s_2 = (x_2 + jy_2)$, and $s_3 = (x_3 + jy_2)$ with $y_2(x_2 - x_3) = 0$. For the real root x_1 the squared distance to be minimized is $d_1^2 = (x_1 - \bar{x}_1)^2$, as shown in Figure 4. For the pair of complex poles, the squared distances to be minimized are: $d_2^2 = (x_2 - \bar{x}_2)^2$, $d_3^2 = (x_3 - \bar{x}_2)^2$, $d_4^2 = (y_2 - \bar{y}_2)^2$, as shown in Figure 4. The objective function for the above three poles is :

$$\textit{Minimize} \quad (x_1 - \bar{x}_1)^2 + (x_2 - \bar{x}_2)^2 + (x_3 - \bar{x}_2)^2 + (y_2 - \bar{y}_2)^2$$

$$\textit{Subject to} \quad (x_2 - x_3)y_2 = 0$$

3.5.2 Constraints Model

Beside the $x's$ and the $y's$ defined above, the only variables left are the parameters of the controllers transfer matrices C_i, $i = 1, \ \ldots \ , N$, or specifically the coefficients of the polynomials of the polynomial matrices $U_i(s)$ and $V_i(s)$, $i = 1, \ \ldots \ , N$. The number of coefficients, from now on called the controllers variables denoted by the vector z, depends on the order of the controllers. To satisfy the second properness condition given in (3.4.12) and at the same time to maximize the number of controllers variables without increasing the order of the controllers, the row degrees of $U_i(s)$ are taken equal to the row degrees of $V_i(s)$, $i = 1, \ \ldots \ , N$.

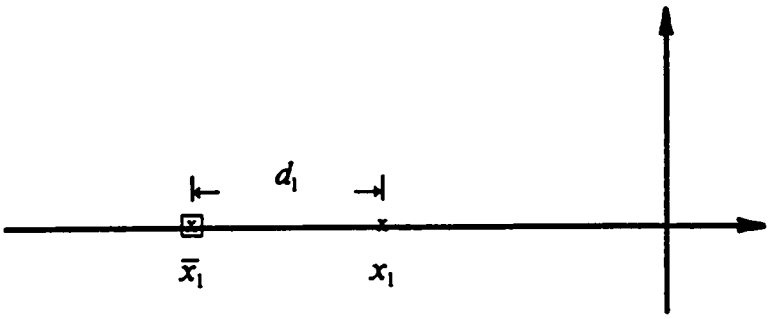
Example 3.5.1

Let $N = 2$, $p_1 = q_1 = 2$, $P_2 = 2$, and $q_2 = 1$. And suppose that the order of the controller is 2 with the following row degrees: $\delta_{r1} U_1(s) = 1$, $\delta_{r2} U_1(s) = 0$ and $\delta_{r2} U_2(s) = 1$, then

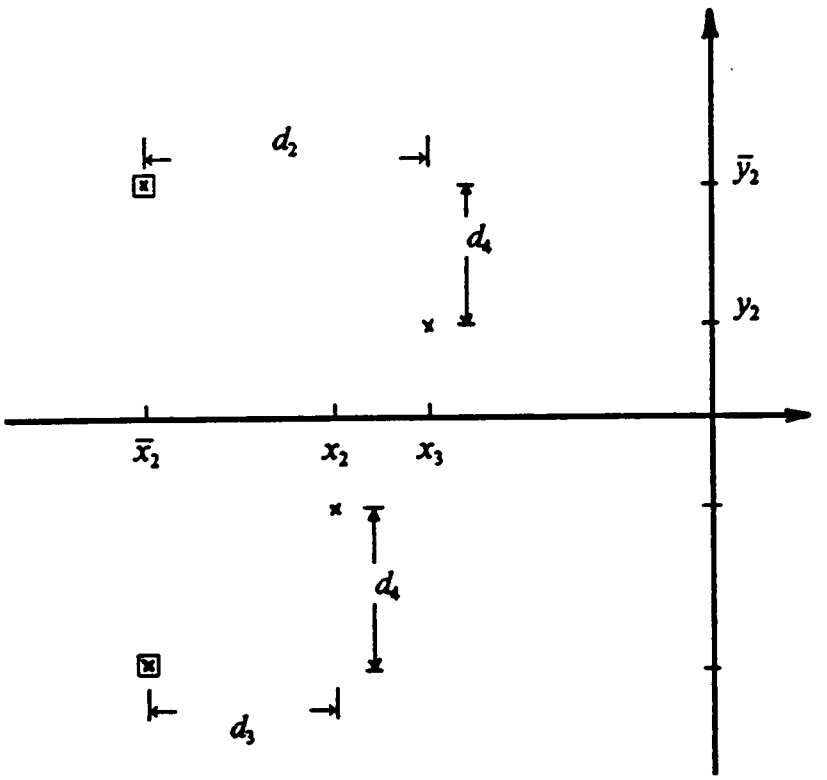
$$U(s) = \begin{bmatrix} z_1 s + z_2 & z_3 s + z_4 & 0 \\ z_5 & z_6 & 0 \\ 0 & 0 & z_7 s + z_8 \end{bmatrix}$$

$$V(s) = \begin{bmatrix} z_9 s + z_{10} & z_{11} s + z_{12} & 0 & 0 \\ z_{13} & z_{14} & 0 & 0 \\ 0 & 0 & z_{15} s + z_{16} & z_{17} s + z_{18} \end{bmatrix}$$

and the controllers variable vector is: $z = [\, z_1, \ \ldots \ , z_{18} \,]$ ■ .



Distance of a real pole



Distances of complex poles

Figure 3. Distances to be minimized for real and complex poles.

For pole assignment, (3.5.2) must be satisfied for all complex numbers s, therefore it cannot be included in that form as a constraint. Instead, the constraints are formed by equating coefficients of like powers of (3.5.2). To do that we define the following functions:

$$\det R(s) = f_n(z)\, s^n + f_{n-1}(z)\, s^{n-1} + \ \ldots \ + f_0(z) \tag{3.5.6}$$

and

$$x_0 (s - s_1) \ \ldots \ (s - s_n) = g_n(x,y)\, s^n + \ \ldots \ + g_0(x,y) \tag{3.5.7}$$

then the constraints which gives the pole assignment are:

$$\begin{aligned} f_i(z) &= g_i(x,y) && i = 1, \ \ldots \ , n \\ (x_j - x_{j+1})\, y_j &= 0 && j = 1, \ \ldots \ , m \end{aligned}$$

where z is the controllers variable vector, $x = [x_1, \ \ldots \ , x_n]^T$, $y = [y_1, \ \ldots \ , y_n]^T$, and m is equal to the number of pairs of complex poles.

3.5.3 Modeling the Stability and Properness Constraints

With the layout of the poles of the overall system given in (3.5.1) the stability constraints are simply:

$$x_i < 0 \qquad i = 1, \ \ldots \ , n \tag{3.5.8}$$

For properness of the overall system transfer matrix the constraint is given by Theorem 3.3:

$$\det \ (U_h \ D_h + V_h \ N_h) \neq 0$$

while the properness of the controllers transfer matrices is given by (3.4.8) :

$$\det \ U_{ih} \neq 0 \qquad i = 1, \ \ldots \ ,N$$

The following example shows how the overall mathematical optimization is put together:

Example 3.5.2 :

Consider a $2x2$ system , where it is required to investigate controlling it from two SISO control stations using static feedback. Let

$$G(s) = \begin{bmatrix} \dfrac{5\,(s+0.2)}{(s+2)\,(s-1.5)} & \dfrac{s^2+2.5s+2.5}{(s+2)\,(s-1.5)} \\ \dfrac{7s-10}{(s+2)\,(s-1.5)} & \dfrac{s^2-0.5s-1}{(s+2)\,(s-1.5)} \end{bmatrix}$$

and

$$C(s) = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix}$$

which can be written in the following right and left coprime matrix fraction:

$$G(s) = N(s)\, D^{-1}(s) = \begin{bmatrix} s+3 & s+1 \\ s+2 & s-2 \end{bmatrix} \begin{bmatrix} s+1 & -0.5 \\ s-4 & s-1 \end{bmatrix}^{-1}$$

and

$$C(s) = U^{-1}\, V = \begin{bmatrix} u_1 & 0 \\ 0 & u_2 \end{bmatrix}^{-1} \begin{bmatrix} v_1 & 0 \\ 0 & v_2 \end{bmatrix}$$

then $(U\, D(s) + V\, N(s)) =$

$$\begin{bmatrix} u_1 (s+1) + v_1 (s+3) & u_1 (-0.5) + v_1 (s+1) \\ u_2 (s-4) + v_2 (s+2) & u_2 (s-1) + v_2 (s-2) \end{bmatrix}$$

Evaluating the determinant of the above matrix we get:

$$\begin{aligned} (U\, D(s) + V\, N(s)) = {} & u_1 u_2 (s^2 + 0.5s - 3) + v_1 u_2 (5s + 1) \\ & + u_1 v_2 (s^2 - 0.5s - 1) + v_1 v_2 (-2s - 8) \end{aligned} \tag{3.5.9}$$

Moreover, $D(s)$ and $N(s)$ can be written in the following form:

$$D(s) = D_h\, H_c(s) + D_l(s) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} + \begin{bmatrix} 1 & -.5 \\ -4 & -1 \end{bmatrix}$$

and

$$N(s) = N_h\, H_c(s) + N_l(s) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} + \begin{bmatrix} 3 & 1 \\ 2 & -2 \end{bmatrix}$$

then

$$\det\,(U_h\, D_h + V_h\, N_h) = u_1\,(u_2 + v_2)$$

Now suppose that the desired characteristic equation is picked arbitrarily, to be $(s^2 + 1.5\,s + 0.5)$ i.e. the desired poles are $\bar{s}_1 = -0.5$ and $\bar{s}_2 = -1.0$. And suppose that the resulting variable poles are complex i.e. $s_1 = -(x_1 + jy_1)$ and $s_2 = -(x_2 - jy_1)$, therefore, the characteristic equation of the resulting complex poles is:

$$s^2 + (x_1 + x_2)s + (x_1\,x_2 + y_1^2) \qquad (3.5.10)$$

Equating coefficients of like powers of (3.5.9) and (3.5.10) we get the three nonlinear equations. Together with the realization and stability constraints, the minimization model is :

[continue next page]

minimize $[(x_1 - 0.5)^2 + (x_2 - 1)^2 + (y_1 - 0)^2]$

subject to

$$u_1 (u_2 + v_2) = 1$$

$$0.5\, u_1 (u_2 - v_2) + v_1 (5\, u_2 - 2\, v_2) = x_1 + x_2$$

$$- u_1 (3\, u_2 + v_2) + v_1 (u_2 - 8\, v_2) = x_1 x_2 + y_1^2$$

$$y_1 (x_1 - x_2) = 0$$

$$u_1 u_2 \neq 0$$

$$x_1 > 0$$

$$x_2 > 0 \qquad \blacksquare.$$

In the above example the determinants involved are 2×2, which are simple to expand. As a result, the functions in eqs (3.5.6) and (3.5.7) can easily be found explicitly. However, practical problems show high dimensional matrices and high system orders. Accordingly, for that type of systems the f_i and g_i, $i = 1, \ \ldots \ , n$ functions need not be known explicitly; rather they can be evaluated implicitly. At each iteration of the optimization algorithm $\det R(s,z)$ is evaluated as a function of the controller vector z. From that, the values of the functions $f_i(z)$, $i = 1, \ \ldots \ , n$ are obtained. Similarly, substituting the vectors x and y into $[x_0 (s - s_1) \ \ldots \ (s - s_n)]$, the values of the functions $g_i(x,y)$, $i = 1, \ \ldots \ , n$ are obtained.

3.5.4 Computational Complexity

As noted from the previous example, the design technique in this research comprises two stages: formulating the problem using polynomial matrices and solving the a nonlinear programming problem.

The complexity of the optimization problem depends on the method used for solving the constrained nonlinear program, as well as on the number of variables and constraints imposed on the problem. In our case, for a closed loop system of order n, the number of constrains is equal to $(2n + 1)$ plus any additional design constraints. The number of variables can be expressed as:

$$n + \sum_{k=1}^{N} (p_k + q_k) \left[\sum_{i=1}^{q_k} (\delta_{ri}\, U_k(s) + 1) \right] \tag{3.5.11}$$

where δ_{ri} is the i^{th} row degree and $U_k(s)$ is a $q_k x q_k$ block of the block diagonal matrix $U(s)$ of (3.3.8).

Starting with the transfer matrix function description $G(s)$ of the plant, the design procedure to assign the closed loop poles involves the following steps:

Step 1 Obtain a right coprime matrix fraction of $G(s) = N(s)\, D(s)^{-1}$ using algorithms of Appendix D. This step is executed once and involves lower triangularization of a polynomial matrix. Therefore, it involves $O(q^3\, d^2)$ computations, where q is the dimension of $D(s)$ and d is the maximum degree of its elements.

Step 2 For a given controller order, set the values of the left coprime matrix fractions $U(s)$ and $V(s)$ of $C(s)$, as defined in (3.3.8a). This step does not involve any computations.

Step 3 Form the polynomial matrix $R(s) = [U(s)D(s) + V(s)N(s)]$ whose determinant is the closed loop characteristic polynomial and form $g(s) = (s - s_1)(s - s_2) \ldots (s - s_n)$ as described in (3.5.7). This step involves $O(q^3 r^2)$ computations, where q is the dimension of $R(s)$ and r is the maximum degree of its elements.

Step 4 Minimize a certain objective function, such that the difference between the roots of det $R(s)$ and the desired poles is minimized. No polynomial-time algorithm is known for finding the optimal solution to this generally nonconvex programming problem.

3.6 Feedback and Feedforward Decentralized Control.

The decentralized controller considered at the very beginning of this chapter consists of pure feedback components. In this section we will discuss controllers which have feedback and feedforward components as illustrated by Figure 4. To derive the transfer matrix for this system we will use the same notation for $G(s)$ and $C(s)$ in equations (3.2.3) and (3.2.4). Denoting the qxq transfer matrix of the feedforward decentralized controller by $H(s)$

$$H(s) = blockdiag\ [\, H_1(s),\ \ldots\ , H_N(s)\,] \tag{3.6.1}$$

where $H_i(s)$, $(i = 1, \ldots, N)$ is a $q_i \times q_i$ transfer matrix of the i^{th} feedforward controller. Repeating the same steps by which we obtain the transfer matrix in (3.2.6), on the system of Figure 4, we obtain the following relation between the external inputs and outputs of that system:

$$Y(s) = G(s)\,H(s)\,[I_q + C(s)\,G(s)\,H(s)\,]^{-1}\ V(s) \tag{3.6.2}$$

writing the transfer matrices in the following matrix fraction forms:

$$G(s) = N(s)\,D^{-1}(s)$$
$$C(s) = U^{-1}(s)\,V(s)$$
$$H(s) = P(s)\,Q^{-1}(s)$$

(3.6.2) yields the following overall system transfer matrix:

$$T(s) = Q(s)\,[U(s)\,Q(s) + V(s)\,N(s)\,D^{-1}(s)\,P(s)\,]^{-1}\,U(s) \tag{3.6.3}$$

Now let us introduce two qxq polynomial matrices $\overline{P}(s)$ and $\overline{D}(s)$ such that:

$$D^{-1}(s)\,P(s) = \overline{P}(s)\,\overline{D}^{-1}(s) \tag{3.6.4}$$
$$P(s)\,\overline{D}(s) = D(s)\,\overline{P}(s)$$

Accordingly the characteristic equation of the overall system can be written as

$$T(s) = Q(s)\,\overline{D}(s)\,[U(s)\,Q(s)\,\overline{D}(s) + V(s)\,N(s)\,\overline{P}(s)\,]^{-1}\,U(s) \tag{3.6.5}$$

The minimization model for the arbitrary pole assignment is constructed in the same manner as in Section 3.4. The only difference is the addition of more constraints and variables due to (3.6.4). More specifically, the constraints and variables can be defined as follows:

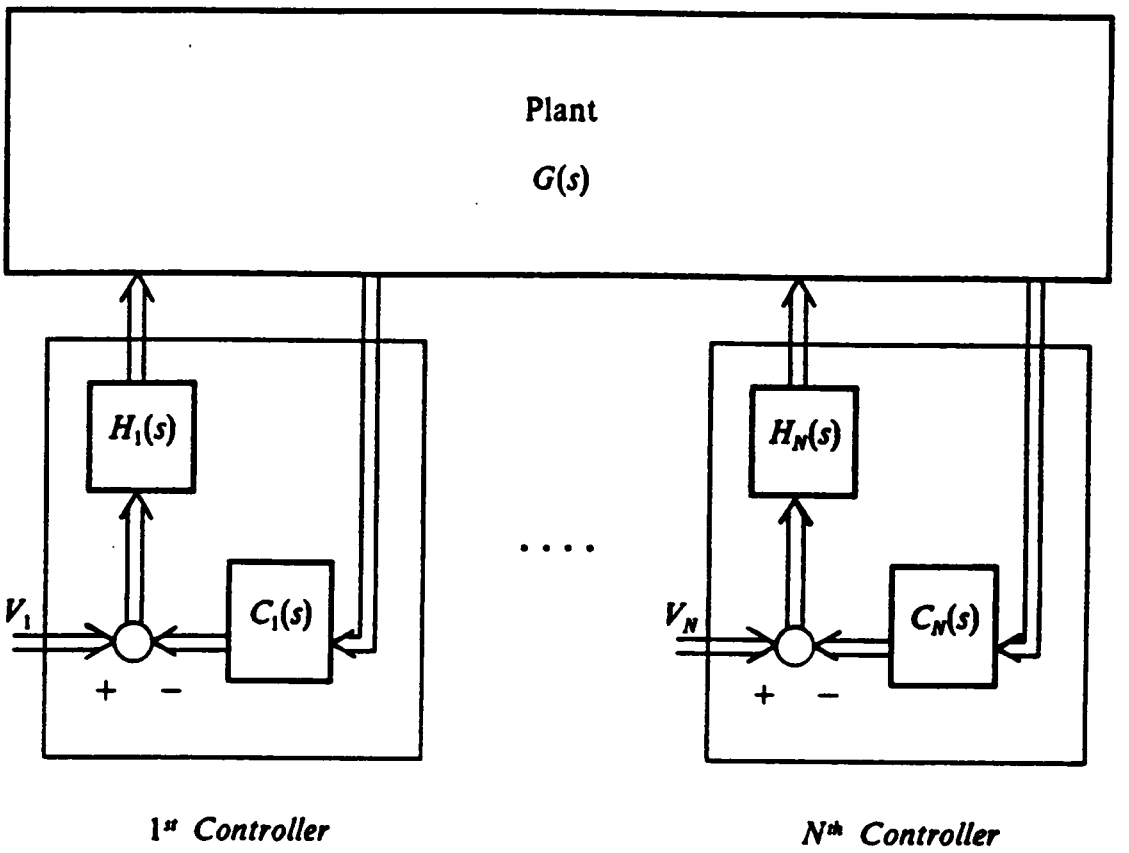


Figure 4. A Plant with Feedback and Feedforward Controllers.

$$f_i(z, \bar{z}, \bar{\bar{z}}) = g_i(x) \qquad i = 0,1, \ \ldots \ , n \tag{3.6.6}$$

$$w_j(\bar{z}, \bar{\bar{z}}) > 0 \qquad j = 1, \ \ldots \ , \bar{n} \tag{3.6.7}$$

where $z = [u_1, u_2, \ \ldots \ ; v_1, v_2, \ \ldots \ ]$ is the vector of coefficients of $U(s)$ and $V(s)$, $\bar{z} = [\bar{d}_1, \bar{d}_2, \ \ldots \ ; \bar{p}_1, \bar{p}_2, \ \ldots \ ]$ is the vector of coefficients of $\bar{D}(s)$ and $\bar{P}(s)$, $\bar{\bar{z}} = [p_1, p_2, \ \ldots \ ; q_1, q_2, \ \ldots \ ]$ and $w_j(\bar{z})$, $(j = 1,2, \ \ldots \ , \bar{n})$ are the coefficients of the polynomial matrix $[P(s)\,\bar{D}(s) - D(s)\,\bar{P}(s)]$ and $\bar{n} = \sum_{i=1}^{q} \sum_{j=1}^{q} (\delta_{ej} D(s) + \delta_{ri} P(s) + 1)$.

The number of constraints and the number of variables in (3.6.7) are large. Therefore, for computational purpose, it is recommended to reduce both the number of constraints and the number of variables before solving the minimization problem. A careful look at the constraints in (3.6.7), one can notice that they are linear in $\bar{z}$. Simple computations reveal that the number of variables in $\bar{z}$ is more than $\bar{n}$ by q^2 . For such a linearly constrained programs one can solve for some of the variables in terms of others [42,43], so that the constraints (3.6.7) are eliminated and the number of variables is reduced.

Introducing the feedforward components in the controllers has increased the freedom in the domain of the solution. This is due to the increase of the number of controller parameters. The disadvantage, of course, is in the complexity of the physical controller, as well as in the mathematical minimization model. More computational burden is added to the minimization problem due to the addition

of the new constraints in (3.6.4). Nevertheless, this is justifiable if one is looking for reducing the controller's order.

3.7 Disturbance Rejection

In this section a detailed analysis of the disturbance rejection problem in the feedback system of Figure 2 is presented. The problem of disturbance rejection involves compensating a given plant so that in the resulting system the transfer matrix which relates the disturbance and the plant output is zero. Thus, the output of the plant is influenced by the reference input and initial condition only. It will be assumed that the disturbance is measurable and accessible to the compensator. The compensator is assumed to have the decentralized structure described in (3.3.2).

Consider the input-output external model of the feedback system in Figure 2:

$$\begin{aligned} y(s) &= G(s)\,u(s) + F(s)\,d(s) \\ u(s) &= C(s)\,y(s) - H(s)\,d(s) + V(s) \end{aligned} \tag{3.7.1}$$

where $d(s)$ is an $rx1$ disturbance vector, $F(s)$ and $H(s)$ are pxr and qxr transfer function matrices, respectively. After some algebraic manipulation we obtain

$$
\begin{aligned}
y(s) = \; & (I_p + G(s)C(s))^{-1} G(s)\, V(s) \\
& + (I_p + G(s)\, C(s))^{-1} (F(s) - G(s)\, H(s))\, d(s)
\end{aligned}
\tag{3.7.2}
$$

Thus, the transfer function matrix due to the disturbance input, denoted by $T_{y/d}$, is

$$
T_{y/d} = (I_p + G(s)\, C(s))^{-1} (F(s) - G(s)\, H(s)) \tag{3.7.3}
$$

Write the transfer matrices $G(s)$ and $F(s)$ in the form of the left coprime matrix fractions; and $C(s)$ and $H(s)$ in the form of right coprime matrix fractions

$$
\begin{aligned}
G(s) &= D(s)^{-1} N(s), \qquad F(s) = D(s)^{-1} M(s) \\
G(s) &= V(s)\, U(s)^{-1}, \qquad H(s) = W(s)\, U(s)^{-1}
\end{aligned}
\tag{3.7.4}
$$

where $D(s), N(s), M(s), V(s), W(s)$ and $U(s)$ are polynomial matrices with appropriate dimensions. Substituting (3.7.4) into (3.7.3) we obtain

$$
T_{y/d} = U(s)\, (D(s)\, U(s) + N(s)V(s))^{-1} (M(s)\, U(s) - N(s)W(s))\, U^{-1}(s) \tag{3.7.5}
$$

The decentralized disturbance rejection problem is reduced to finding a block diagonal matrix $U(s)$ such that

$$
M(s)\, U(s) = N(s)\, W(s) \tag{3.7.6}
$$

which can be included as a constraint on the optimization problem of Section 3.5.

3.8 Controller Order Minimization

For system order minimization we start with the least possible order the controllers can take, which is in most cases a zero order, i.e. static feedback. The optimization problem is then solved with the zero order controller. The solution is one of the following three possibilities:

Case I Optimal Solution, with objective function minimum value equal to zero. In this case the solution is exact pole placement, and the decentralized zero order controllers can assign the desired poles.

Case II Optimal Solution, with objective function minimum value not equal to zero. In this case a solution exists but the zero order controllers cannot assign the desired poles exactly. Instead the optimization algorithm provides alternative, closest poles to the desired ones. It is up to the designer either to accept the resulting poles with zero order controller or to increase the order of one or more local controllers, then solve the optimization problem again.

Case III Infeasible Solution. This case occurs either due to the existence of unstable fixed modes, which have to be removed, or there are too many constraints for the number of variables. For the former situation the test for fixed modes is given earlier in this chapter (Section 3.3). For the latter situation the order of one or more controllers

has to be increased and the optimization problem has to be solved again.

It is worth noting that increasing the order of the controllers will increase the order of the overall system by the same amount. This is a result of Theorem 3.1.

It remains to be shown that for systems free of fixed modes, or those whose fixed modes are isolated, increasing the controller order will solve the pole assignment problem, and also, to give an upper bound on the order of the decentralized controller. To do so, it is necessary to introduce some useful notations. Let the elements of the k^{th} rows of $U(s)$ and $V(s)$ defined in (3.3.8a) be functions of s and the remaining rows have scalar real mumbers elements. Let the non zero elements of the k^{th} row of $U(s)$ and $V(s)$ be denoted by

$$\{u_{k,j+1}(s), u_{k,j+2}(s), \ \dots \ , u_{k,j+q_\mu}(s); v_{k,l+1}(s), v_{k,l+2}(s), \ \dots \ , v_{k,l+p_\mu}(s)\} \quad (3.8.1)$$

where $j = \sum_{i=1}^{k-1} q_i$, $\mu \in \{1,2, \ \dots \ ,N\}$ and $l = \sum_{i=1}^{k-1} p_i$. Let the elements of the k^{th} row of $R(s) = [U(s)\, D(s) + V(s)\, N(s)]$ be denoted by

$$\{r_{k1}(s), r_{k2}(s), \ \dots \ , r_{kq}(s)\} \quad (3.8.2)$$

where $q = \sum_{i=1}^{N} q_i$ and $r_{ki}(s) = \sum_{m=j+1}^{j+q_\mu} u_{km}(s)\, d_{mi}(s) + \sum_{m=l+1}^{l+p_\mu} v_{km}(s)\, n_{mi}(s)$. Now expanding det $R(s)$ using the elements of the k^{th} row, we obtain

$$\det R(s) = r_{k1}(s)\, m_{k1}(s) + \ \dots \ + r_{kq}(s)\, m_{kq}(s) \quad (3.8.3)$$

where $m_{ki}(s)$ is the minor of $R(s)$ obtained from deleting the k^{th} row and the i^{th} column. These minors are functions of elements of $D(s)$, $N(s)$ as well as, the constant controller parameters. For the pole placement problem, let the desired closed loop characteristic polynomial be $p(s)$ then it is required that

$$r_{k1}(s)\, m_{k1}(s) + \ \ldots \ + r_{kq}(s)\, m_{kq}(s) = p(s) \tag{3.8.4}$$

Using results of linear diophantine equations for polynomials [3], equation (3.8.4) is regarded as a linear diophantine equation in the variable polynomials $r_{ki}(s)$, $i = 1, \ \ldots \ , q$. Accordingly, equation (3.8.4) has a solution if and only if the greatest common divisor of $m_{ki}(s)$, $i = 1, \ \ldots \ , q$ divides $P(s)$. With this notation we introduce the following result.

Theorem 3.5

Let the closed loop system of Figure 2 be free of fixed modes. And let the $m_{ki}(s)$, $i = 1, \ \ldots \ ,q$ be coprime polynomials. Then an upper bound of the order of the decentralized controller, denoted by $O_{u.b.}$, which arbitrarily assigns the closed loop poles is given by:

$$O_{u.b.} = \min_{\mu} \left(\left\lceil \frac{deg \ \det \ D(s)}{q_\mu + p_\mu - 1} \right\rceil + 1 \right) \tag{3.8.5}$$

where q_μ and p_μ , $\mu \in \{ 1, \ \ldots \ , N \}$ are the dimensions of the N control stations defined in Section 3.1. The ceiling function $\lceil . \rceil$ produces the least integer greater than or equal to it's argument.

Proof

The upper bound $O_{u.b.}$ will be found by expanding the determinant of $R(s)$ defined in (3.3.2) by the cofactors of the k^{th} row. Thus, a system of linear equations is formed. The number of variables in that system of equations is equal to the number of control parameters in the k^{th} row. The $O_{u.b.}$ is the number of control parameters of the k^{th} row which makes the linear system of equations solvable, i.e. the number of equations equals the number of unknowns. Once the number of equations equals the number of unknowns the system of equations is consistent and has a unique solution because of the absence of the fixed modes, and the coprimeness of $m_{ki}(s)$, $i = 1, \ \ldots \ , q$.

The number of control parameters of the k^{th} row of $R(s)$ is equal to the parameters in the k^{th} rows of $U(s)$ and $V(s)$. Writing this in row degree notation

$$no.\ vars. \ = \ q_\mu (\delta_{rk} U(s) \ + \ 1) \ + \ p_\mu (\delta_{rk} V(s) \ + \ 1)$$

where δ_{rk} denotes the k^{th} row degree. Since $\delta_{rk} U(s) \geq \delta_{rk} V(s)$ then the maximum number of variables of the k^{th} row of $R(s)$ is given by:

$$max.\ no.\ vars. \ = \ (q_\mu + p_\mu) (\delta_{rk} U(s) \ + 1)$$

The number of equations is equal to $(deg \ \det \ R(s) + 1)$, which can be written as

$$no.\ eqs.\ = \sum_{i=1}^{q} \delta_{ri}\, U(s) + \sum_{j=1}^{q} \delta_{cj}\, D(s) + 1$$

where δ_{ri} and δ_{cj} denotes the i^{th} row degree and the j^{th} column degree, respectively. For the linear system of equations to be consistent we need $max.\ no.\ vars. \geq no.\ eqs.$, using the above results we have the following condition

$$(q_\mu + p_\mu)(\delta_{rk}\, U(s) + 1) \geq \sum_{i=1}^{q} \delta_{ri}\, U(s) + \sum_{j=1}^{q} \delta_{cj}\, D(s) + 1$$

$$\delta_{rk}\, U(s) \geq \frac{\left(\sum_{i=1}^{q} \delta_{ri}\, U(s) + \sum_{j=1}^{q} \delta_{cj}\, D(s) \right)}{(q_\mu + p_\mu - 1)} - 1 \qquad (3.8.6)$$

For the purpose of forming a linear system of equations it is advisable to set all $\delta_{ri}\, U(s)$, $(i = 1, \ \ldots\ , q)$, $i \neq k$, to zero. This is done to avoid the unnecessary increase in the number of equations, and accordingly, the increase of $O_{u.b.}$. Hence, (3.8.6) gives

$$\delta_{rk}\, U(s) \geq \frac{\sum_{j=1}^{q} \delta_{cj}\, D(s)}{q_\mu + p_\mu - 1} - 1$$

and (3.8.5) follows. It is evident that $O_{u.b.}$ is reduced if $(q_\mu + p_\mu)$ is increased. Therefore, when expanding det $R(s)$ it is recommended to choose the k^{th} row which has the maximum $(q_\mu + p_\mu)$; and (3.8.5) follows ■.

3.9 Experimental Examples.

In this section several examples will be presented to clarify the solution procedure discussed in the preceding sections. The computations involved in obtaining the mathematical model and the solution of these examples are too complex and time consuming to perform by hand. Computer programs have been developed to do the computations on polynomial matrices, as well as to solve for the controllers parameters. Appendix E contains the source listing of the programs and routines that manipulate polynomial matrices; while Appendix F contains the source listing of the program that performs the constrained minimization of a nonlinear function subject to nonlinear constraints.

In each of the following examples we start with checking the decentralized fixed modes of the plant. After all fixed modes are isolated, we solve first for the zero order decentralized controller. If the desired closed loop poles are not satisfied the order of the controller is increased by one, the process is repeated again and again until the desired closed loop poles are satisfied. Finally the example is solved with feedback and feedforward controllers as described in Section 3.6 .

To get more insight about the resulting overall system poles, several program runs are made for each controller order with different desired poles configuration. In each case the mathematical programming model to be minimized is indicated together with the resulting solution. The solution includes the overall system poles and the controllers parameters.

Example 3.9.1

Consider the 2x2 system given in Example 3.5.2 earlier in this chapter. The transfer matrices are repeated here for the reader's convenience. Let the open loop transfer matrix be:

$$G(s) = \begin{bmatrix} \dfrac{5\,(s+0.2)}{(s+2)\,(s-1.5)} & \dfrac{s^2+2.5s+2.5}{(s+2)\,(s-1.5)} \\ \dfrac{(7s-10)}{(s+2)\,(s-1.5)} & \dfrac{s^2-0.5s-1}{(s+2)\,(s-1.5)} \end{bmatrix}$$

$$= \begin{bmatrix} s+3 & s+1 \\ s+2 & s-2 \end{bmatrix} \begin{bmatrix} s+1 & -0.5 \\ s-4 & s-1 \end{bmatrix}^{-1}$$

The open loop characteristic equation is equal to det $D(s) = (s+2)(s-1.5)$ which is unstable. And let the zero order decentralized controller be:

$$C(s) = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix} = \begin{bmatrix} u_1 & 0 \\ 0 & u_2 \end{bmatrix}^{-1} \begin{bmatrix} v_1 & 0 \\ 0 & v_2 \end{bmatrix} \qquad (3.9.1)$$

To find the fixed modes of the plant we follow the procedure of Section 3.4. Using the notation introduced in that section we have $N = 2$, $q_1 = q_2 = 1$, $p_1 = p_2 = 1$, $J_1 = \{ D_l^1 ; N_l^1 \} = \{ (s+1) \quad -0.5) ; (s+3 \quad s+1) \}$, $J_2 = \{ D_l^2 ; N_l^2 \} = \{ (s-4) \quad s-1) ; (s+2 \quad s-2) \}$ and

$$\Psi_{J_1,J_2} = \left\{ \begin{array}{ll} \det \begin{bmatrix} s+1 & -0.5 \\ s-4 & s-1 \end{bmatrix}; & \det \begin{bmatrix} s+3 & s+1 \\ s-4 & s-1 \end{bmatrix}; \\ \det \begin{bmatrix} s+1 & -0.5 \\ s+2 & s-2 \end{bmatrix}; & \det \begin{bmatrix} s+3 & s+1 \\ s+2 & s-2 \end{bmatrix} \end{array} \right\}$$

$$= \{ s^2 + 0.5s - 3 ; \quad 5s + 1 ; \quad s^2 - 0.5s - 1 ; \quad -2s - 8 \}$$

Apparently the *g.c.d.* of the above polynomials is a nonzero real number. Therefore, according to Theorem 3.2 the plant is free of fixed modes. Knowing that, let us now try to solve for the controllers parameters.

Using (3.8.5) with $\mu = 1$, the upper bound of the controller's order, $O_{u.b.}$, is found to be equal to one. Therefore, with an order 1 controller any desired poles $-x_1, -x_2, -x_3$ can be assigned arbitrarily by solving a linear system of equations in the parameters of the first rows of $U(s)$ and $V(s)$.

Case 1: In this case the controller order = 0, the desired characteristic equation is taken arbitrarily as $(s + 1)(s + 0.5)$, and the resulting closed loop poles are assumed real $(-x_1, -x_2)$. To solve for the controller parameters we follow the procedure outlined in Section 3.5 to obtain the following mathematical model:

Minimize $[(x_1 - 1.0)^2 + (x_2 - 0.5)^2]$

Subject to

$$u_1(u_2 + v_2) = 1$$

$$0.5\,u_1(u_2 - v_2) + v_1(5\,u_2 - 2\,v_2) = x_1 + x_2$$

$$-u_1(3\,u_2 + v_2) + v_1(u_2 - 8\,v_2) = x_1 x_2$$

$$x_i > 0, \qquad i = 1,2$$

Solution:

$min = 37.91848$

$x_1 = 7.13746$ $\qquad x_2 = 0.0$

$u_1 = 0.25718$ $\qquad v_1 = 0.16831$ $\qquad c_1 = 0.65444$

$u_2 = 6.23300$ $\qquad v_2 = -2.34462$ $\qquad c_2 = -0.376162$

The zero order controller did not assign the desired poles, nevertheless, it provides two real poles which are the nearest in distance to the desired ones.

Case 2: The difference between this case and Case1 is in the region of stability. The last two constraints in Case 1 are changed to $x_i > 0.1, \quad i = 1,2$.

Solution:

$min = 51.65361$

$x_1 = 8.17591$ $\qquad x_2 = 0.1$

$u_1 = 0.46069$ $\qquad v_1 = 0.32891$ $\qquad c_1 = 0.71395$

$u_2 = 3.69296$ $\qquad v_2 = -1.52228$ $\qquad c_2 = -0.41221$

Case 3: The last two constraints in Case1 are changed to $x_i > 0.25, \quad i = 1,2$.

Solution:

$min = 86.74135$

$x_1 = 10.31015 \qquad x_2 = 0.25$

$u_1 = 0.13984 \qquad v_1 = 0.11499 \qquad c_1 = 0.82230$

$u_2 = 13.44720 \qquad v_2 = -6.29597 \qquad c_2 = -0.46820$

Case 4: Let us take the resulting poles as a complex pair: $(-x - jy)$, $(-x + jy)$. And let the region of stability be $(-\infty, 0)$, we get the following mathematical model:

Minimize $[(x-0.5)^2 + y^2]$

Subject to

$u_1 (u_2 + v_2) = 1$

$0.5\, u_1 (u_2 - v_2) + v_1 (5\, u_2 - 2\, v_2) = 2x$

$-\, u_1 (3\, u_2 + v_2) + v_1 (u_2 - 8\, v_2) = x^2 + y^2$

$x > 0$

Solution:

$min = 1.20746$

$y = 0.97850 \qquad x = 0.0$

$u_1 = 6.45240 \qquad v_1 = -1.8001 \qquad c_1 = -0.278981$

$u_2 = 0.09426 \qquad v_2 = 1.45554 \qquad c_2 = 15.44176$

Case 5 Changing the stability region of Case 4 to $(-\infty, -0.25)$, the last constraint is changed to: $x > 0.25$.

Solution:

$min = 2.82065$

$y = 1.66077 \qquad x = 0.25$

$u_1 = -0.48066 \qquad v_1 = 0.18812 \qquad c_1 = -0.39138$

$u_2 = 0.25981 \qquad v_2 = -2.34029 \qquad c_2 = -9.0077$

In Cases 4 and 5, where the resulting poles are complex, we obtained closer poles to the desired ones than those of Cases 1-3, where the resulting poles are real. In all of the above cases the desired poles have not been assigned. Therefore the controller order is increased by one. Let the new controller be:

$$C(s) = \begin{bmatrix} c_1(s) & 0 \\ 0 & c_2 \end{bmatrix} = \begin{bmatrix} u_1 s + u_2 & 0 \\ 0 & u_3 \end{bmatrix}^{-1} \begin{bmatrix} v_1 s + v_2 & 0 \\ 0 & v_3 \end{bmatrix}$$

and let

$$\det \left(\begin{bmatrix} u_1 s + u_2 & 0 \\ 0 & u_3 \end{bmatrix} \begin{bmatrix} s+1 & -0.5 \\ s-4 & s-1 \end{bmatrix} + \begin{bmatrix} v_1 s + v_2 & 0 \\ 0 & v_3 \end{bmatrix} \begin{bmatrix} s+3 & s+1 \\ s+2 & s-2 \end{bmatrix} \right)$$

$$= f_3(z)\, s^3 + f_2(z)\, s^2 + f_1(z)\, s + f_0(z)$$

and

$$(s + x_1)(s + x_2)(s + x_3) = g_3(x)\, s^3 + g_2(x)\, s^2 + g_1(x)\, s + g_0(x)$$

where $z = [\, u_1, u_2, u_3;\ v_1, v_2, v_3 \,]^T$ is the controller parameter vector, and $x = [\, x_1, x_2, x_3 \,]^T$ is the resulting poles vector.

Case 6 In this case the controller is one, and the desired poles are chosen to be -1.0, -0.5 and -0.3. The mathematical model can be written as follows:

$$\textit{Minimize} \quad [(x_1 - 1.0)^2 + (x_2 - 0.5)^2 + (x_3 - 0.3)^2]$$

Subject to

$$f_i(z) = g_i(x) \qquad i = 0,1,2,3$$

$$x_i > 0 \qquad i = 1,2,3$$

Solution:

$min = 0.0$ (i.e. the desired poles are assigned)

$x_1 = 1.0$	$u_1 = 0.21300$	$v_1 = -0.11877$
$x_2 = 0.5$	$u_2 = 0.43361$	$v_2 = -0.13423$
$x_3 = 0.3$	$u_3 = 1.37603$	$v_3 = 3.31877$

Case 7 Changing the desired poles of Case 6 to -1, $-2-j$ and $-2+j$, we get:

Solution:

$min = 0.0$ (i.e. the desired poles are assigned)

$x_1 = 1.0$	$u_1 = 3.00673$	$v_1 = -1.17904$
$x_2 = 2.0$	$u_2 = 4.74326$	$v_2 = -0.11228$
$y = 1.0$	$u_3 = -0.59815$	$v_3 = 0.93073$

Case 8 In this case the controller of Figure 4 is applied with zero order. Assuming $G(s)$, $C(s)$ and the desired poles as in Case 1. And taking $H(s) = diag\,[h_1 \quad h_2] = diag\,[\frac{p_1}{q_1} \quad \frac{p_2}{q_2}]$ we get the following solution.

Solution:

$min = 37.91851$

$x_1 = 7.13747 \qquad x_2 = 0.0$

$u_1 = 4.78951 \qquad v_1 = 0.99279 \qquad c_1 = 0.20728$

$u_2 = 1.77076 \qquad v_2 = -2.52552 \qquad c_2 = -1.426235$

$p_1 = 5.06513 \qquad q_1 = 1.60431 \qquad h_1 = 0.31674$

$p_2 = 0.34942 \qquad q_2 = 1.32483 \qquad h_2 = 3.79151$

and the intermediate results:

$$\overline{D}(s) = \begin{bmatrix} 0.629s + 0.371 & 0.457s + 0.265 \\ 16.605s - 43.975 & 12.206s - 32.093 \end{bmatrix}, \quad \overline{P}(s) = \begin{bmatrix} 3.188 & 2.316 \\ 2.614 & 1.949 \end{bmatrix}$$

The poles obtained in this case are the same as those obtained in Case 1. Several program runs have been made with the same conditions as in the previous cases and the same results are obtained ■ .

Example 3.9.2

To illustrate the solution procedure in cases where the plant contains fixed modes under decentralization, we shall consider the following $2x2$ example. Let

$$G(s) = \begin{bmatrix} \dfrac{-(s+.2)(2.5s+2)}{(s+.2)(s-1)} & \dfrac{4s^2+8s+6}{(s+.2)(s-1)} \\ \dfrac{-3.5(s+.2)^2}{(s+.2)(s-1)} & \dfrac{2(s+.2)(3s+4)}{(s+.2)(s-1)} \end{bmatrix}$$

$$= \begin{bmatrix} s-1 & s+2 \\ 2s+.4 & s+.2 \end{bmatrix} \begin{bmatrix} 2s+4 & -2s-2 \\ 1.5s+.3 & -s-.2 \end{bmatrix}^{-1}$$

The open loop characteristic equation is equal to $\det D(s) = (s+.2)(s-1)$ which is unstable. And let the zero order decentralized controller be as in (3.9.1). To find the fixed modes of the plant we follow the procedure of Section 3.3.1. Using the notation introduced in that section we have $N=2$, $q_1 = q_2 = 1$, $p_1 = p_2 = 1$,

$$J_1 = \{D_1^1 ; N_1^1\} = \{(\ 2s+4 \quad -2s-2) ; (\ s-1 \quad s+2)\}$$

$$J_2 = \{D_1^2 ; N_1^2\} = \{(1.5s+.3 \quad -s-.2) ; (2s+.4 \quad s+.2)\ \}$$

and

$$\Psi_{J_1,J_2} = \left\{ \begin{array}{l} \det \begin{bmatrix} 2s+4 & -2s-2 \\ 1.5s+.3 & -s-.2 \end{bmatrix}; \ \det \begin{bmatrix} s-1 & s+2 \\ 1.5s+.3 & -s-.2 \end{bmatrix}; \\ \det \begin{bmatrix} 2s+4 & -2s-2 \\ 2s+.4 & s+.2 \end{bmatrix}; \ \det \begin{bmatrix} s-1 & s+2 \\ 2s+.4 & s+.2 \end{bmatrix} \end{array} \right\}$$

$$= \{(s+.2)(s-1); -(s+.2)(2.5s+2);\ 2(s+.2)(3s+4); -(s+.2)(s+5)\}$$

Apparently the *g.c.d.* of the above polynomials is $(s+.2)$, which is the fixed mode of the plant. Evaluating $\det R(s) = \det(U\,D(s) + V\,N(s))$ we have

$$\det R(s) = (s+.2)[u_1u_2(s-1) + u_1v_2(6s+8) + v_1u_2(-2.5s-2) + v_1v_2(-s-5)]$$

The fixed mode (s + .2) must exist in the desired characteristic equation. Therefore, it is only necessary to assign one pole. Let the desired characteristic equation be $(s + .2)(s + 1)$, then the equations to be solved are

$$(u_2 + 6v_2)u_1 - (2.5u_2 + v_2)v_1 = 1$$
$$(-u_2 + 8v_2)u_1 - (2.0u_2 + 5v_2)v_1 = 1$$

The above equations can be solved linearly by setting $u_2 = 2, v_2 = 1$, and solving for u_1 and v_1.

Solution:

$$u_1 = \frac{1}{12} \qquad v_1 = \frac{-1}{18} \qquad c_1 = \frac{-2}{3}$$
$$u_2 = 2 \qquad v_2 = 1 \qquad c_2 = \frac{1}{2} \qquad \blacksquare.$$

The above parameters of the zero order controller assigns the desired pole.

Example 3.9.3

Consider a 3 input, 4 output plant with a transfer matrix model $G(s)$ together with its right coprime matrix fraction $N(s)D^{-1}(s)$.

$$G(s) = \begin{bmatrix} \frac{s^3+4s^2+s+2}{(s-1)(s^2+4)} & \frac{-2(s+1)}{(s-1)(s^2+4)} & \frac{-4}{(s-1)(s^2+4)} \\ \frac{s^3+1}{(s-1)(s^2+4)} & \frac{(s-1)(s^2-4s+5)}{(s-1)(s^2+4)} & \frac{s^2-4s+2}{(s-1)(s^2+4)} \\ \frac{s^2-2s-5}{(s-1)(s^2+4)} & \frac{(s-1)(s^2-2s+3)}{(s-1)(s^2+4)} & \frac{s^3+4s^2-2}{(s-1)(s^2+4)} \\ \frac{s^3+2s^2+3s-8}{(s-1)(s^2+4)} & \frac{(s-1)(s^2-7s+4)}{(s-1)(s^2+4)} & \frac{7s^2-2s-4}{(s-1)(s^2+4)} \end{bmatrix}$$

$$= \begin{bmatrix} s+1 & 1 & 0 \\ s-1 & s & s-2 \\ 0 & s+1 & s+2 \\ s & 2s+2 & 3 \end{bmatrix} \begin{bmatrix} s-1 & 1 & 0 \\ -1 & s+1 & -2 \\ 0 & 2 & s-1 \end{bmatrix}^{-1}$$

The open loop characteristic equation is equal to det $D(s) = (s-1)(s^2+4)$ which is unstable. The plant is to be controlled from two control stations with the following dimensions, $q_1 = 2, p_1 = 2, q_2 = 1$ and $p_2 = 2$. The first order controller is described by

$$C(s) = \begin{bmatrix} c_1 & c_2 & 0 & 0 \\ c_3 & c_4 & 0 & 0 \\ 0 & 0 & c_5 & c_6 \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & 0 \\ u_3 & u_4 & 0 \\ 0 & 0 & u_5 \end{bmatrix}^{-1} \begin{bmatrix} v_1 & v_2 & 0 & 0 \\ v_3 & v_4 & 0 & 0 \\ 0 & 0 & v_5 & v_6 \end{bmatrix}$$

The plant is tested for fixed modes using the same procedure of the previous examples. We have

$$J_1 = \{ D_1^1; D_2^1; N_1^1; N_2^1 \}$$

$$J_2 = \{ D_1^2; N_1^2; N_2^2 \}$$

and

$$\Psi_{J_1,J_2} = \left\{ \det \begin{bmatrix} D_1^1 \\ D_2^1 \\ D_1^2 \end{bmatrix}; \det \begin{bmatrix} D_1^1 \\ D_2^1 \\ N_1^2 \end{bmatrix}; \ldots ; \det \begin{bmatrix} N_1^1 \\ N_2^1 \\ N_2^2 \end{bmatrix} \right\}$$

It is easily calculated from (3.4.4) that, Ψ_{J_1,J_2} contains 18 polynomials. To show that the plant is free of fixed modes it is sufficient to find two coprime polynomials of Ψ_{J_1,J_2} . Evaluating the following polynomials

$$\det \begin{bmatrix} D_1^1 \\ D_2^1 \\ D_1^2 \end{bmatrix} = (s-1)(s^2+4); \qquad \det \begin{bmatrix} D_1^1 \\ D_2^1 \\ N_1^2 \end{bmatrix} = (s^3+4s^2-2)$$

which are coprime. Therefore, the system is free of fixed modes.

Using (3.8.5) with $\mu = 1$, the upper bound of the controller's order, $O_{u.b.}$, is equal to zero. A direct consequence of Theorem 3.5, the arbitrary pole placement is fulfilled by solving a linear system of equations. Let the characteristic equation to be assigned be $(s+2)(s^2+s+1) = s^3+3s^2+3s+2$, and set the following controller parameters

$$u_3 = -2 \qquad u_4 = 1 \qquad u_5 = 2$$

$$v_3 = 3 \qquad v_4 = -1 \qquad v_5 = 2 \qquad v_6 = 1$$

Evaluating det $[U\,D(s) + V\,N(s)]$ and equating coefficients of like powers of this determinant and the desired characteristic equation, the following linear system of equations and its solution results:

$$\begin{bmatrix} 4 & 4 & -1 & 3 \\ 11 & 19 & -25 & 10 \\ -26 & 6 & 97 & -27 \\ -35 & -15 & -115 & -90 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 3 \\ 2 \end{bmatrix}$$

solution:

$$u_1 = 0.232295 \qquad u_2 = 0.032706$$

$$v_1 = 0.161879 \qquad v_2 = -0.181330$$

The decentralized static controller is evaluated as:

$$C = U^{-1}\,V = \begin{bmatrix} 0.21417 & -.49923 & 0 & 0 \\ 3.42835 & 1.99846 & 0 & 0 \\ 0 & 0 & 1 & .5 \end{bmatrix}$$

3.10 Design Examples

In this section several design examples will be solved using the design approach discussed earlier in this chapter. It is the intent of this section to present examples which are solved by other authors using different approaches. The design method of Section 3.5 will be applied to those examples; and the results ob-

tained are compared to the results obtained by other authors. Besides the stability constraints (3.5.8) it is of interest to append some important design constraints such as asymptotic regulation constraints including settling time and maximum overshoot. Another important practical constraiant is to bound the controller gain so that saturation effects will not arise.

Example 3.10.1 [47]: Consider the linear system of Example 1.2.3 which has been solved by Jamshidi [47]. The transfer function matrix $G(s)$ and its right coprime matrix fraction are

$$\begin{aligned} G(s) &= \frac{1}{(s+1)(s+2)} \begin{bmatrix} 0.1 & s+2.05 \\ s+1 & 0.5(s-1) \end{bmatrix} \\ &= \begin{bmatrix} 0.1 & 0 \\ 1 & -10 \end{bmatrix} \begin{bmatrix} 1 & 10(s+2.05) \\ -.1s & .95s+2 \end{bmatrix}^{-1} \end{aligned} \tag{3.10.1}$$

Let us start with a static decentralized controller of the form

$$C(s) = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix} = \begin{bmatrix} u_1 & 0 \\ 0 & u_2 \end{bmatrix}^{-1} \begin{bmatrix} v_1 & 0 \\ 0 & v_2 \end{bmatrix} \tag{3.10.2}$$

From the structure of $C(s)$ the number of control stations is $N = 2$, the order of the open loop system is $n = 2$, and the dimension of the two subcontrollers are $p_1 = q_1 = 1$, $p_2 = q_2 = 1$. It has been shown [47] that the system has no fixed modes. Thus, we can proceed with the design procedure of placing the closed loop poles. Assume that requirement on the transient response to a step should be as

fast in responding as possible and with a maximun overshoot of less than 5 percent. Furthermore, the settling time should be less than four seconds.

Let the second characteristics polynomial of the closed loop system be:

$$\begin{aligned} det\,[U\,D(s) + V\,N(s)] &= s^2 + 2\zeta\omega_n\, s + \omega_n^2 \\ &= s^2 + 2x\, s + (x^2 + y^2) \end{aligned} \tag{3.10.3}$$

where ζ is the damping ratio and ω_n is the natural frequency of the system. The maximum 5 percent overshoot requirement is satisfied by the following constraint

$$\zeta > .707 \qquad \text{or} \qquad x > y \tag{3.10.4}$$

and the settling time constraint $T_s \leq 4\ sec$ is satisfied by

$$\zeta\,\omega_n \geq 1 \qquad \text{or} \qquad x \geq 1 \tag{3.10.5}$$

where the settling time is defined as the time after which the response of the system remain withen 2 percent of its desired reference value. Suppose that the desired poles are $-1 \pm j1$ which satisfies the constraints (3.10.4) and (3.10.5) as shown in Figure 6. The optimization model can be written as:

Minimize $[(x-1)^2 + (y-1)^2]$

Subject to

$$u_1 u_2 = 1$$

$$(3u_1 + .095v_1)u_2 - 10u_1 v_2 = 2x$$

$$u_1(2u_2 - 30.5v_2) + v_1(.2u_2 - v_2) = x^2 + y^2$$

$$x \geq 1$$

$$x \geq y$$

Solution:

$$min = 0.6257385$$

$$x = 1.690340 \qquad y = 0.611650$$

$$u_1 = 0.78545 \qquad v_1 = -2.136503$$

$$u_2 = 1.27315 \qquad v_2 = -0.081365$$

Although the desired poles are not assigned, the solution has a relatively good transient behaviour. For the sake of assigning the desired poles let us consider a first order controller of the form:

$$C(s) = \begin{bmatrix} c_1 s + c_2 & 0 \\ 0 & c_3 \end{bmatrix} = \begin{bmatrix} u_1 s + u_2 & 0 \\ 0 & u_3 \end{bmatrix}^{-1} \begin{bmatrix} v_1 s + v_2 & 0 \\ 0 & v_3 \end{bmatrix} \qquad (3.10.6)$$

with the desired poles required to be $(-1 \pm j1, -2)$ we obtain the following solution:

Table 2. Comparison of design methods for Example 3.10.1

comparison item	Jamshidi [47]	design method of Section 3.5	design method of Section 3.5
type of controller	decentralized	decentralized	decentralized
closed-loop poles	-.166 ± j .138 , -1.23, -2.05	-1.69 ± j .612 -2	-1 ± j 1 ,
settling time (sec)	24.1	2.4	4
percent overshoot	2.3	.02	4.3
controller max gain	.62	2.72	1.98
controller order	2	0	1

Solution:

$$min = 0.0$$

$$x_1 = 1.000000 \qquad y_1 = 1.000000 \qquad x_2 = 2.00000$$

$$u_1 = 2.144283 \qquad u_2 = 0.711788 \qquad u_3 = 0.466352$$

$$v_1 = -0.144283 \qquad v_2 = -0.211591 \qquad v_3 = 0.922409$$

The desired poles are assigned. Table 2 summarizes the above result and compares it with those of Example 1.2.3 [47].

Example 3.10.2 [70],[69],[25]: (Rosenbrock Problem)

Consider the linear time invariant two-input, two-output system

$$\dot{x} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} x + \begin{bmatrix} -1/6 & 0 \\ 2/3 & 1 \\ 0 & 1/2 \end{bmatrix} u + E\omega \qquad (3.10.6)$$

$$y = \begin{bmatrix} 3 & -3/4 & -1/2 \\ 2 & -1 & 0 \end{bmatrix} x$$

It is desired to control the system such that the output y tracks a reference set of points y_d. This problem is solved using centralized methods by Rosenbrock [70] and Davison and Ferguson [69]. Rosenbrock obtained a centralized controller

$$u = \begin{bmatrix} 700 & 300 \\ -75 & -150 \end{bmatrix} (y - y_d)$$

which did not achieve robust regulation as illustrated by Table 3. Davison and Ferguson suggested the following controller

$$u = K_0 y + K_1 \int_0^t y - y_d) \, dt' \tag{3.10.7}$$

where K_0 and K_1 are $2x2$ constant gain matrices. With the following constraint imposed on the gains: $|k_{ij}| \leq 100$, the resulting closed loop poles were $(-1.23 \pm j3.21, -2.0, -2.67 \pm j6.67)$.

Davison and Chang [25] used a decentralized controller of the form (3.10.7) with K_0 and K_1 diagonal matrices. The problem was solved subject to the following constraints

$$\begin{aligned} &\text{Gains constraint:} \quad \sqrt{\sum_{i,j} k_{ij}} \leq 10\,000 \\ &\text{Damping factor constraint:} \quad \zeta \geq 0.45 \end{aligned} \tag{3.10.8}$$

The closed loop poles obtained was $(-.045, -.37 \pm j.773, -.578 \pm j.659)$, which implies a settling time of 10.8 *sec* and an overshoot of 22.3 percent .

Let us now apply the design procedure of Section 3.5. It is desired to find a robust decentralized controller which satisfies the constraints in (3.10.4) and (3.10.5). The system (3.10.6) has the following transfer function matrix and right coprime matrix fraction descriptions:

$$G(s) = \frac{1}{(s+1)^2}\begin{bmatrix} -(s-1) & -(s-2) \\ -(s-1/3) & -(s-1) \end{bmatrix}$$

$$= \begin{bmatrix} -1 & 0 \\ 1 & -1/3 \end{bmatrix}\begin{bmatrix} 3(s+1) & (s+1)(s-2) \\ 2(s+1) & (s+1)(s-1) \end{bmatrix}^{-1}$$

Assume the structure of (3.10.2) for the decentralized controller, and the poles $(-1 \pm j, -2)$ for the closed loop system. Then we have the following optimization problem and its solution:

Minimize $[(x_1 - 1)^2 + (y_1 - 1)^2 + (x_2 - 2)^2]$

Subject to

$$u_1 u_2 = 1$$

$$u_1 (3u_2 - v_2) - u_2 v_1 = 2x_1 + x_2$$

$$3u_1 u_2 = x_1^2 + y_1^2 + 2x_1 x_2$$

$$u_1 (u_2 + v_2) + v_1 (u_2 + 0.33333 v_2) = (x_1^2 + y_1^2) x_2$$

$$x_i \geq 1, \qquad i = 1,2$$

$$x_1 \geq y_1$$

Solution:

$min = 1.989094$

$x_1 = 1.000000$ $\quad y_1 = 0.005468$ $\quad x_2 = 1.000000$

$u_1 = 0.700125$ $\quad v_1 = -0.004095$

$u_2 = 1.428326$ $\quad v_2 = 0.008379$

Table 3 summarizes the above resultes and compares it with the results of other authors.

The solution shows that the controller gains are small, this indicates that the zero order controller is not able to assign the desired poles.

Consider the first order controller (3.10.6). It is required to assign the following closed loop poles: $(-1 \pm j1, -2 \pm j1)$ with the constraints x_i, $i = 1,2$ are changed to x_i, $i = 1,2$.

Solution:

$$min = 2.194453$$

$$x_1 = 0.99687 \qquad y_1 = 0.006023$$

$$x_2 = 0.90515 \qquad y_2 = 0.911964$$

$$u_1 = 1.32780 \qquad u_2 = 1.62525 \qquad u_3 = 0.753118$$

$$v_1 = 0.57031 \qquad v_2 = 0.57051 \qquad v_3 = -0.007191$$

Example 3.10.3 [71],[69],[25]: Consider the following linear model of a pressurized head box:

$$\dot{x} = \begin{bmatrix} -.395 & .01145 \\ -.011 & 0 \end{bmatrix} x + \begin{bmatrix} .03362 & 1.038 \\ .000966 & 0 \end{bmatrix} u + E\omega$$
$$y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (3.10.9)$$

The transfer function matrix and the right coprime matrix fractions are described by:

Table 3. Comparison of design methods for Example 3.10.2

comparison item	Rosenbrock [70]	Davison & Ferguson [69]	Davison & Chang [25]	design method of Section 3.5	design method of Section 3.5
type of comparison	centralized	centralized	decentralized	decentralized	decentralized
closed-loop poles	-.75 ± j 3.8 , -177	-1.23 ± j 3.21 , -2.67 ± j 6.67 , -2	-.37 ± j .77 , -.578 ± j .659 , -.045	-1 ± j .0055 , -1	-.997 ± j .006 , -.905 ± j .912
settling time (sec)	5.3	3.25	88.9	4	4.4
percent overshoot	56	30	22.2	0	0
controller max gain	300	100	1.9	.006	.43
controller order	0	2	2	0	1

$$G(s) = \frac{1}{s^2 + .395s + .00012595} \begin{bmatrix} .03362s + .00001106 & 1.038s \\ .000966s + .00001175 & -.011418 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & 0 \\ -34.4978 & -2.93624 \end{bmatrix} \begin{bmatrix} -35712.0255s - 11.38716 & -3039.5906s \\ 1155.7209s + .36882 & 98.44994s + .03239 \end{bmatrix}^{-1}$$

The open loop system poles are $(-.395, -.00032)$. Davison and Ferguson used a centralized controller of the form (3.10.7) to control the system. The closed loop poles they obtained were $(-2.2 \pm j2.2, -44 \pm j62)$ which indicates a settling time of 1.818 *sec* and a maximum overshoot of 4.3 percent; the constraint on the gains was $|k_{ij}| \leq 10000$. MacFarlane and Belletrutti [71] solved the same problem with the same central controller (3.10.7) using the 'characteristic locus design method'. They obtained the following closed loop poles $(-.02 \pm j.03, -1, -101)$ which indicate a settling time of 200 *sec* and a maximum overshoot of 0.9 percent. The gains they obtained were less than or equal to 1030. Davison and Chang [25] used a decentralized controller of (3.10.7). They obtained the following closed loop poles $(-2.45 \pm j2.75, -9.37, -122)$ with a maximum gain of 38 300.

Let us apply the method of Section 3.5 to this system. Assume the controller (3.10.2) and the constraints (3.10.4) and (3.10.5). The following optimization model is formulated:

[continue next page]

Minimize $[(x-1)^2 + (y-1)^2]$

Subject to

$$-2928.31467 u_1 u_2 = 1$$

$$u_1(-1156.6833 u_2 - .00001766 v_2) - 98.44994 u_2 v_1 = 2x$$

$$u_1(-.3688212 u_2 - 33.4355 v_2) + v_1(-.03239 u_2 + 2.9362 v_2) = x^2 + y^2$$

$$x \geq 1$$

$$x \geq y$$

Solution:

$$min = 0.0$$

$$x = 1.00000 \qquad y = 1.00000$$

$$u_1 = 0.11069 \qquad v_1 = 5.28433$$

$$u_2 = -0.30005 \qquad v_2 = 10.40399$$

Table 4 summarizes the above resultes and compares it with the results of other authors.

To compare results with those of Example 1.2.3, the desired poles are changed to $-2.2 \pm j2.2$, we obtain the following solution:

[continue next page]

Solution:

$$min = 0.0$$

$$x = 2.20000 \qquad y = 2.20000$$

$$u_1 = 0.14499 \qquad v_1 = 17.2725$$

$$u_2 = -0.23552 \qquad v_2 = 17.41869$$

It has to be clarified that Tables 2,3 and 4 presents comparisons of design items which are concerned with stability and transient response issues. The design methods adopted by other authors, specially Davison [24],[69], are aimed to minimize a certain performance index. The performance index is chosen such that the overall system maintains stability, fast response and reduction of disturbance effects. Therefore, their design methods are oriented towards optimal control ideas. On the other hand, the design method of Section 3.5 considers a pure pole placement problem. The poles are chosen such that stability, transient response and other design criteria are satisfied. However, the design criteria which are not satisfied by pole placement are included as constraints in the minimization problem of the design.

3.11 Conclusions

The problem of decentralized pole placement of a general linear, time invariant and initially relaxed system is studied. The input/output transfer matrix description of the plant and controller and their coprime polynomial matrix

Table 4. Comparison of design methods for Example 3.10.3

comparison item	McFarlane & Belletruti	Davison & Ferguson [69]	Davison & Chang [25]	design method of Section 3.5	design method of Section 3.5
type of controller	centralized	centralized	decentralized	decentralized	decentralized
closed-loop poles	-.02 ± j .03 , -1,-101	-2.2 ± j 2.2 -44 ± j 62	-2.45 ± j 2.75 , -9.37 , -122	-1 ± j 1 ,	-2.2 ± j 2.2 ,
settling time (sec)	200	1.82	1.63	4	1.82
percent overshoot	.9	4.3	6.5	4.3	4.3
controller max gain	1 030	10 000	38 300	47.7	119.2
controller order	2	2	2	0	0

fractions are used. Conditions for existence of fixed modes as well as realization and stability of the closed loop system are obtained in terms of the coprime polynomial matrix fractions. The pole placement problem is solved using mathematical programming techniques. The solution policy is based on reducing the controller's order as much as possible. It has been shown that by increasing the order of the controller, and accordingly, the order of the overall system, arbitrary pole assignment is achieved. An upper bound of the controller's order is found. With the upper bound order any closed loop poles can be assigned arbitrarily provided the system is free of fixed modes.

Although no general statement can be made, as can be seen from the previous tables, the present technique appears to outperform previous cited methods, particularly in achieving reasonable results with a minimal order controller.

Chapter 4: Multi-Area Interconnected Power System Control

4.1 Introduction

One of the major applications of decentralized control of large scale systems is load and frequency control of multi-area interconnected power systems. The control of large interconnections of power generation systems plays a vital role in modern societies. Safety in case of an unexpected loss of generation unit, cost effective of power generation and mutual help between the power generation areas are some among the reasons for adopting such vital control.

In electric power generation, it is desired to interconnect neighboring power systems for mutual advantages. Each power generation area helps in the sharing of overall power generation and load consumption. This is done over tie-lines which permit receiving and transmitting power from and to neighboring areas. Through the tie-lines interconnected power systems can sell and buy power to each other, and/or restore machine frequency, if a sudden loss of a generator should occur in the system.

An electric power generating system must operate at nominal values of frequency, voltage and load flow. These nominal values are kept in this state by controlling the generated power of each unit. As the demand changes from its nominal value the frequency and voltage of the system will deviate from their nominal values. Therefore, an automatic controller is required to sense this change and restore the system to its nominal state.

When two or more power generating areas interconnect their systems, the controller has more tasks to do. The controller must restore the nominal values of the net power flow between the control areas. An unexpected load change in a certain area must be satisfied by a change of power generation in the same area keeping the tie-lines power flow at their steady state nominal values.

As the interconnected power system network grows, the necessity for a decentralized controller grows. Unlike a centralized controller, the decentralized controller is composed of simple (low order) subcontrollers, one at each controlling area. A subcontroller senses inputs and produces control outputs for its area only. The collective efforts of all the subcontrollers are directed to satisfy the overall network control strategies. In adopting such a decentralized controller, many advantages over the centralized one occur. Reduction of communication links and eventually, the reliability of the controller is one advantage. Moreover, the cost of building such communication links will be reduced drastically, specially if the centralized controller is far from the generating areas. Another advantage is the simplicity of the controller. It is easier to construct, maintain and work with several low order controllers than one complex controller.

In this chapter a multi-area power generation system is introduced. The load and frequency control of this system is studied using the decentralized control method discussed in the previous chapter. In particular, it is desired to find a decentralized controller for a multi-area power system such that the closed loop system is stable (the system poles are assigned arbitrarily, if possible) and such that the tie-lines power flow and the generating area frequencies are regulated to certain load reference set points. Elgerd and Fosha [36] studied this problem without explicitly taking into account the concept of decentralization. Davison and Tripathi [23] also developed decentralized controllers for multi-area power system. Their treatment of the problem follows the work of Davison [22],[24] which has been summarized in Section (1.2). The proposed treatment here, gives a frequency domain solution using a polynomial approach.

In Section 4.2 an input/output models of a two-area power system and the decentralized controller are introduced. In Sections 4.3 and 4.4 the decentralized pole placement is solved using zero order controllers in Section 4.3 and first order controllers in Section 4.4.

4.2 Two-Area Power System and Controller Models

For simplicity and clarity, a two-area power system example is studied [20], [35],[36]. The treatment of a general multi-area system follows in a similar manner. Suppose that we have two power system areas connected by a tie-line as

in Figure 5. A system block diagram representing the interconnected power system is shown in Figure 6.

Before discussing the decentralized controller, it is useful to define the variables of Figure 6. The i represents the area number, $i = 1,2$.

$\Delta\omega_i$ = change of nominal rotational speed (frequency)

ΔP_{Li} = change of electric power load

ΔP_{mechi} = change of mechanical power

ΔP_{tie} = change of nominal value of tie-line power

M_i = angular momentum of the machine

D_i = percent change in load divided by percent change in frequency

T = tie-line stiffness coefficient

R_i = pu change in frequency divided by pu change in unit output

load ref. = load reference value

In developing the block diagram of Figure 6, it is assumed that the models are linear and time invariant, although in practice physical power systems appear to possess hysteresis nonlinearities. The system would thus be more accurately modeled if such nonlinearities were taken into account. As far as this work concern, the models are assumed linear. Another vital assumption we will make in solving the problem is that the load changes ΔP_{L_1} and ΔP_{L_2} are step changes. In practice, the load change in a certain area is the consumption of electricity by the public in that area. Therefore, it is a random process. Better analysis would be obtained if the load change was considered as a random process rather than a deterministic step signal.

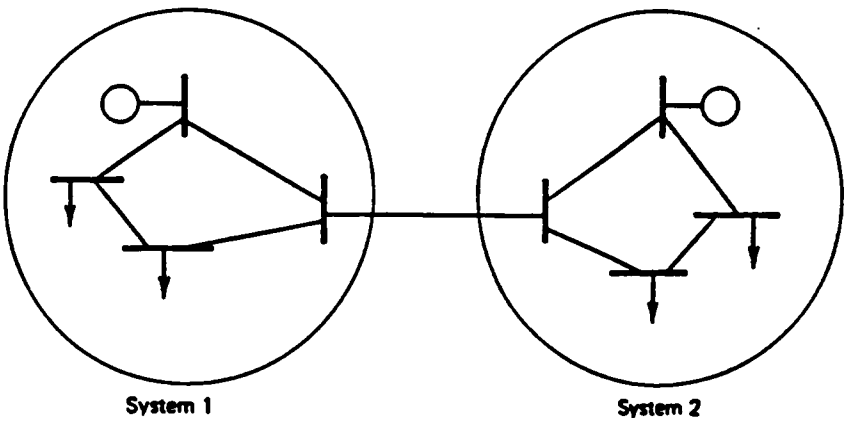


Figure 5. Two-Area System.

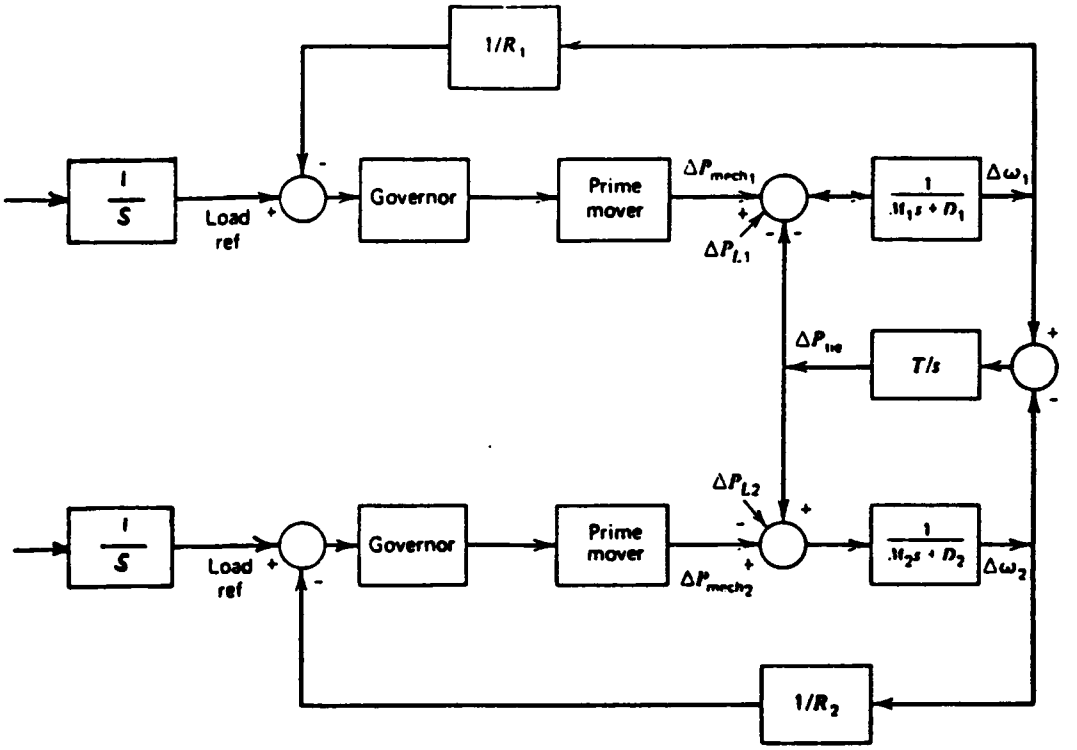


Figure 6. Block diagram of two area system.

The suggested controller of the form (4.2.2) for this power system is illustrated in Figure 7. The decentralized controller $C(s)$ is a $2x4$ block diagonal transfer matrix given by

$$C(s) = \begin{bmatrix} C_{11}(s) & C_{12}(s) & 0 & 0 \\ 0 & 0 & C_{21}(s) & C_{22}(s) \end{bmatrix} \tag{4.2.1}$$

The decentralized load and frequency problem is to find a controller $C(s)$ such that the poles of the overall system are placed arbitrarily, if possible, in the s-plane. The transient response is determined by by the location of the assigned closed loop poles. The steady state response of the system must satisfy certain control requirements. The control specifications suggested by the North American Power System Interconnection Committee (NAPSIC) [20],[36] are summarized by the following

1. The steady state frequency change due to a step-load change must be zero.
2. The transient frequency oscillation must be within $\pm$ 0.2 Hz under the nominal value.
3. The steady state tie-line power flow change due to a step-load change in either area must be zero.
4. The time error should not exceed $\pm$ 3 seconds.

Requirements 1 and 2 are to keep the frequency as smooth as possible. Requirement 3 is important for interconnected areas policy, which requires that each area must carry its own load in steady state. Requirement 4 is to reduce the timing error in synchronous clocks operated from the system frequency. One

way of satisfying these requirements, is called *area control error* (ACE) which was suggested by (NAPSIC), [20],[36]. With the configuration of Figure 7. the area control error requirements are

$$\begin{aligned} [C_{11}(s)]_{s.s.} &= B_1\,[C_{12}(s)]_{s.s.} \quad \text{or} \quad C_{11}(0) = B_1\,C_{12}(0) \\ [C_{22}(s)]_{s.s.} &= B_2\,[C_{21}(s)]_{s.s.} \quad \text{or} \quad C_{22}(0) = B_2\,C_{21}(0) \end{aligned} \tag{4.2.2}$$

4.3 Zero Order Controllers

Let us start with the lowest order controller $C(s)$ of (4.2.1). With the area control error requirement of (4.2.2) the controller can be written as

$$\begin{aligned} C(s) &= \begin{bmatrix} c_{11} & c_{12} & 0 & 0 \\ 0 & 0 & c_{21} & c_{22} \end{bmatrix} \\ &= \begin{bmatrix} u_1 & 0 \\ 0 & u_2 \end{bmatrix}^{-1} \begin{bmatrix} B_1 v_{12} & v_{12} & 0 & 0 \\ 0 & 0 & v_{21} & B_2 v_{21} \end{bmatrix} \end{aligned} \tag{4.3.1}$$

The open loop system is a ninth order system. It is reduced to fifth order by assuming that the time constants of the governer and prime mover of Figure 6, are neglected compared to the time constant of the rotating mass and load [20],[35]. Taking the following values of parameters of Figure 6, [20],[34],[35].

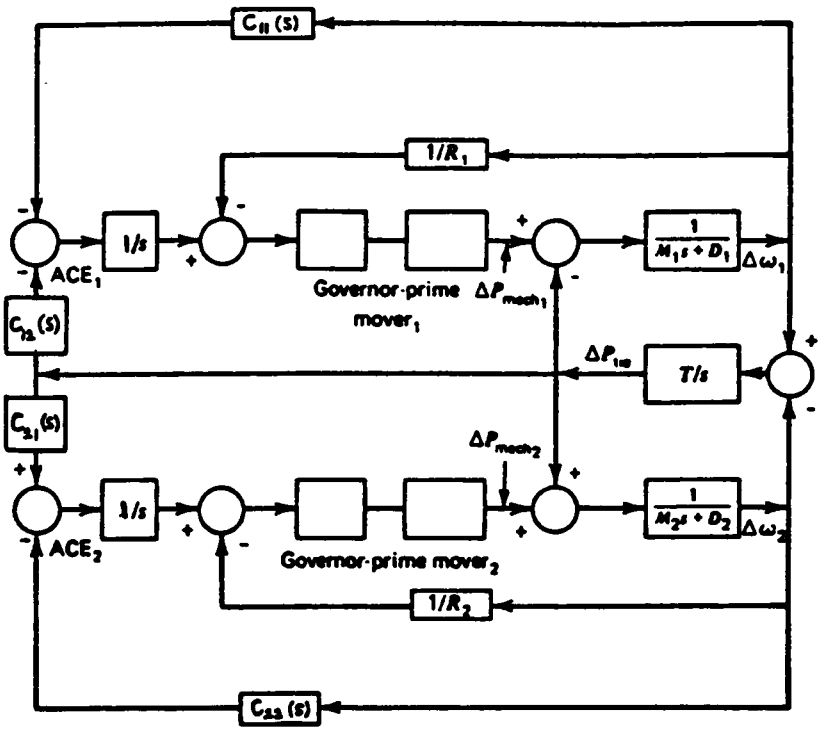


Figure 7. Block diagrams of the two-area system with the controller.

$$M_1 = 3.5\ pu \quad D_1 = 1.0\ pu \quad R_1 = .2\ pu \quad B_1 = 5\ pu$$

$$M_2 = 4.0\ pu \quad D_2 = .75\ pu \quad R_2 = .1\ pu \quad B_2 = 10\ pu \quad T = 5\ pu$$

The open loop transfer matrix of the two-area power system is

$$G(s) = \begin{bmatrix} \frac{4s^2+10.75s+3.5}{\Delta_1} & \frac{3.5}{\Delta_1} \\ \frac{14s+37.625}{\Delta_1} & \frac{-12.25s-21}{\Delta_1} \\ \frac{14s+37.625}{\Delta_1} & \frac{-12.25s-21}{\Delta_1} \\ \frac{3.5}{\Delta_1} & \frac{3.5s^2+6s+3.5}{\Delta_1} \end{bmatrix} \tag{4.3.2}$$

$$= \begin{bmatrix} 0 & 1 \\ 3.5 & 0 \\ 3.5 & 0 \\ -s & 1 \end{bmatrix} \begin{bmatrix} 3.5s & (3.5s+6)s \\ -(4s^2+10.75s+3.5)s & (4s+10.75)s \end{bmatrix}^{-1}$$

where $\Delta_1 = (14s^3+61.625s^2+90.75s+58.625)\,s^2$. The open loop poles are $(0, 0, -2.45743, -.97218 \pm j.87114)$. Now suppose that we want to assign the following poles $(-1 \pm j, -2 \pm j, -2.5)$. Then

$$\begin{aligned} \det\,[U\,D(s) + V\,N(s)] = {} & u_1u_2\,(14s^5+61.625s^4+90.75s^3+58.625s^2) \\ & + u_1v_{21}\,(35s^3+47.75s^2+14s) \\ & + v_{12}u_2\,(20s^3+67.75s^2+55.125s) + v_{12}v_{21}\,(50s+17.5) \end{aligned}$$

To solve for the controller parameters we follow the procedure outlined in Section 4.4. The following mathematical model is obtained.

Minimize $[(x_1-1)^2 + (y_1-1)^2 + (x_2-2)^2 + (y_2-1)^2 + (x_3-2.5)^2]$

Subject to

$$14u_1u_2 = 1$$

$$61.625u_1u_2 = 2(x_1 + x_2) + x_3$$

$$u_1(90.75\,u_2 + 35v_{21}) + 20v_{12}u_2 = 2x_3(x_1 + x_2) + z_1$$

$$u_1(58.625\,u_2 + 47.75v_{21}) + 67.75v_{12}u_2 = x_3z_1 + z_2$$

$$14u_1v_{21} + v_{12}(55.125\,u_2 + 50v_{21}) = x_3z_2 + (x_1^2 + y_1^2)(x_2^2 + y_2^2)$$

$$17.5v_{12}v_{21} = x_3(x_1^2 + y_1^2)(x_2^2 + y_2^2)$$

$$z_1 = (x_1^2 + y_1^2) + (x_2^2 + y_2^2) + 4x_1x_2$$

$$z_2 = 2x_2(x_1^2 + y_1^2) + 2x_1(x_2^2 + y_2^2)$$

$$x_i > 0, \qquad i = 1,2,3$$

Solution:

$min = 4.691993$ (i.e. poles are not assigned)

$x_1 = 1.049413$ $y_1 = 1.024926$

$x_2 = 0.118303$ $y_2 = 0.020155$

$x_3 = 2.066353$

$u_1 = 0.043640$ $u_2 = 1.636763$

$v_{12} = 0.007261$ $v_{21} = 0.503293$

With the zero order controller (4.2.1) the poles at the origin of the open loop transfer matrix (4.2.2) have been shifted to the left by a small amount. This shift is at the expense of adding a small oscillation and shifting the real pole to the

right. To see more of this trade off, let us require that the real part of all desired poles be to the left of $-.5$ in the s-plane i.e. $x_i \geq .5, \quad i = 1,2,3$.

Solution:

$min = 5.937393$ (i.e. poles are not assigned)

$x_1 = 1.109088$ $y_1 = 0.029088$

$x_2 = 0.500000$ $y_2 = 0.000030$

$x_3 = 1.183610$

$u_1 = 0.043640$ $u_2 = 1.636763$

$v_{12} = 0.007261$ $v_{21} = 0.503293$

4.4 First Order Controllers

From the results of the previous section it is clear that the zero order controller cannot assign the desired poles. However, there is a trade off in the resulting poles, as is usually the case, between fast responses and oscillatory ones. In this section we will increase the order of the controller for each area by one. Consider the following controller which satisfies the area control error of (4.2.2).

$$C(s) = \begin{bmatrix} u_1 s + u_2 & 0 \\ 0 & u_3 s + u_4 \end{bmatrix}^{-1} \begin{bmatrix} v_1 s + B_1 v_4 & v_3 s + v_4 & 0 & 0 \\ 0 & 0 & v_5 s + v_6 & v_7 s + B_2 v_6 \end{bmatrix} \tag{4.4.1}$$

The above decentralized controller makes the system a seventh order system. Now let us try to assign the following desired poles: $-1 \pm j, -1.5 \pm .5j, -2 \pm j$ and -3. To keep the mathematical programming model simple, only the real pole (x_3) is taken as free variable. The Determinant of $[U(s)\,D(s) + V(s)\,N(s)]$ is evaluated to obtain the following mathematical programming model:

[continue next page]

Minimize $(x_3 - 3)^2$

Subject to

$$14u_1u_3 = 1$$

$$u_1z_1 + 14u_2u_3 = 9 + x_3$$

$$u_1z_2 + u_2z_1 + 4v_1u_3 = 35.5 + 9x_3$$

$$u_1z_3 + u_2z_2 + v_1w_1 + u_3(20v_4 + 14v_3) = 78 + 35.5x_3$$

$$u_1z_4 + u_2z_3 + v_1w_2 + v_4(5w_1 + 14u_3) + v_3t_1 = 101.5 + 78x_3$$

$$14u_1v_6 + u_2z_4 + v_1w_3 + v_4(5w_2 + t_1) + v_3t_2 = 75 + 101.5x_3$$

$$v_6(14u_2 - 3.5v_1 + 35v_3) + v_4(5w_3 + t_2) = 25 + 75x_3$$

$$17.5v_4v_6 = 25x_3$$

$$z_1 = 61.625u_3 + 14u_4$$

$$z_2 = 90.75u_3 + 61.625u_4 + 3.5v_7$$

$$z_3 = 58.625u_3 + 90.75u_4 - 12.25v_5 + 6.0v_7 + 35v_6$$

$$z_4 = 58.625u_4 - 21v_5 + 3.5v_7 + 47.75v_6$$

$$w_1 = 10.75u_3 + 4u_4$$

$$w_2 = 3.5u_3 + 10.75u_4 + v_7$$

$$w_3 = 3.5u_4 - 3.5v_5 + 10v_6$$

$$t_1 = 37.625u_3 + 14u_4$$

$$t_2 = 37.625u_4 + 3.5v_7$$

$$x_3 > 0$$

Solution:

$min = 0.0$ (i.e. the desired poles are assigned)

$x_3 = 2.999983$

$u_1 = 0.283642$ $u_2 = 1.084441$ $u_3 = 0.251826$ $u_4 = 0.950625$

$v_1 = 4.310633$ $v_2 = 11.54073$ $v_3 = 0.780207$ $v_4 = 2.308145$

$v_5 = 4.134943$ $v_6 = 1.856768$ $v_7 = 3.825120$ $v_8 = 18.56768$

The controller (4.4.1) has placed the closed loop poles as desired. Now let us try to assign the following seven real poles: $(-1, -1, -1, -2, -2, -2, -7)$. Letting x_7 be the only free pole, we have the following solution.

Solution:

$min = 0.0$ (i.e. the desired poles are assigned)

$x_3 = 7.000000$

$u_1 = 0.227827$ $u_2 = 1.767075$ $u_3 = 0.313521$ $u_4 = 1.204549$

$v_1 = 5.011422$ $v_2 = 12.334795$ $v_3 = 0.315863$ $v_4 = 2.466959$

$v_5 = 2.623633$ $v_6 = 1.297143$ $v_7 = 2.985817$ $v_8 = 12.97143$

All seven real poles has been assigned as desired.

According to(3.6.1) the upper bound of the controller's order, with the restriction of the area control error (ACE), is 2.

The requirement of (3.6.1) is that one of the subcontrollers must be a second order while the other is kept at zero order. This requirement is relaxed when using the the solution method of section (3.4), which uses mathematical programming

techniques. As a result, the controller (4.4.1) equally distributes the effort among the two areas.

4.5 Conclusions

In this chapter the load and frequency control of a two-area power system has been studied. Decentralized controllers are used to arbitrarily assign the poles of the closed loop system. The method of solution is based on polynomial algebra and mathematical programming techniques. It has been shown that zero order controllers have a limited capability of modifying the poles of the two-area power generating system, while first order controllers have the required general capability. Therefore, using first order controllers, the transient response of the plant can be improved.

Chapter 5: Conclusions and Future Work.

In the world of control theory there are two basic approaches for analysis and design. The state variable description (known as modern control theory or the time-domain approach) and the input-output frequency domain approach (known as the classical control theory). The latter approach has been followed exclusively throughout this work.

Recently, the control literature showed an increasing interest in the relatively new field of the decentralized control. In response to this activity, the major problem which has been considered in this research is the decentralized pole placement for large-scale systems.

The basic assumptions that were made in developing the present mathematical model are linearity, time invariance and initial relaxedness of the system. These assumptions are essential for describing the input/output characteristics of the system as transfer matrices. Another major assumption for simplification was that all inputs of the system are deterministic. Finally, it has been assumed that the plant transfer matrix is free of hidden modes, i.e. it completely characterizes the plant.

The large scale system was considered as one complex structure with several control stations. It was made clear that the only access to the system is from these control stations. Moreover, it is explicitly indicated that each subcontroller can observe only its local measurements.

The analysis and the design phases were presented in Chapter 3. The coprime polynomial matrix fraction decomposition algorithm for the transfer matrices of the plant and the controller were used during the analysis. The controller design problem was solved using mathematical programming techniques. In summary the following points outline the contributions of this research:

1. A method of characterizing fixed modes using polynomial matrix fractions.

2. The design of output feedback decentralized compensators which assign the poles of the overall system with the objective of reducing the order of the controller.

3. An upper bound on the order of the decentralized controller was found. When the controller's order is taken as the upper bound, the decentralized pole placement problem is reduced to solving a linear system of equations.

The advantages of using the pole assignment method presented here over existing methods can be summarized as follows:

1. The design problem of the decentralized pole assignment is reduced to solving a nonlinear mathematical programming problem which is a well defined problem in the literature.

2. The method performs an intensive search to minimize the order of the controller.

3. In case the desired poles can not be assigned for a certain controller order, the method will provide alternative poles close to the desired ones such that the distance norm is minimized.

4. The method provides freedom of selecting which subcontroller to increase its order. Therefore, the complexity of the controller can be distributed over all subcontrollers.

5. A variety of constraints can be added to the design problem easily. This is done by appending those constraints to the minimization problem.

6. It is a unified design methodology. The method is applicable to a variety of control problems: arbitrary pole placement, noise rejection, system parameter disturbance rejection and others.

Future Work

The following potential issues can be pursued:

- Extending the method to include stochastic control problems.

- Applying the method to optimal control problems. This can be done by modifying the mathematical programming model used to solve the controller such that it can optimize a certain performance index.

- Find a tighter upper bound for the controller order than the one developed in (3.7.1).

- Developing a procedure to choose which subcontroller is to be selected such that increasing its order will minimize the order of the overall decentralized controller.

- Applying this method to systems which undergo structural perturbations, i.e. addition or removal of a subsystem or part of it, or a link between two subsystems. The interest in this problem is to maintain the system stability and/or pole assignment under the new structure of the system.

REFERENCES

1. H. H. Rosenbrock, State-Space and Multivariable Theory , Nelson, London, 1970.

2. T. Kailath, Linear Systems , Englewood Cliffs, N.J. Prentice-Hall, 1980.

3. V. Kucera, Discrete Linear Control , John Wiley, 1979.

4. S. Barnett, Polynomials and Linear Control Systems , Marcel Dekker, New York, 1983.

5. W. A. Wolovich, Linear Multivariable Systems , Springer-Verlag, New York, 1974.

6. C. T. Chen, Linear System Theory and Design , H.R.W., New York, 1984.

7. L. A. Zadeh and C. A. Desoer, Linear System Theory , Krieger, New York, 1979.

8. S. Krishnarao and C. T. Chen, "Two Polynomial Matrix Operations," IEEE Trans, AC, Vol. AC-29, No. 4, April 1984.

9. N. Munro, Modern Approach to Control System Design , Peregrinus, New York, 1979.

10. F. R. Gantmacher, The Theory of Matrices , Chelsea, 1959.

11. G. Stang, Linear Algebra and its Applications , Academic Press, New York, 1980.

12. F. E. Hohn, Elementary Matrix Algebra , Macmillan, New York, 1973.

13. Mahmoud, Hassan and Darwish, Large-Scale Control Systems , Marcel Dekker, New York, 1985.

14. D. D. Siljak, Large Scale Dynamic Systems , North Holland, New York, 1978.

15. M. Singh, Decentralized Control , North-Holand, New York, 1981.

16. H. Witsenhausen, "A Counter Example in Stochastic Optimal Control, " SIAM J. of control, 6, 1, 1968.

17. N. Sandell, P. Varaiya, M. Athans and M. Safonov, "A Survey of Decentralized Control Methods of Large Scale Systems," IEEE Trans. Autom. Control, vol. AC-23, No. 2, April 1978, pp. 108-128.

18. J. P. Corfmat and A. S. Morse, "Control of Linear Systems Through Specified Input Channels," SIAM J. Control, vol. 14, Jan. 1976.

19. J. P. Corfmat and A. S. Morse, "Decentralized Control of Linear Multivariable Systems," Automatica, vol. 12, Sep. 1976.

20. A. Wood and B. Wollenberg, Power Generation Operation and Control , John Wiley & sons, New York 1984.

21. S. H. Wang and E. Davison "On the Stabilization of Decentralized Control Systems," IEEE Trans., Vol. AC-18, pp. 473-478, Oct. 1973.

22. E. Davison, "Decentralized Stabilization and Regulation," in Directions in Large Scale Systems, Y.C. Ho and S. K. Mitter, Ed. New York, Plenum, 1976.

23. E. Davison and N. Tripathi, "The Optimal Decentralized Control of Large Power Systems: Load and Frequency Control," IEEE Trans. Auto. Control, vol. Ac-23, no.2, April 1978, pp 312-25.

24. E. Davison, "The Robust Decentralized Control of a General Servomechanism Problem," IEEE Trans. Auto. Control, vol. AC-21, No. 1, Feb. 1976, pp 14-24.

25. E. Davison and T. Chang, "The Design of Decentralized Controllers for the Robust Servomechanism Problem Using Parameter Optimization Methods," Proceedings of the 1982 American Control Conference, Arlington, VA, USA 14-16 June 1982, vol.3, pp. 905-9.

26. P. Fessas, "Decentralized Control of Linear Systems via Polynomial Matrix Methods," Int. J. Control, 1979, 30 pp. 259-276 and 1980, 32 pp. 127-147.

27. _____, "Two Equivalent Approaches in the Decentralized Control of Linear Systems", IEEE Trans., 1981 AC-26.

28. _____, "Matrix Fraction Description Approach to Decentralized Control," IEE Proc., Vol. 129, Pt. D, No. 5, Sep. 1982.

29. _____, "D-Controllability of Linear Systems Defined by their Transfer Matrices," IEE Proc., Vol. 129, Pt. D, No. 5, Sep. 1982.

30. _____, "Decentralized Control of N-Channel Systems", Ph.D. Dissertation, Swiss Federal Institute of Technology, Zurich 1979.

31. A. Ramakrrishna and N. Viswanadham, "Decentralized Control of Interconnected Dynamical Systems," IEEE Trans, Vol. AC-27, No. 1, Feb. 1982.

32. M. Vidyasagar and N. Viswanadham, "Algebraic Characterization of Decentralized Fixed Modes and Pole Assignment," Syst. & Control Lett. (Netherlands), vol. 3, no. 2, p69-72, July 1983.

33. D. K. Lindner, "Distributing the Control Effort in Decentralized Systems," Proc. of the 4th IFAC/IFORS Symposium on Large Scale Systems Theory and Application, Zurich, Switzerland, Vol. 1, 1986, pp. 287-290.

34. V. Armentano and M. G. Singh, "A Procedure to Eliminate Decentralized Fixed Modes with Reduced Information Exchange," IEEE Trans, Vol. AC-27, No. 1, Feb. 1982.

35. IEEE Committee Report, "Dynamic Models for Steam and Hydro Turbines in Power System Studies," IEEE Trans, 1972.

36. O. Elgerd, Electric Energy Systems Theory , McGraw-Hill, New York, 1982.

37. O. I. Elgerd and C. E. Fosha, "Optimum Megawatt-Frequency Control of Multiarea Electric Energy System," and "The Megawatt-Frequency Control Problem: A new Approach Via Optimal Control Theory," IEEE Trans. Power Appar. Syst., Vol. PAS-89, no. 4, April 1970, pp. 556-576.

38. D. G. Luenberger, Introduction to Linear and Nonlinear Programming , Addison Wesley, Menlo Park, California, 1973.

39. M. S. Bazaraa, Nonlinear Programing Theory and Algorithms , John Wiley, New York, 1979.

40. M. Avriel, Nonlinear Programming Analysis and Methods , Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

41. Burden, Faires and Renolds, Numerical Analysis , Prindle, Weber & Schmidt, Boston, 1978.

42. B. A. Murtagh and M. A. Suanders, Large-Scale Linearly Constrained Optimization, Mathematical Programming, 14, 1978, pp. 41-72.

43. L. S. Lasdon, "Large-Scale Nonlinear Programming," Dec. 6, 1982, Dept. of General Business, Working Paper 82/83-3-1.

44. S. Balachandran, "Dynamic Scaling of Nonlinear Programs and its Implementation within the Augmented Lagrangian Penalty Function Technique," Ph.D. Dissertation, I.O.R. Dept., Virginia Polytechnic institute and State University.

45. M. P. Kastner, "A New Look at Large-Scale Systems and Decentralized Control: Recent Graduates Speak out," IEEE Conference on Decision and Control, 1978, pp. 472-473.

46. D. B. Petkvski, "Decentralized Exponential Stabilization of Large-Scale Systems: A Minmax Approach," IEEE 1980.

47. M. Jamshidi, Large-Scale Systems Modeling and Control , North-Holland, New York, 1983.

48. H. Witsenhausen, "Separation of Estimation and Control for Discrete Time Systems," Proc. IEEE, pp. 1557-1566, 1971.

49. M. Aoki, "On Decentralized linear Stochastic Control Problems with Quadratic Cost," IEEE Trans. AC 18, pp. 243-258, 1973.

50. N. Sandell and M. Athans, "Solution of Some Non-Classical LQG Stochastic Decision Problems," IEEE Trans. AC 19, 2, pp. 108-116, 1974.

51. M. Hassan and M. Singh, "Robust Decentralized Controller for Linear Interconnected Dynamic Systems," Proc. IEEE, Vol. 125, 5, pp 429-432, 1978.

52. N. Viswanadham and A. Ramakrishna, "Decentralized Regulation of Large Dynamic Systems with Engineering Applications, " North-Holland Publishing Company, Large Scale Systems 2, 1981, pp. 191-204.

53. M. Mahmoud and M. Singh, "Decentralized State Reconstruction of Interconnected Discrete Systems," North-Holland Publishing Company, Large Scale Systems 2, 1981, pp.151-158.

54. C. T. Chen, "Representation of linear time-invariant composite systems," IEEE Trans., vol. AC-12, p.201, 1967.

55. E. J. Davison, " A Method for Simplifying Linear Dynamic Systems," IEEE Trans. Automat. Contr., vol. AC-11, Jan 1966, pp. 93-101.

56. F. N. Bailey and H. K. Ramapriyan, "Bound on Suboptimality in the Control of Linear Dynamic Systems," IEEE Trans. Automat Contr., vol. AC-18, Oct. 1973, pp.532-534.

57. P. V. Kokotovic, "Separation of Time Scales in Modeling and Control," 1975 IEEE Conf. on Decision and Control, Houston, TX, Dec. 1975.

58. B. Anderson and D. Clements, "Algebraic Characterization of Fixed Modes in Decentralized Control," IFAC J. Automatica 17: 703-712, 1981.

59. B. Anderson, "Transfer Function Matrix Description of Decentralized Fixed Modes," IEEE Trans. Automat. Cont., Vol. AC-27, no. 6, Dec. 1982.

60. E. J. Davison and U. Ozguner, "Characterization of Decentralized Fixed Modes for Interconnected Systems," Automatica, Vol. 19, No. 2, pp. 169-182, 1983.

61. H. Seraji, "On Fixed Modes in Decentralized Control Systems," Int. J. Control, Vol. 35, No. 5, p. 775-784, 1982.

62. M. Singh, A. TiTli, and K. Malinowski, "Decentralized Control Design: An Overview," Large Scale Systems, vol. 9, 1985, pp. 215-230.

63. E. J. Davison, "The Robust Control of a Servomechanism Problem for Linear Time Invariant Multivariable Systems," IEEE Trans. Autom. Cont., vol. AC-21, pp. 25-34.

64. E. J. Davison, "Decentralized Robust Control of Unknown systems using tuning regulators", IEEE Trans. Autom. Cont., vol. AC-23, pp. 276-289.

65. E. J. Davison, "The Robust Decentralized Control of a Servomechanism Problem of Composite Systems with input-output interconnections," IEEE Trans. Autom. Cont., vol. AC-24, pp. 325-327.

66. E. J. Davison and T. Chang, "The Design Decentralized Controlers for the Robust Servomechanism Problem Using Parameter Optimization Methods - Some Case Studies," IEEE 1982.

67. J. Bernussou and A. Titli, Interconnected Dynamical Systems : Stability, Decomposition and Decentralization , North-Holand, Amesterdam, 1982.

68. M. Hassan and M. Singh, "A Decentralized Controller with trajectory improvement," Proc. IEE, May 1980.

69. E. J. Davison and J. Ferguson, "The Design of Controllers for the Mutivariable Robust Control of a Servomechanism Problem Using Parameter Optimization Methods," IEEE Trans. Autom. Cont., vol. AC-26, Feb. 1981, pp. 93-110.

70. H. Rosenbrock, "Design of Multivariable Control Systems Using the Nyquist Array," Proc. IEE, vol. 116, 1969, pp.1929-1936.

71. A. MacFarlane and J. Belletrutti,"The Characteristic Locus Design Method," Automatica, vol. 9, 1973, pp.575-588.

72. A. Waren, M. Hung and L. Lasdon, "The Status of Nonlinear Programming Software: An Update," Operation Research, vol. 35, no. 4, July-August 1987.

Appendix A. Determinant Expansion Theorem

Theorem: Let A_i denote the i^{th} column of an nxn matrix A. And let

$$A_i = \sum_{k=1}^{p} B_k$$

then

$$\det A = \sum_{k=1}^{p} \det [A_1, A_2, \ldots, A_{i-1}, B_k, A_{i+1}, \ldots, A_n]$$

The theorem is proved in exactly the manner suggested by the following example:

$$\det \begin{bmatrix} b_{11} + b_{12} & a_1 \\ b_{21} + b_{22} & a_2 \end{bmatrix} = (b_{11} + b_{12})\, a_2 - (b_{21} + b_{22})\, a_1$$

$$= (b_{11}\, a_2 - b_{21}\, a_1) + (b_{12}\, a_2 - b_{22}\, a_1)$$

$$= \det \begin{bmatrix} b_{11} & a_1 \\ b_{21} & a_2 \end{bmatrix} + \det \begin{bmatrix} b_{12} & a_1 \\ b_{22} & a_2 \end{bmatrix}$$

Appendix B. Optimization Algorithms

B.1 Minimizing a Function in One Variable

The problem to be solved is

$$\textit{Minimize} \quad f(x) \qquad \text{where } x, f(\cdot) \in R \tag{B.1}$$

A well known method for minimizing a function f in one variable x is the *Quadratic Fit Method.* The idea behind this method is to approximate f by a quadratic function ϕ:

$$\phi(x) = a_2 x^2 + a_1 x + a_0$$

where the coefficients a_0, a_1 and a_2 are evaluated by equating $\phi(x_i)$ with $f(x_i)$ at three points x_1, x_2 and x_3 and solving a system of three equations in three unknowns. The approximate minimum of f, provided $x_2 > 0$ is then $(\hat{x} = -a_1/2a_2)$ which is the minimum of ϕ .

We can find a new quadratic approximation and a better approximation of f by choosing a new set of three points as follows. The new x_2 is set to $\hat{x}$ if

$f(\hat{x}) < f(x_2)$, otherwise x_2 remains as is. The new x_1 and x_3 are chosen to be the two points adjacent to the new x_2. The algorithm is terminated when $| f(\hat{x}) - \phi(\hat{x}) | < \varepsilon_1$.

To avoid the situation where $\hat{x}$ is a local maxima, we have to bracket the initial three points x_1, x_2 and x_3 such that $f(x_2) < f(x_1)$ and $f(x_2) < f(x_3)$. The following algorithm describes the detailed steps for quadratic polynomial approximation method with bracketing.

Algorithm B.1

Input x_0 initial starting point

h interval length for bracketing process ($h > 0$)

ε_1 tolerance required in solution

N_1 max. number of iterations for bracketing

N_0 max. number of iterations for quadratic fit

output solution x or failure message

Note Steps 1, ... , 7 are for bracketing,

Steps 8, ... , 15 are for quadratic fit

Step 1 evaluate $y_1 = f(x_1)$, set $x_1 = x_1 + h$, evaluate $y_2 = f(x_2)$

Step 2 If $y_1 \geq y_2$ then GO TO *Step* 3 else GO TO *Step* 6

Step 3 set $i = 0$; while $(y_1 \geq y_2)$ and $(i < N_1)$ do

$x_1 = x_1 + h$; $y_1 = y_2$; $y_2 = f(x_1)$; $i = i + 1$

Step 4 set $x_3 = x_1$; $x_2 = x_3 - h$; $x_1 = x_2 - h$; GO TO *Step* 8

Step 5 set $y_1 = y_2$; $x_1 = x_1 - h$; $y_2 = f(x_1)$

Step 6 set $i = 0$, While $(y_1 \geq y_2)$ and $(i < N_1)$ do

$x_1 = x_1 - h$; $y_1 = y_2$; $y_2 = f(x_1)$; $i = i + 1$

Step 7 set $x_2 = x_1 + h$; $x_3 = x_2 + h$; GO TO *Step* 8

Step 8 $k = 0$; while $(k < N_0)$ *do* *Steps* 9 – 14

Step 9 solve the following system of equations

$$a + bx_1 + cx_1^2 = f(x_1)$$

$$a + bx_2 + cx_2^2 = f(x_2)$$

$$a + bx_3 + cx_3^2 = f(x_3)$$

Step 10 evaluate $\hat{x} = -b/2c$; $y_1 = f(x)$; $y_2 = f(x_2)$; $w_1 = a + b\hat{x} + c\hat{x}^2$

Step 11 if $| y_1 - w_1 | < \varepsilon_1$ then output(Solution = $\hat{x}$) ; and STOP

Step 12 if $\hat{x} > x_2$ then

if $y_1 < y_2$ then $x_1 = x_2$; $x_2 = \hat{x}$ else $x_3 = \hat{x}$

Step 13 if $\hat{x} < x_2$ then

if $y_1 < y_2$ then $x_2 = \hat{x}$; $x_3 = x_2$ else $x_1 = \hat{x}$

Step 14 $k = k + 1$ GO TO Step 9

Step 15 output(Method failed after N_0 iterations)

B.2 Unconstrained Minimization

The problem considered is:

$$\textit{Minimize} \quad f(x) \qquad x \in R^n, \quad f(\cdot) \in R \tag{B.2}$$

A quasi-Newton method will be used to solve this problem. A quasi-Newton method is an iterative process to solve unconstrained programs like the one in (B.2). The idea of this method depends on generating a sequence of vectors $\{ x_k \}$ in R^n according to the following recursive formula

$$x_{k+1} = x_k - \alpha_k S_k g_k$$

where S_k is a symmetric nxn positive definite matrix, α_k is a scaler chosen to minimize $f(x_{k+1})$ and g_k is the gradient of f i.e. $g_k = \nabla f(x_k)$. If S_k is the inverse of the Hessian of f we obtain Newton's method, while if $S_k = I$ we have the steepest decent method. Quasi-Newton methods are based on selecting S_k as an approximation to the inverse of the Hessian of f. One of the best known methods to update the approximation of the inverse of the hessian is the Davidon-Fletcher-Powel method which utilizes the following formula:

$$S_{k+1} = S_k + \frac{p_k p_k^T}{p_k^T q_k} - \frac{S_k q_k q_k^T S_k}{q_k^T S_k q_k} \qquad (B.3)$$

where $q_k = g_{k-1} - g_k$ and and $p_k = x_{k-1} - x_k$. The following algorithm is used to solve the unconstrained program (B.2) using the update of the Hessian given by (B.3).

Algorithm B.2

Input x initial starting point

S Initial approximation of the inverse of the Hessian of f

ε_1 tolerance required in solution

N max. number of iterations

output solution x or failure message

Step 1 evaluate $g = \nabla f(x)$; set $k = 1$

Step 2 set $d = -Sd$ else GO TO *Step* 6

Step 3 minimize $f(x+\alpha d)$ with respect to α, and obtain $\hat{\alpha}$

set $p = \hat{\alpha}\, d$; $\quad x = x + p$

Step 4 while $(k < N)$ do *Steps* $5 - 10$

Step 5 set $g_1 = g$; $\quad g = \nabla f(x)$; $\quad q = g_1 - g$

Step 6 update $S = S + \dfrac{p\,p^T}{p^T q} - \dfrac{S\,q\,q^T S}{q^T S\,q}$

Step 7 set $d = -\,S\,g$

Step 8 minimize $f(x+\alpha d)$ with respect to α, and obtain $\hat{\alpha}$

set $p = \hat{\alpha}\, d$; $\quad x = x + p$

Step 9 if $\| p \| < \varepsilon_1$ then output(x); and STOP

Step 10 $k = k + 1$

Step 11 output(max. number of iterations exceeded).

B.3 Equality and Inequality Constrained Problem

Consider the following constrained mathematical program:

$$\begin{aligned} &\textit{Minimize} && f(x) && x \in R^n, \quad f(\cdot) \in R \\ &\textit{Subject to} && g_i(x) = 0 && i = 1,2, \ldots, e \\ & && g_i(x) \geq 0 && i = e+1, \ldots, m \end{aligned} \tag{B.3}$$

The algorithm to solve the above program is presented in [40],[44] using the augmented Lagrangian penalty function technique. For this program let us define the following augmented Lagrangian penalty function $M(x, \lambda, \sigma)$:

$$M(x, \lambda, \sigma) = f(x) + \sum_{i=1}^{e} (.5\, \sigma_i\, g_i^2(x) - \lambda_i\, g_i(x)) + \sum_{i=e+1}^{m} (1/4\sigma_i)\, \Phi(x, \lambda, \sigma)$$

and the gradient of M with respect to x is:

$$\nabla_x M(x, \lambda, \sigma) = \nabla_x f(x) - \sum_{i=1}^{e} (\lambda_i - \sigma_i\, g_i(x))\, \nabla_x\, g_i(x) - \sum_{i=e+1}^{m} \Theta\,(\lambda_i - 2\sigma_i\, g_i(x))\, \nabla_x\, g_i(x)$$

where $\Theta(t) = \max\{t,0\}$ and $\Phi(x, \lambda, \sigma) = \{[\Theta(\lambda_i - 2\sigma_i g_i(x))]^2 - \lambda_i^2\}$.

Algorithm B.3

Input x^0 initial starting point

λ^0 initial values of multipliers

σ^0 initial values of penalty factors

a^0 initial max. constraint violation in absolute value

ε_1 tolerance required in solution

ε_2 tolerance required in constraints

N_0 max. number of iterations

output solution x or failure message

Step 1 set $k = 0$

Step 2 while ($\| x^{k+1} - x^k \| > \varepsilon_1$ or $\| a^{k+1} - a^k \| > \varepsilon_2$)

do *Steps* 3 – 5; otherwise output(x^k), and Stop

Step 3 $\underset{x}{Minimize}\ M(x, \lambda^k, \sigma^k)$

set the unconstrained minimizer of M to x^{k+1}

Step 4 update the following:

$$\lambda_i^{k+1} = \lambda_i^k - \sigma_i^k g_i(x^{k+1}) \qquad i = 1, \ldots, e$$

$$\lambda_i^{k+1} = \Theta\,[\lambda_i^k - 2\sigma_i^k g_i(x^{k+1})\,] \qquad i = e+1, \ldots, m$$

$$\sigma_i^{k+1} = 10\,\sigma_i^k$$

$$a^{k+1} = \text{max. violation of the constraints at } x^{k+1}$$

Step 5 set $k = k + 1$

Step 6 output(Max. number of iterations exceeded).

Appendix C. Coprimeness Test For Polynomials

Definition C.1 : Let $a(s)$ and $b(s)$ be two polynomials of degrees n and m respectively, with $m \leq n$. And let

$$a(s) = s^n + a_1 s^{n-1} + \ldots + a_n$$

$$b(s) = s^m + b_1 s^{m-1} + \ldots + b_m \qquad (C.1)$$

then the ***Sylvester's resultant matrix*** associated with $a(s)$ and $b(s)$ is

$$S = \begin{bmatrix} 1 & a_1 & a_2 & \ldots & a_n & 0 & \ldots & 0 & 0 \\ 0 & 1 & a_1 & \ldots & a_{n-1} & a_n & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \ldots & \ldots & \ldots & a_{n-1} & a_n \\ 1 & b_1 & b_2 & \ldots & b_m & 0 & \ldots & 0 & 0 \\ 0 & 1 & b_1 & \ldots & b_{m-1} & b_m & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \ldots & \ldots & \ldots & b_{m-1} & b_m \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ m \text{ rows} \\ \\ \\ \\ \\ n \text{ rows} \\ \\ \end{matrix} \qquad (C.2)$$

Theorem C.1 : Let $a(s)$ and $b(s)$ be as defined in (C.1). Then $a(s)$ and $b(s)$ are coprime if and only if the Sylvester's resultant matrix associated with them is nonsingular.

Appendix D. Polynomial Matrices Algorithms

This appendix presents necessary algorithms for operations on polynomials and polynomial matrices. Algorithm D.1 evaluates the greatest common left divisor of two polynomials. Algorithm D.2 calculates the determinant of a square polynomial matrix. Algorithm D.3 calculates the greatest common divisor of two polynoial matrices. The last algorithm produces the right coprime matrix factorization of a transfer function matrix. For detailed analysis of algorithms pertained to arithmatic on polynomial matrices the reader is advised to refer to [3],[4].

Algorithm D.1 [3]:

Given two polynomials a,b this algorithm produces the greatest common divisor of a and b together with two pairs of coprime polynomials p,q and r,s such that

$$\begin{aligned} a\,p + b\,q &= g \\ a\,r + b\,s &= 0 \end{aligned} \qquad (D.1)$$

Set $F = [a, b]$ and $V = I_{2x2}$

Step 1 Set $V = I_2$.

Step 2 Select the lower degree nonzero polynomial in F. If only nonzero polynomial, go to *Step* 6.

Step 3 Divide the leading coefficient of the lower degree polynomial in F into the leading coefficient of the other polynimial, calling the result λ. Subtract the degree of the lower degree polynomial in F from the degree of the other polynomial calling the result n.

Step 4 Subtract λd^n times the lower degree polynomial. Perform same operation on the corresponding column of V.

Step 5 Go to *Step* 2.

Step 6 If the nonzero polynomial appears in the second column of F, interchange the columns of both F and V. Stop.

When the above procedure is completed, the polynomials (a,b) in F are replaced by $(g,0)$. And the identity in V is replaced by

$$V = \begin{bmatrix} p & r \\ q & s \end{bmatrix}$$

Algorithm D.2 [3]

Given an mxm square polynomial matrix A, this algorithm calculates det A, or a scalar multiple of it.

Step 1 $k = 0$

Step 2 $k = k + 1$

Step 3 If $k = m$ go to *Step* 10

Step 4 Determine position of least degree nonzero polynomial in the k^{th} row of columns $k,k+1, \ldots m$ of A, say a_{kj}. If all zero go to *Step* 10 .

Step 5 If $j \neq k$ interchange columns j and k of A

Step 6 If only one nonzero polynomial in k^{th} row of the columns A, go to *Step* 3 .

Step 7 Divide the leading coefficient of a_{kk} into the leading coefcient of a_{kn}, $n = k+1, \ldots ,m$, calling the result λ_n . Subtract the degree of a_{kk} from the degree of a_{kn}, $n = k+1, \ldots ,m$, calling the result u_n.

Step 8 Subtract $\lambda_n d^{u_n}$ times column k of A from column n , $n = k+1, \ldots ,m$.

Step 9 Go to *Step* 4.

Step 10 $\delta = a_{11}\, a_{22} \ldots a_{mm}$, Stop.

At the end of the process det A is simply the product of the diagonal elements of A which is δ .

Algorithm D.3 [3]

Given *lxm* and *lxq* polynomial matrices A and B; this algorithm produces the greatest common left divisor of A and B , G along with two pairs of right coprime polynomial matrices P,Q and R,S such that

$$\begin{aligned} A\,P + BQ &= G \\ A\,R + BS &= 0 \end{aligned} \qquad (D.2)$$

Analogues to the scalar case of Algorithm D.1, form the matrix $F = [A \quad B]$ and transform it using elementary column operations to a generalized lower triangular matrix form $[\overline{G} \quad 0]$. The same column operation are performed on an identity matrix V to produce

$$V = \begin{bmatrix} \overline{P} & R \\ \overline{Q} & S \end{bmatrix} \tag{D.3}$$

The matrices G,P and Q are obtained by appending $l - r$ zero columns to $\overline{G}, \overline{P}$ and $\overline{Q}$, where r is the rank of F.

Step 1 Set $V = I_{m+q}$.

Step 2 $k = 0$.

Step 3 $k = k + 1$.

Step 4 If $k = l + 1$ or $k = m + q$, Stop.

Step 5 Determine position of least degree nonzero polynomial in row k of the block of columns $k, k + 1, \ldots, m + q$ of F, say f_{kj}. IF all zero go to *Step* 3.

Step 6 If $j \neq k$ interchange columns j and k of both F and V.

Step 7 If only one nonzero polynomial in row k of block columns $k, K + 1, \ldots, m + q$ of F, go to *Step* 3.

Step 8 Divide the leading coefficient of f_{kk} into the leading coefficient of f_{kn}, $n = k + 1, \ldots, m + q$ calling the result λ_n. Subtract the degree of f_{kk} from the degree of f_{kn}, $n = k + 1, \ldots, m + q$, calling the result u_n.

Step 9 Subtract $\lambda_n d^{u_m}$ times column k of F from column

$n,\ n = K+1,\ \ldots\ , m+q$. Perform same operation on V.

Step 10 Go to *Step* 5.

When the process is completed, $\overline{G}$ is formed by the nonzero columns of F; let r be the number of those columns. The matrices $\overline{P}$, $\overline{Q}$ of dimensions mxr , qxr and the matrices R , S are extracted from V as shown in (D.3).

Algorithm D.4 [3]

This algorithm produces a right coprime fraction $B_1 A_1^{-1}$ of an lxm transfer function matrix S. Write S in the form $S = A_r^{-1} B_r$, where A_r is a diagonal matrix consisting of the least common denominators of the rows of S and B_r is the matrix of numerators of S when written in terms of the least common denominators. The fraction (A_r, B_r) is not, in general, left coprime fraction. Thus, we can cancel out the greatest common left divisor of A_r and B_r. This can be done using Algorithm D.3, where equation (D.2) is calculated

$$A_r P + B_r Q = G$$

$$A_r R + B_r S = 0$$

then $B_1 = -R$ and $A_1 = S$.

Appendix E. Program Listings for Polynomial Matrices

```
PROGRAM GCD;
CONST
    maxDim    = 3;      { upper bound on No. of rows and columns }
    maxDeg    = 18;     { for InversePM =(maxdim*(maxdim+1)*N)/2 }
                        { for GCDPM,DeT,SmithF  = maxdim*N      }
                        { for SpectralFactorPM=(N+1)*(N+2)/2    }
                        { for others = N   =max. deg of polys.  }

    RGCDX    = 6;       { if A is lxm  and B is Pxm then use :  }
    RGCDY    = 9;       { RGCDX = 1+p  and  RGCDY = 1+p+m       }
    INVRX    = 8;       { for A  nxn  INVRX=2*n ; INVRY=n       }
    INVRY    = 4;
    Iwidth    = 5;      { integer field width.                  }
    fieldwidth = 15;    { real field width.                     }
    precision  = 9;     { real precision.                       }
    zeros     = -1;     { indication that the polynomial is zero }
    epslon    = 1E-9;
    upbound   = 1E+9;

TYPE
    Xdim      = 1 .. maxdim;
    Zdim      = 0 .. maxdeg;
    Poly_mat  = ARRAY [ Xdim,Xdim,Zdim] oF Real;
    order     = Array [ Xdim,Xdim    ] oF Integer;
    matrix    = Array [ Xdim,Xdim    ] oF real;
    Vector    = Array [ Xdim         ] oF Integer;
    poly      = Array [ Zdim         ] oF real;
    name      = string[20];

VAR  IN1,OUT1 : Text;      { variable device for I/order }
    Inputfile,
    outputfile: String[14]; { I/O file names }

    i,j,k,d,d1                    : integer;
    A,B,G,L,P,Q,R,S,AV,A1,B1      : poly_mat;
    oA,oB,oG,oL,oP,oQ,oR1,oS,oAV,oa1,ob1: order;
    rA,cA,dA,rB,cB,dB,
```

```
    rA1,cA1,dA1,rb1,cb1,db1,
    rG,cG,dG,rl,cL,dL,rP,cP,dP,rQ,cQ,dQ,
    rR,cR,dR,rS,cS,dS,raV,caV,daV,
    leastcol,leastrow,degdiff,leastdeg,rank: integer;
    offdiagNoT0,flag1,flag2: boolean;
    temp,temp1,t0,t1,t2 : real;
{-----------------------------------------------}
Procedure readPM (VAR A: poly_mat; var oa: order;
            var ra,ca,da: integer);
VAR i,j,k : integer;
BEGIN
 readln(IN1,ra,ca,da);
 FoR i:= 1 To ra Do for j:=1 to ca do
 BEGIN
   read(IN1,oA[i,j]);
   FoR k:= 0 To oA[i,j] Do read(IN1,A[i,j,k]);
   readln(IN1);
   FoR k:= (oA[i,j]+1) To maxdeg Do A[i,j,k] := 0.0;
 END;
END; {readPM}
{-------------------------------------------------------}
Procedure writePM(var ra,ca,da : integer;
          var oA: order; var A: poly_mat; title:name);
VAR i,j,k : integer;
BEGIN
 writeln(out1);
 writeln(out1,' Dimension of ',title,' : ',
            ra:1width,ca:1width,da:1width);

 FoR i:= 1 To ra Do
 BEGIN
  writeln(out1);writeln(out1,'  Row# ',i:2); writeln(out1);
  FoR j:= 1 To ca Do
  BEGIN
   write(out1,oA[i,j]:2,' : ');
     FoR k:= 0 To oa[i,j] Do
     write(out1,A[i,j,k]:fieldwidth:precision);
     writeln(out1);
 END;
 END;writeln(out1);
END;
{-------------------------------------------------------------}
Procedure EvaluateDeg(var ra,ca,maxda: integer;
            var A: poly_mat; var oa: order; var da: integer);
{This procedure evaluates the degrees of the individual elements}
{of a polynomial matrix A}

VAR i,j,k,d,d1 : integer;
BEGIN
  da := zeros;
  { evaluate the drees of A }
```

```
  d1 := maxda+1;
  FoR i:= 1 To ra Do for j:=1 to ca do
  BEGIN
    k:=d1;
    Repeat k:=k-1 UNTIL ((abs(A[i,j,k])>epslon) oR (k=0));
      IF (k>0) THEN oA[i,j] := k
            ELSE IF abs(A[i,j,k])<epslon THEN oA[i,j] := zeros
                                         ELSE oA[i,j] := k;
      IF (oA[i,j]>da) THEN da := oA[i,j];
  END;
END; {evalaluatedeg}
{-------------------------------------------------------------}
Procedure NegativePM(var ra,ca,da : integer;var a:poly_mat);
VAR i,j,k,d : integer;
BEGIN
  FoR i:= 1 To ra Do for j:=1 to ca do
    FoR d:= 0 To da Do A[i,j,d]:=-A[i,j,d];
END; {NegativePM}
{-----------------------------------------------------}
Procedure copyPM(var ra,ca,da: integer; { dimension of A }
          var oA      : order;      { drees of A.    }
          VAR A,B     : poly_mat;   { output matrix. }
          VAR oB      : order ;     { drees of B.    }
          VAR rb,cb,db : integer);   { dimension of B. }
VAR i,j,k,d : integer;
BEGIN
    rb:=ra; cb:=ca;  db:=da;
    FoR i:= 1 To ra Do
      FoR j:= 1 To ca Do
      BEGIN
          oB[i,j]:=oA[i,j];
          FoR d:= 0 To da Do B[i,j,d]:=A[i,j,d];
      END;
END; {copyPM}
{-------------------------------------------------------------}
Procedure transposePM(VAR ra,ca,da: integer; { dimension of A }
                VAR oA      : order;  { drees of A    }
                VAR A       : poly_mat);{ input matrix. }

VAR  B    : poly_mat; { copy   of A  }
     oB   : order;    { degrees of B }
     i,j,d: integer;
BEGIN
    FoR i:=1 To ca Do
      FoR j:=1 To ra Do
      BEGIN
          oB[i,j]:=oA[j,i];
          FoR d:=0 To da Do B[i,j,d]:=A[j,i,d];
      END;
    copyPM(ca,ra,da,oB,B,A,oA,i,j,d);
```

```
    ra:=i; ca:=j;
END;{transposePM}
{-------------------------------------------------------------}
Procedure AddPM(var ra,ca,da,db : integer;  { dimension of A, B }
          var A,B,c          : poly_mat;  { input matrices.  }
          VAR oC             : order;     { drees of C.      }
          VAR rowc,colc,degC  : integer);  { max. deg. of C   }
VAR i,j,k,d : integer;
BEGIN
    IF da < db
    Then BEGIN
          k:=da+1;
          FoR i:=1 To ra Do
          FoR j:=1 To ca Do
          BEGIN
            FoR d:=0 To da Do C[i,j,d]:=A[i,j,d]+B[i,j,d];
            FoR d:=k To db Do C[i,j,d]:=B[i,j,d];
          END;
          k:=db;
        END
    Else BEGIN
          k:=db+1;
          FoR i:=1 To ra Do
          FoR j:=1 To ca Do
          BEGIN
            FoR d:=0 To db Do C[i,j,d]:=A[i,j,d]+B[i,j,d];
            FoR d:=k To da Do C[i,j,d]:=A[i,j,d];
          END;
          k:=da;
        END; rowc:=ra; colc:=ca;
    EvaluateDeg(ra,ca,k,C,oC,degC);
END;{addPM}
{-------------------------------------------------------------}
PROCEDURE MultPM(var ra,ca,cb,da,db :integer; {dimension of A & B }
              var oA,oB   : order ;  {drees   of A & B }
              var A,B,c   : poly_mat; {input matrices   }
              VAR oc      : order;    {drees of dd      }
              VAR rc,cc,dc: integer); { max. deg. of DD }
{This procedure multiplies two polynomial matrices A and B }
{to produce a polynomial matrix C }

 VAR i,j,k,d,d1,k1 : integer;
 BEGIN
    rc:=ra; cc:=cb;
    { initialize DD }   k1:=da+db;
    FoR i:=1 to ra Do
       FoR j:=1 To cb Do
          FoR  d1:=  0 To k1 Do c[i,j,d1]:=0.0;

    {start multiplying }
    FoR i:=1 To ra Do
```

```
      FoR j:=1 To cb Do
        FoR k := 1 To ca Do
          IF ((oA[i,k] < > zeros) AND (oB[k,j] < > zeros)) THEN
          FoR d1:=0 To oA[i,k] Do
            FoR d :=0 To oB[k,j] Do
              c[i,j,d+d1] := c[i,j,d+d1]+A[i,k,d1]*B[k,j,d];
    EvaluateDeg(ra,cb,k1,c,oc,dc);
END; {multPM}
{-------------------------------------------}
procedure RGCDPM(var ra,ca,rb,da,db: integer ;
          var oA,oB              : order  ;
          VAR a,b,G,L,P,Q,R,S    : poly_mat;
          VAR oG,oL,oP,oq,or1,os :order;
          VAR rg,cg,rl,cl,rp,cp,
             rq,cq,rr,cr,rs,cs,
             dg,dl,dp,dq,dr,ds,rank:integer);
{This procedure evaluates the greatest common right divisor G
of two polynomial matrices A and B }

var av : ARRAY[1..RGCDx,1..RGCDy,Zdim] oF real;
   ab : array[1..rgcdx,xdim,zdim] of real;
   c  : array[1..rgcdx,xdim,zdim] of real;
   oab: array[1..rgcdx,xdim] of integer;
   oc : array[1..rgcdx,xdim] of integer;
   oav: array[1..rgcdx,1..rgcdy]      of integer;
   i,j,k,d,d1: integer;
   ii,jj,dd,d2,iii,k1: integer;
BEGIN
  raV := ra+rb;
  caV := ca+raV;

  { initialize AV and G,P,Q,L,R,S to zeros }
  FoR i:= 1 To maxdim Do
    FoR j:= 1 To maxdim Do
    BEGIN
      FoR d:=0 To maxdeg Do
      BEGIN
        G[i,j,d] :=0.0; l[i,j,d]:=0.0; p[i,j,d]:=0.0;
        q[i,j,d] :=0.0; r[i,j,d]:=0.0; r[i,j,d]:=0.0;
      END;
      oG[i,j] :=-1; ol[i,j]:=-1; op[i,j]:=-1;
      oq[i,j] :=-1; os[i,j]:=-1; or1[i,j]:=-1;
    END;

  { initialize AV }
  FoR i:=1 To raV Do
  BEGIN
    FoR j:=1 To caV Do
    BEGIN
      FoR d:= 0 To maxdeg Do  AV[i,j,d]:=0.0;
      oAV[i,j]:=zeros;
```

```
    END;
    AV[i,i+ca,0]:=1.0;
    oAV[i,i+ca]:=0;
  END;

  {copy A & B to AV }
  FoR  j:=1 To ca Do
  BEGIN
    FoR i:=1 To ra Do
    BEGIN
      FoR d:= 0 To oA[i,j] Do av[i,j,d]:=a[i,j,d];
      oAV[i,j]:=oA[i,j];
    END;
    FoR i:=1 To rb Do
    BEGIN
      FoR d:= 0 To oB[i,j] Do AV[i+ra,j,d]:=B[i,j,d];
      oAV[i+ra,j]:=oB[i,j];
    END;
  END;



  k:=1; t1:=1E+30; t2:=1E-30; ii:=0;
  daV := zeros;
  WHILE ((k<raV) AND (k<=ca)) Do
  BEGIN
    offDiagNoT0 := true ;
    WHILE offDiagNoT0 Do
    BEGIN
      { determine position of leastdree nonzero-poly in col k }
      leastrow := zeros;
      leastdeg := maxdeg+1;
      FoR i:=k To raV Do
      IF ((oAV[i,k]<leastdeg) AND (oAV[i,k]<>zeros)) THEN
      BEGIN
        leastdeg := oAV[i,k];
        leastrow := i;
      END;

      {check if the matrix has zero row}
      IF leastrow = zeros
      THEN  offDiagNoT0:=false
      ELSE  BEGIN
            { interchange rs if leastrow <> k }
            IF leastrow <> k THEN
{begin
writeln('interchange rows: ',leastrow:5,k:5);}
            FoR j:=k to caV Do
            BEGIN
              i:= oAV[k,j];
              oAV[k,j]:=oAV[leastrow,j];
```

```
                oAV[leastrow,j]:=i;
                FoR d:=0 To maxdeg Do
                BEGIN
                  Temp:=AV[k,j,d];
                  AV[k,j,d]:=AV[leastrow,j,d];
                  AV[leastrow,j,d]:=temp;
                END;
              END;

{ debug stat}
{for ii:=1 to rav do
begin
 writeln(' Row= ',ii:2);
 for jj:=1 to cav do
 begin
   write(oav[ii,jj]:2,' ');
   for dd:=0 to oav[ii,jj] do write(av[ii,jj,dd]:10:5);
   writeln;
 end;
 writeln;
end; readln;
end;}
              {check if AV(i,j)=0 for i>j }
              FoR j:=k+1 To raV Do
              offdiagNoT0 := offDiagNoT0 AND (oAV[j,k]=zeros);
              offdiagNoT0 := NoT offdiagNoT0;

              IF offdiagNoT0 THEN
              { row operation }
              FoR i:= k+1 To raV Do
              IF oAV[i,k] <> zeros THEN
              BEGIN
                temp:= AV[i,k,oAV[i,k]] / AV[k,k,oAV[k,k]];
                {temp:= AV[i,k,oAV[i,k]];
                temp1:=av[k,k,oav[k,k]];}
                degdiff:=oAV[i,k]-oAV[k,k];
                {for j:=ra+1 to cav do
                 for d:=0 to oav[i,j] do
                   av[i,j,d]:=av[i,j,d]*temp1;}
                FoR j:=k To caV Do
                IF oAV[k,j]<>zeros THEN
                BEGIN

                  {for d:=0 to oav[i,j] do
                    AV[i,j,d]:=temp1*AV[i,j,d];}
                  IF oAV[k,j]<>zeros THEN
                  FoR d:=degdiff To (degdiff+oAV[k,j]) Do
                  BEGIN
                    AV[i,j,d]:=AV[i,j,d]-
                              AV[k,j,d-degdiff]*temp;
```

```
                    if abs(av[i,j,d]) < epslon then av[i,j,d]:= 0.0;

                  END;
                  {calculate dree of AV(i,j)}
                  d:= maxdeg + 1;
                  Repeat d:= d-1 UNTIL ((AV[i,j,d] < > 0.0)oR(d= 0));
                  IF d> 0 THEN oAV[i,j]:= d
                       ELSE IF AV[i,j,d]= 0.0
                            THEN oAV[i,j]:= zeros
                            ELSE oAV[i,j]:= d;
                  IF daV < oAV[i,j] THEN daV:= oAV[i,j];
                END;


{for ii:= 1 to rav do
begin
  writeln(' Row= ',ii:2);
  for jj:= 1 to cav do
  begin
    write(oav[ii,jj]:2,' ');
    for dd:= 0 to oav[ii,jj] do write(av[ii,jj,dd]:10:6,' '); writeln;
    if oav[i,jj]> 3 then
    begin
      write(' ');
      for dd:= 4 to 6 do write(av[i,jj,dd],' '); writeln;
      if oav[i,jj] > 6 then
      begin
        write(' ');
        for dd:= 7 to oav[i,jj] do write(av[i,jj,dd],' ');
        writeln;
      end;
    end;
  end;
  writeln;
end; readln; }



              END;
            END;{else}
        END;{while}
        k:= k + 1;
    END;{while}

{for ii:= 1 to rav do
begin
  writeln(' Row= ',ii:2);
  for jj:= 1 to cav do
  begin
    write(oav[ii,jj]:2,' ');
    for dd:= 0 to oav[ii,jj] do write(av[ii,jj,dd]:10:5);
```

```
    writeln;
  end;
  writeln;
end;}

  { evaluate rank }
  rank:=0;
  Repeat rank:=rank+1 UNTIL ((oAV[rank,rank]=zeros)oR(rank=ca));
  IF oAV[rank,rank]=zeros THEN rank:=rank-1;

  {extract G,L,P,Q,R and S}
  cg := ca;  rg := ca;      dg := 0;
  cl := ca;  rl := raV-rank;
  cp := ra;  rp := ca;      dp := 0;
  cq := rb;  rq := ca;      dq := 0;
  cr := ra;  rr := rl;      dr := 0;
  cs := rb;  rs := rl;      ds := 0;

  { extract G,Q,P,Q,R,and S fron AV }
  k:=ca+cp;
  FoR i:= 1 To rank Do
  BEGIN
    FoR j:= 1 To cg Do
    BEGIN
      oG[i,j]:=oAV[i,j];
      FoR d:= 0 To oG[i,j] Do G[i,j,d]:=AV[i,j,d];
      IF dg<oG[i,j] THEN dg:=oG[i,j];
    END;
    FoR j:= 1 To cp Do
    BEGIN
      oP[i,j]:=oAV[i,j+ca];
      FoR d:= 0 To oP[i,j] Do P[i,j,d]:=AV[i,j+ca,d];
      IF dp<oP[i,j] THEN dp:=oP[i,j];
    END;
    FoR j:= 1 To cq Do
    BEGIN
      oQ[i,j]:=oAV[i,j+k];
      FoR d:= 0 To oQ[i,j] Do Q[i,j,d]:= AV[i,j+k,d];
      IF dq<oQ[i,j] THEN dq:=oQ[i,j];
    END;
  END;

    FoR i:= 1 To rr Do
    BEGIN
        FoR j:= 1 To cr Do
        BEGIN
            oR1[i,j]:=oAV[i+rank,j+ca];
            FoR d:=0 To oR1[i,j] Do R[i,j,d]:=AV[i+rank,j+ca,d];
            IF dr<oR1[i,j] THEN dr:=oR1[i,j];
        END;
        FoR j:=1 To cs Do
```

```
        BEGIN
            oS[i,j]:=oAV[i+rank,j+K];
            FoR d:=0 To oS[i,j] Do S[i,j,d]:=AV[i+rank,j+k,d];
            IF ds<oS[i,j] THEN ds:=oS[i,j];
        END;
    END;
    MultPM(rr,cr,ca,dr,da,oR1,oA,R,A,L,oL,rl,cl,dl);

  {find max ele. in the ith row of r,
  then divide the ith row of r and s}
  {for i:=1 to rr do
  begin
    temp:=-1;
    for j:= 1 to cr do
     for d:=0 to or1[i,j] do if abs(r[i,j,d])>temp
       then temp:=r[i,j,d];
    for j:=1 to cr do
    begin
      for d:=0 to or1[i,j] do r[i,j,d]:=r[i,j,d]/temp;
      for d:=0 to os[i,j]  do s[i,j,d]:=s[i,j,d]/temp;
    end;
  end;}

END;{RGCDPM}
{-----------------------------------------------}
  procedure LGCDPM (var ra,ca,cb,da,db   : integer ;
                VAR  oA,oB            : order  ;
                VAR  A,B,G,L,P,Q,R,S   : poly_mat;
                VAR  oG,oL,oP,oQ,oR1,oS: order;
                VAR  rg,cg,rl,cl,rp,cp,
                     rq,cq,rr,cr,rs,cs,
                     dg,dl,dp,dq,
                     dr,ds,rank        : integer);
{This procedure evaluates the greatest common left divisor G
of two polynomial matrices A and B }

VAR     A1,B1                  : poly_mat;
        oA1,oB1          : order;
        ra1,ca1,rb1,cb1 : integer;
BEGIN
    copyPM(ra,ca,da,oA,A,A1,oA1,ra1,ca1,dg);
    copyPM(ra,cb,db,oB,B,B1,oB1,rb1,cb1,dg);

    transposePM(ra1,ca1,da,oA1,A1);
    transposePM(rb1,cb1,db,oB1,B1);
{   writePM(ra1,ca1,da1,oa1,a1,'A1');
   writepm(rb1,cb1,db1,ob1,b1,'B1');}

    RGCDPM(ra1,ca1,rb1,da,db,oA1,oB1,A1,B1,
         G,L,P,Q,R,S,oG,oL,oP,oQ,oR1,oS,
         rg,cg,rl,cl,rp,cp,rq,cq,rr,cr,
```

```
        rs,cs,dg,dl,dp,dq,dr,ds,rank);
{  writePM(rR,cR,dR,oR1,R,'r');
   writepm(rS,cS,dS,oS,S,'s');}

    transposePM(rg,cg,dg,oG,G);
    transposePM(rl,cl,dl,oL,L);
    transposePM(rp,cp,dp,oP,P);
    transposePM(rq,cq,dq,oQ,Q);
    transposePM(rr,cr,dr,oR1,R);
    transposePM(rs,cs,ds,oS,S);

END;{LGCDPM}
{-------------------------------------------------------------------}
procedure InversePM (var ra,da: integer; { dimension of A        }
                     var  oA    : order;   { drees of A          }
                     var  A,u,v : poly_mat; { matrix A and its inverse}
                     VAR  ov,ou : order ;  { drees of the inverse V }
                     VAR  dV,du : integer); { max deg of V        }
{This procedure evaluates the inverse of a polynomial matrix A     }

var av : ARRAY[1..INVRx,1..INVRy,Zdim] oF real;
   oAV: array[1..invrx,1..invry]      of integer;
   Temp1,Temp2  : Array[zdim] oF real ;
   InversExists : boolean;
   i,j,k,d,d1,ra1,da1: integer;
BEGIN
  ra1 := ra+1;
  raV := ra+ra;
  da1 := da+1;

  {copy A & V to AV }
  FoR  j:=1 To ra Do
  BEGIN
    FoR d:=0 To da Do
    BEGIN
      FoR i:=1 To ra Do AV[i,j,d]:=A[i,j,d];
      FoR i:= ra1 To raV Do AV[i,j,d]:=0.0;
      for i:=1 to ra do u[i,j,d]:=0.0;
    END;
    FoR d:=da1 To maxdeg Do FoR i:=1 To raV Do AV[i,j,d]:=0.0;
    FoR i:= 1 To ra Do oAV[i,j]:=oA[i,j];
    FoR i:= ra1 To raV Do oAV[i,j]:=zeros;
    AV[ra+j,j,0]:=1.0;
    oAV[ra+j,j]:=0;
    FoR i:=1 To ra Do ou[i,j]:=-1;
  END;

  { lower triangularize A }
  inversExists := true ;
  k:=1;
  daV := zeros;
```

```
WHILE (InversExists AND (k < ra)) Do
BEGIN
  offDiagNoT0 := true ;
  WHILE offDiagNoT0 AND Inversexists Do
  BEGIN
    { determine position of leastdree nonzero-poly in row k }
    leastcol := zeros;
    leastdeg := maxdeg + 1;
    FoR j:= k To ra Do
    IF ((oAV[k,j] < leastdeg) AND (oAV[k,j] < > zeros)) THEN
    BEGIN
      leastdeg := oAV[k,j];
      leastcol := j;
    END;

    {check if the matrix has zero row}
    IF leastcol = zeros
    THEN BEGIN
          writeln('ERRoR : Singular Matrix .');
          inversExists: = false;
        END
    ELSE BEGIN
          { interchange columns is leastcol < > k }
          IF leastcol < > k THEN
          FoR i:= k to raV Do
          BEGIN
            j:= oAV[i,k];
            oAV[i,k]: = oAV[i,leastcol];
            oAV[i,leastcol]: = j;
            FoR d:= 0 To maxdeg Do
            BEGIN
              Temp: = AV[i,k,d];
              AV[i,k,d]: = AV[i,leastcol,d];
              AV[i,leastcol,d]: = temp;
            END;
          END;

          {check if AV(i,j) = 0 for i < j }
          FoR i:= k + 1 To ra Do
          offdiagNoT0 := offDiagNoT0 AND (oAV[k,i] = zeros);
          offdiagNoT0 := NoT offdiagNoT0;

          IF offdiagNoT0 THEN
          { column operation }
          FoR j:= k + 1 To ra Do
          IF oAV[k,j] < > zeros THEN
          BEGIN
            temp:= AV[k,j,oAV[k,j]] / AV[k,k,oAV[k,k]];
            degdiff: = oAV[k,j]-oAV[k,k];
            FoR i:= k To raV Do
            IF oAV[i,k] < > zeros THEN
```

```
                BEGIN
                  FoR d:=degdiff To (degdiff+oAV[i,k]) Do
                  BEGIN
                    AV[i,j,d]:=AV[i,j,d]-temp*AV[i,k,d-degdiff];
                    IF abs(AV[i,j,d])<epslon THEN AV[i,j,d]:=0.0;
                  END;

                  {calculate dree of AV(i,j)}
                  d:=maxdeg+1;
                  Repeat d:=d-1 UNTIL ((AV[i,j,d]<>0.0)oR(d=0));
                  IF d>0 THEN oAV[i,j]:=d
                       ELSE IF AV[i,j,d]=0.0
                            THEN oAV[i,j]:=zeros
                            ELSE oAV[i,j]:=d;
                  IF daV<oAV[i,j] THEN daV:=oAV[i,j];
                END;
              END;
            END;{else}
          END;{while}
  k:=k+1;
END;{while}
dV := daV;

{copy lower part of AV to V }
FoR i:=1 To ra Do
  FoR j := 1 To ra Do
  BEGIN
    oV[i,j] := oAV[i+ra,j];
    FoR d:=0 To maxdeg Do  V[i,j,d]:=AV[i+ra,j,d];
  END;

{ make off_diagonal elements AV(i,j) zeros; for i<j }
k:=ra;
WHILE k>1 Do
BEGIN
  FoR j:=1 To (k-1) Do
  IF oAV[k,j]<>zeros THEN
  BEGIN
    { update the diadonal elements of AV }
    FoR d:=0 To maxdeg Do  Temp1[d]:=0.0;
    IF oAV[j,j] <>zeros THEN
    FoR d:=0 To oAV[j,j] Do
      FoR d1:=0 to oAV[k,k] Do
       Temp1[d+d1]:=Temp1[d+d1]+AV[j,j,d]*AV[k,k,d1];
    FoR d:=0 To maxdeg Do IF abs(temp1[d])<epslon
    THEN AV[j,j,d] := 0.0
    ELSE AV[j,j,d] := temp1[d];

    d := maxdeg+1;
    REPEAT d:=d-1 UNTIL ((AV[j,j,d]<>0.0) oR (d=0));
    IF d>0 THEN oAV[j,j]:=d
```

```
            ELSE IF AV[j,j,d]=0.0 THEN oAV[j,j]:=zeros
                                  ELSE oAV[j,j]:=d;

        { update elements of V }
        FoR i:=1 To ra Do
        BEGIN
          FoR d:=0 To maxdeg Do  BEGIN
                                  Temp1[d]:=0.0;
                                  Temp2[d]:=0.0;
                                 END;
          IF oV[i,j] <> zeros THEN
          FoR d:=0 To oV[i,j] Do
          FoR d1:=0 To oAV[k,k] Do
           Temp1[d+d1] := Temp1[d+d1]+V[i,j,d]*AV[k,k,d1];

          IF oV[i,k] <> zeros THEN
           FoR d:=0 To oV[i,k] Do
            FoR d1:= 0 To oAV[k,j] Do
             Temp2[d+d1]:=Temp2[d+d1]+V[i,k,d]*AV[k,j,d1];

          FoR d:=0 To maxdeg Do IF abs(temp1[d]-temp2[d]) < epslon
          THEN V[i,j,d]:=0.0
          ELSE V[i,j,d]:= temp1[d]-temp2[d];

          d := maxdeg+1;
          REPEAT d:=d-1 UNTIL ((V[i,j,d] <> 0.0) oR (d=0));
          IF d>0 THEN oV[i,j]:=d
                ELSE IF V[i,j,d]=0.0 THEN oV[i,j]:=zeros
                                     ELSE oV[i,j]:=d;
          IF dV<oV[i,j] THEN dV:=oV[i,j];
        END;
      END;
      k:=k-1;
END;{while}

    { evaluate u }
    du:= zeros;
    FoR i := 1 To ra Do
    BEGIN
        FoR d := 0 To oAV[i,i] Do u[i,i,d] := AV[i,i,d];
        d:=maxdeg+1;
        REPEAT d:=d-1 UNTIL ((u[i,i,d] <> 0.0) oR (d=0));
        IF d>0 THEN ou[i,i]:=d
              ELSE IF u[i,i,d]=0.0 THEN ou[i,i]:=zeros
                                   ELSE ou[i,i]:=d;
        IF du<ou[i,i] THEN du:=ou[i,i];
    END;

END; {InversePM }
{=========================================}
BEGIN
```

```
    { open I/O files }
    {write(' Enter input file : ');
    readln(inputfile);
    write(' Enter output file : ');
    readln(outputfile);}
    assign(IN1,'inpm.dat');
    assign(out1,'outpm.dat');
    reset(IN1);
    rewrite(out1);

  { read left matrix fraction of the plant}
  readPM(A1,oA1,rA1,cA1,dA1);
  readPM(B1,oB1,rB1,cB1,dB1);

  { evaluate the right coprime matrix fraction (-R,S) }
  LGCDPM(rA1,cA1,CB1,dA1,dB1,oA1,oB1,A1,B1,G,A,P,Q,R,S,oG,oA,oP,oQ,
     oR1,oS,rG,cG,rA,cA,rP,cP,rQ,cQ,rR,cR,rS,cS,dG,dA,dP,dQ,dR,dS,rank);
  negativePM(rr,cr,dr,r);

  writepm(rS,cS,dS,oS,S,'s');
  writePM(rR,cR,dR,oR1,R,'r');

  {check the result }
  negativePM(rr,cr,dr,r);
  multpm(ra1,ca1,cr,da1,dr,oa1,or1,a1,r,q,oq,rq,cq,dq);
  multpm(rb1,cb1,cs,db1,ds,ob1,os ,b1,s,p,op,rp,cp,dp);
  writepm(rq,cq,dq,oq,q,'q');
  writePM(rp,cp,dp,op,p,'p');
  addpm(rq,cq,dq,dp,q,p,g,og,rg,cg,dg);
  writePM(rg,cg,dg,og,g,'g');

   close(IN1);
   close(oUT1);

END. {GCD}
```

Appendix F. Program Listings for Minimization Algorithms

```
program min;
const
    nv = 14;      { number of variables          }
    mc = 5;       { number of constraints        }
    ec = 3;       { num. of equality constraints }
    epsl = 1E-5;  { line search tolerance        }
    epsi = 1E-7;  { uncontrained solution toler. }
    epso = 1e-6;  { max. const. violation toler. }
    no0  = 50;    { max. num. of outer iterations}
    ni0  = 40;    { max. num. of inner iterations}
    nl0  = 50;    { max. num. of line search ite.}
    nl1  = 1000;  { max. num. of bracketting ite.}
   maxDim= 2;     { upper bound on No. of rows and columns }
   maxDeg= 2;     { max. deg. of polys.                    }
   zeros = -1;    { indication that the polynomial is zero }
   eps   = 1E-12;

type vector = array[1..nv]      of real;
    vector1= array[1..mc]      of real;
    vector2= array[0..mc]      of real;
    matrix = array[1..nv,1..nv] of real;
    matrix1= array[1..nv,1..mc] of real;
    poly   = array[0..maxdeg]  of real;
    Poly_mat= ARRAY[1..maxdim,1..maxdim,0..maxdeg] OF Real;
    order  = ARRAY[1..maxdim,1..maxdim] of integer;
    name   = string[20];

var i,j    : integer;
   fm      : real;
   x1      : vector;
   lam,seg : vector1;
   fac     : vector2;
   d,n,u,v : poly_mat;
   p,q,pb,dba:poly_mat;
   op,oq,opb,odba:order;
   rp,cp,dp,rq,cq,dq,rpb,cpb,dpb,rdba,cdba,ddba:integer;
   od,on,ou,ov: order;
   pp      : poly;
   rd,cd,dd,rn,cn,dn,ru,cu,du,rv,cv,dv,dpp:integer;
   in1     : text;
{-------------------------------------------------------}
Procedure EvalDegp(var p:poly; var n:integer);
{This procedure evaluates the degree of a polynomial p }
```

```
VAR k:integer;  flag:boolean;
BEGIN
  k:=maxdeg;  flag:=true;
  while (k>=0) and flag do if abs(p[k])<eps then k:=k-1
                                    else flag:=false;
  if flag then n:=-1 else n:=k;
END; {evaldegp}
{-------------------------------------------------------}
procedure multp(n1,n2:integer; p1,p2:poly; var n3:integer;
            var p3: poly);
{This procedure multiplies two polynomials to produce p3}

var i,j:integer;
begin
 if (n1=-1) or (n2=-1)then n3:=-1 else
 begin
  n3:=n1+n2;
  for i:=0 to n3 do p3[i]:=0.0;
  for i:=0 to n1 do
  for j:=0 to n2 do p3[i+j]:=p3[i+j]+p1[i]*p2[j];
 end;
end;
{-------------------------------------------------------}
PROCEDURE detpm(var rA,dA: integer; var oA: order;
            var A: poly_mat;
            var det: poly; var ndet:integer);
{This procedure evaluates the determinant of a polynomial
matrix A. The valve of the determinant is in det        }

VAR i,j,k,d,d1,i1,j1,k1:integer;  temp:real;
   av : poly_mat;
   oav: order;
   p1 : poly;
   leastcol,leastdeg,degdiff:integer;
   flag,zerorow:boolean;
BEGIN
  { initialize AV }
  FOR i:=1 TO ra DO for j:=1 to ra do
  begin
    for d:=0 to maxdeg do AV[i,j,d]:=0.0;
    for d:=0 to da do AV[i,j,d]:=a[i,j,d];
    oav[i,j]:=oa[i,j];
  END;
  for i:=0 to maxdeg do p1[i]:=0.0;

{writeln;
for i1:=1 to ra do
for j1:=1 to ra do
begin
  write(oav[i1,j1],' : ');
  for k1:=0 to  oav[i1,j1] do write(av[i1,j1,k1]:8:4);
```

```
  writeln;
end; writeln;}

  k:= 1;
  zerorow:= true ;
  WHILE zerorow and (k < ra) do
  BEGIN

  { determine position of leastdegree nonzero-poly in row k }
  leastcol := zeros;
  leastdeg := maxdeg+ 1;
  FOR j:= k TO ra DO
   IF ((oAV[k,j] < leastdeg) AND (oAV[k,j] < > zeros)) THEN
   BEGIN
     leastdeg := oAV[k,j];
     leastcol := j;
   END;

  {check if the matrix has zero row}
  IF leastcol = zeros
    THEN zerorow:= false
    ELSE BEGIN
         { interchange columns if leastcol < > k }
         IF leastcol < > k THEN
           FOR i:= k to ra DO
           BEGIN
            j:= oAV[i,k];
            oAV[i,k]:= oAV[i,leastcol];
            oAV[i,leastcol]:= j;
            FOR d:= 0 TO maxdeg DO
            BEGIN
              Temp:= AV[i,k,d];
              AV[i,k,d]:= AV[i,leastcol,d];
              AV[i,leastcol,d]:= temp;
            END;
           END;

         {check if AV(i,j)= 0 for i < j }
         flag:= true;
         FOR i:= k+ 1 TO ra DO
         flag:= flag and (oAV[k,i]= zeros);

         if flag then k:= k+ 1 else
         { column operation }
         FOR j:= k+ 1 TO ra DO
          IF oAV[k,j] < > zeros THEN
          BEGIN
            temp:= AV[k,j,oAV[k,j]] / AV[k,k,oAV[k,k]];
            degdiff:= oAV[k,j]-oAV[k,k];
            FOR i:= k TO ra DO
            IF oAV[i,k] < > zeros THEN
```

```
            BEGIN
              FOR d:=degdiff TO (degdiff+oAV[i,k]) DO
                BEGIN
                  AV[i,j,d]:=AV[i,j,d]-temp*AV[i,k,d-degdiff];
                  if abs(av[i,j,d])<eps then av[i,j,d]:=0.0;
                END;

              {calculate degree of AV(i,j)}
              d:=maxdeg+1;
              Repeat d:=d-1 UNTIL ((AV[i,j,d]<>0.0)OR(d=0));
                IF d>0 THEN oAV[i,j]:=d
                     ELSE IF AV[i,j,d]=0.0
                          THEN oAV[i,j]:=zeros
                          ELSE oAV[i,j]:=d;
            END;
          END;
      END;{else}

{writeln;
writeln(zerorow,'  k=',k:2);
for i1:=1 to ra do
for j1:=1 to ra do
begin
  write(oav[i1,j1],' : ');
  for k1:=0 to  oav[i1,j1] do write(av[i1,j1,k1]:8:4);
  writeln;
end; writeln;}
   END;{while}


{writeln;
for i:=1 to ra do
for j:=1 to ra do
begin
  write(oav[i,j],' : ');
  for k:=0 to  oav[i,j] do write(av[i,j,k]:8:4);
  writeln;
end; writeln;}

   {evaluate detA}
   if zerorow then
   begin
   ndet:=oav[1,1]; for i:=0 to oav[1,1] do det[i]:=av[1,1,i];
   for i:=2 to ra do
   begin
     k:=oav[i,i]; for j:=0 to oav[i,i] do p1[j]:=av[i,i,j];
     multp(ndet,k,det,p1,ndet,det);
   end;
   evaldegp(det,ndet);
   for i:=0 to ndet do det[i]:=det[i]/det[ndet];
   end
```

```
   else ndet:=-1;
END;{detpm}
{-------------------------------------------------------------------}
Procedure EvaluateDeg(ra,ca,maxda: integer;
            var A: poly_mat; var oa: order; var da: integer);
{This procedure evaluates the degrees of the individual elements}
{of a polynomial matrix a.                                        }

VAR i,j,k,d,d1 : integer;
BEGIN
  da := zeros;
  { evaluate the degrees of A }
  d1 := maxda+1;
  FOR i:= 1 TO ra DO for j:=1 to ca do
  BEGIN
    k:=d1;
    Repeat k:=k-1 UNTIL ((abs(A[i,j,k])> eps) OR (k=0));
      IF (k>0) THEN oA[i,j] := k
            ELSE IF abs(A[i,j,k])<eps THEN oA[i,j] := zeros
                                      ELSE oA[i,j] := k;
      IF (oA[i,j]> da) THEN da := oA[i,j];
    END;
END; {evalaluatedeg}
{-----------------------------------------------------}
Procedure readPM (VAR A: poly_mat; var oa: order;
            var ra,ca,da: integer);
VAR i,j,k : integer;
BEGIN
 read(IN1,ra,ca,da); readln(in1);
 FOR i:= 1 TO ra DO for j:=1 to ca do
 BEGIN
   read(IN1,oA[i,j]);
   FOR k:= 0 TO oA[i,j] DO read(IN1,A[i,j,k]);
   readln(IN1);
   FOR k:= (oA[i,j]+1) TO maxdeg DO A[i,j,k] := 0.0;
 END;
END; {readPM}
{-----------------------------------------------------}
Procedure writePM(var ra,ca,da : integer;
          var oA: order; var A: poly_mat; title:name);
VAR i,j,k : integer;
BEGIN
 writeln;
 writeln(' Dimension of ',title,' : ',
       ra:3,ca:3,da:3);

 FOR i:= 1 TO ra DO
 BEGIN
  writeln;writeln('  Row# ',i:3);
  FOR j:= 1 TO ca DO
  BEGIN
```

```
    write(oA[i,j]:3,'  :    ');
    FOR k:= 0 TO oA[i,j] DO write(A[i,j,k]:9:4);
    writeln;
   END;
  END;
END; {writepm}
{-------------------------------------------------------}
Procedure NegativePM(var ra,ca,da : integer;var a:poly_mat);
VAR i,j,k,d : integer;
BEGIN
  FOR i:= 1 TO ra DO for j:=1 to ca do
    FOR d:= 0 TO da DO A[i,j,d]:=-A[i,j,d];
END; {NegativePM}
{-------------------------------------------------------------}
Procedure AddPM(var ra,ca,da,db : integer;  { dimen. of a,b  }
          var A,B,c          : poly_mat;  { input matrices. }
          VAR oC             : order;     { degrees Of C.  }
          VAR rc,cc,dc       : integer);  { max. deg. of C  }
{This procedure adds two poly. matrices a and b toproduce c   }

VAR i,j,k,d : integer;
BEGIN
   IF da < db
   Then BEGIN
        k:=da+1;
        FOR i:=1 TO ra DO
        FOR j:=1 TO ca DO
        BEGIN
          FOR d:=0 TO da DO C[i,j,d]:=A[i,j,d]+B[i,j,d];
          FOR d:=k TO db DO C[i,j,d]:=B[i,j,d];
        END;
        k:=db;
       END
   Else BEGIN
        k:=db+1;
        FOR i:=1 TO ra DO
        FOR j:=1 TO ca DO
        BEGIN
          FOR d:=0 TO db DO C[i,j,d]:=A[i,j,d]+B[i,j,d];
          FOR d:=k TO da DO C[i,j,d]:=A[i,j,d];
        END;
        k:=da;
       END;  rc:=ra; cc:=ca;
   EvaluateDeg(ra,ca,k,C,oC,dc);
END;{addPM}
{-----------------------------------------------------------------}
PROCEDURE MultPM(ra,ca,cb,da,db :integer; var oa,ob: order;
            var a,b,c     : poly_mat; var oc : order;
            var rc,cc,dc  : integer);
{This procedure multiplies two polynomial matrices a,b of }
{dimensions ra,ca and rb,cb. The result is a poly. matrix }
```

```
{c of dimension rc,cc. The integer matrices : oa,ob,oc    }
{are the degrees of the polynomials of a,b,c. The integers}
{da,db,dc are the max. degree of the polys of a,b,c.      }

 VAR i,j,j1,k,d,d1,k1 : integer;

 BEGIN
     rc:=ra; cc:=cb;
     {initialize c}   k1:=da+db;
     FOR i:=1 to ra DO
        FOR j:=1 TO cb DO
           FOR  d1:= 0 TO k1 DO  c[i,j,d1] := 0.0;

     {start multiplying }
     FOR i:=1 TO ra DO
       FOR j:=1 TO cb DO
         FOR k := 1 TO ca DO
           IF (oA[i,k]<>zeros) AND (oB[k,j]<>zeros) THEN
           begin
             FOR d1:=0 TO oa[i,k] do
              for d:=0 to ob[k,j] do
              c[i,j,d+d1]:=c[i,j,d+d1]+a[i,k,d1]*b[k,j,d];
           end;
     EvaluateDeg(ra,cb,k1,c,oc,dc);
 END; {multPM}
{-------------------------------------------------------}
procedure copyv(var d1,d2: vector);
{ This procedure copies vector d1 to d2      }
var i:integer;
begin for i:=1 to nv do d2[i]:=d1[i]; end; {copyv}
{-------------------------------------------------------}
procedure multmv(var s:matrix; var g,d:vector);
{ This procedure multiplies a matrix s by a vector g to give}
{ a vector d                                              }
var i,j:integer;
begin
  for i:=1 to nv do
  begin
    d[i]:=0.0; for j:=1 to nv do d[i]:=d[i]+s[i,j]*g[j];
  end;
end;{multmv}
{-------------------------------------------------------}
procedure multvm(var q:vector; var s:matrix;var sq:vector);
{ This procedure multiplies a row vector q by matrix s to   }
{ give a row vector sq                                      }
var i,j:integer;
begin
 for j:=1 to nv do
 begin
   sq[j]:=0.0;
   for i:=1 to nv do sq[j]:=sq[j]+q[i]*s[i,j];
```

```
  end;
end; {multvm}
{-------------------------------------------------------}
procedure multvvs(var p1,p2:vector; var t:real);
{ This procedure multiplies row and col. vectors to give    }
{ a scaler t.                                               }
var i:integer;
begin
  t:=0.0;  for i:=1 to nv do t:=t+p1[i]*p2[i];
end;{multvvs}
{-------------------------------------------------------}
procedure multvvm(var p1,p2:vector; var r:matrix);
{ This procedure multiplies row and col. vecrors to give    }
{ a matrix r.                                               }
var i,j:integer;
begin
  for i:=1 to nv do for j:=1 to nv do r[i,j]:=p1[i]*p2[j];
end;{multvvm}
{-------------------------------------------------------}
procedure maxabs(var p:vector; var w1:real);
{ This procedure evaluates the maximum component }
{ of the vector p in absolute value.             }
var i :integer;
begin
  w1:=-1.0;
  for i:=1 to nv do if abs(p[i])>w1 then w1:=abs(p[i]);
end;{maxabs}
{-------------------------------------------------------}
procedure evalfg(var x:vector;var f:real; var g:vector1);
{ This procedure evaluates the Obj. func. and constraints   }
{ at x.  f is obj. func. value. g  is vector of cons. values}

var y1,y2,y3,y4,y5,y6,y7,y8,y9 : real;
    r1,r2,r3,r4,r5,r6,r7,r8    : real;
    a,a1,a2,a3     : poly_mat;
    oa,oa1,oa2,oa3 : order;
    pp,p0,p1   : poly;
    i,j,ra,ca,da,ra1,ca1,da1,
    ra2,ca2,da2,dpp,dp0,dp1,
    ra3,ca3,da3   :integer;
begin

   { example 3.1 cons. as functions, two complex poles, order 0
   y1:=x[1]-0.5;  y2:=x[2];
   f:=y1*y1+y2*y2;
   g[1]:=x[3]*(x[4]+x[6])-1.0;
   g[2]:=0.5*x[3]*(x[4]-x[6])+x[5]*(5.0*x[4]-2.0*x[6])-x[1]-x[1];
   g[3]:=-x[3]*(3.0*x[4]+x[6])+x[5]*(x[4]-8.0*x[6])-(x[1]*x[1]+x[2]*x[2]);
   g[4]:=x[1];}

   { example 3.1 as above but two real poles
```

```
y1:=x[1]-0.5;  y2:=x[2]-1.0;
f:=y1*y1+y2*y2;
y1:=x[3]*(x[4]+x[6]);
g[1]:=y1-1.0;
g[2]:=0.5*x[3]*(x[4]-x[6])+x[5]*(5.0*x[4]-2.0*x[6])-(x[1]+x[2]);
g[3]:=-x[3]*(3.0*x[4]+x[6])+x[5]*(x[4]-8.0*x[6])-x[1]*x[2];
g[4]:=x[1];
g[5]:=x[2];}

{ example 3.1 const. from detR(s)
y1:=x[1]-0.5;  y2:=x[2]-0.5;
f:=y1*y1+y2*y2;
y1:=x[3]*(x[4]+x[6]);
g[1]:=y1-1.0;
u[1,1,0]:=x[3];
u[2,2,0]:=x[4];
v[1,1,0]:=x[5];
v[2,2,0]:=x[6];
multpm(ru,cu,cd,du,dd,ou,od,u,d,a1,oa1,ra1,ca1,da1);
multpm(rv,cv,cn,dv,dn,ov,on,v,n,a2,oa2,ra2,ca2,da2);
addpm(ra1,ca1,da1,da2,a1,a2,a,oa,ra,ca,da);
detpm(ra,da,oa,a,pp,dpp);
g[2]:=pp[1]*y1-x[1]-x[2];
g[3]:=pp[0]*y1-x[1]*x[2];
g[4]:=x[1]-0.25;
g[5]:=x[2]-0.25;}

{ example 3.1 const. from detR(s), feedforward, order 0
y1:=x[1]-1.0;  y2:=x[2]-0.5;
f:=y1*y1+y2*y2;
y2:=x[12]+x[14]; y3:=x[11]+x[13];
y4:=x[4]*x[10];  y5:=x[3]*x[9];
y1:=(x[6]*y2+y4*dba[2,2,1])*(x[5]*y3+y5*dba[1,1,1])-
   (x[6]*y3+y4*dba[2,1,1])*(x[5]*y2+y5*dba[1,2,1]);

g[1]:=y1-1.0;
u[1,1,0]:=x[3];
u[2,2,0]:=x[4];
v[1,1,0]:=x[5];
v[2,2,0]:=x[6];
p[1,1,0]:=x[7];
p[2,2,0]:=x[8];
q[1,1,0]:=x[9];
q[2,2,0]:=x[10];
pb[1,1,0]:=x[11];
pb[1,2,0]:=x[12];
pb[2,1,0]:=x[13];
pb[2,2,0]:=x[14];
dba[1,1,0]:=(x[11]-0.5*x[13])/x[7];
dba[1,1,1]:=x[11]/x[7];
dba[1,2,0]:=(x[12]-0.5*x[14])/x[7];
```

```
dba[1,2,1]:=x[12]/x[7];
dba[2,1,0]:=(-4.0*x[11]-x[13])/x[8];
dba[2,1,1]:=(x[11]+x[13])/x[8];
dba[2,2,0]:=(-4.0*x[12]-x[14])/x[8];
dba[2,2,1]:=(x[12]+x[14])/x[8];
multpm(ru,cu,cq,du,dq,ou,oq,u,q,a1,oa1,ra1,ca1,da1);
multpm(ra1,ca1,cdba,da1,ddba,oa1,odba,a1,dba,a2,oa2,ra2,ca2,da2);
multpm(rv,cv,cn,dv,dn,ov,on,v,n,a1,oa1,ra1,ca1,da1);
multpm(ra1,ca1,cpb,da1,dpb,oa1,opb,a1,pb,a3,oa3,ra3,ca3,da3);
addpm(ra2,ca2,da2,da3,a2,a3,a,oa,ra,ca,da);
detpm(ra,da,oa,a,pp,dpp);
g[2]:=pp[1]*y1-x[1]-x[2];
g[3]:=pp[0]*y1-x[1]*x[2];
g[4]:=x[1];
g[5]:=x[2];
writeln(g[1]:12:6,g[2]:12:6,g[3]:12:6,g[4]:12:6); }

{ example 3.1 const. from detR(s), feedforward, order 0}
y1:=x[1]-1.0;  y2:=x[2]-0.5;
f:=y1*y1+y2*y2;
y2:=(x[11]-0.5*x[13])/x[7];
y1:=x[11]/x[7];
y4:=(x[12]-0.5*x[14])/x[7];
y3:=x[12]/x[7];
y6:=(-4.0*x[11]-x[13])/x[8];
y5:=(x[11]+x[13])/x[8];
y8:=(-4.0*x[12]-x[14])/x[8];
y7:=(x[12]+x[14])/x[8];
r1:=x[3]*x[9]*y1+x[5]*(x[11]+x[13]);
r2:=x[3]*x[9]*y2+x[5]*(3.0*x[11]+x[13]);
r3:=x[3]*x[9]*y3+x[5]*(x[12]+x[14]);
r4:=x[3]*x[9]*y4+x[5]*(3.0*x[12]+x[14]);
r5:=x[4]*x[10]*y5+x[6]*(x[11]+x[13]);
r6:=x[4]*x[10]*y6+2.0*x[6]*(x[11]-x[13]);
r7:=x[4]*x[10]*y7+x[6]*(x[12]+x[14]);
r8:=x[4]*x[10]*y8+2.0*x[6]*(x[12]-x[14]);
g[1]:=r1*r7-r3*r5-1;
g[2]:=r1*r8+r2*r7-(r3*r6+r4*r5)-x[1]-x[2];
g[3]:=r2*r8-r4*r6-x[1]*x[2];
g[4]:=x[1];
g[5]:=x[2];
{writeln(g[1]:12:6,g[2]:12:6,g[3]:12:6,g[4]:12:6);writeln;}

{ example 3.1 const. from equations, order 1, three real poles
y1:=x[1]-1.0;  y2:=x[2]-0.5; y3:=x[3]-0.3;
f:=y1*y1+y2*y2+y3*y3;
y1:=x[4]*(x[6]+x[9]);
g[1]:=y1-1.0;
g[2]:=x[4]*(-x[6]-2.0*x[9])+x[7]*(3.0*x[6]-4.0*x[9])+
      (1.5*x[4]+x[5]+2.0*x[7])*(x[6]+x[9])-(x[1]+x[2]+x[3]);
g[3]:=(x[4]+x[5]+3.0*x[7]+x[8])*(-x[6]-2.0*x[9])-
```

```
        (-0.5*x[4]+x[7]+x[8])*(-4.0*x[6]+2.0*x[9])+
        (1.5*x[5]+2.0*x[8])*(x[6]+x[9])-(x[1]*x[2]+x[1]*x[3]+x[2]*x[3]);
    g[4]:=(x[5]+3.0*x[8])*(-x[6]-2.0*x[9])-
        (-0.5*x[5]+x[8])*(-4.0*x[6]+2.0*x[9])-x[1]*x[2]*x[3];
    g[5]:=x[1];
    g[6]:=x[2];
    g[7]:=x[3];}

    { example 3.1 const. from detR(s), order 1, three real poles
    y1:=x[1]-1.0;  y2:=x[2]-0.5; y3:=x[3]-0.3;
    f:=y1*y1+y2*y2+y3*y3;
    y1:=x[4]*(x[6]+x[9]);
    g[1]:=y1-1.0;
    u[1,1,1]:=x[4];
    u[1,1,0]:=x[5];
    u[2,2,0]:=x[6];
    v[1,1,1]:=x[7];
    v[1,1,0]:=x[8];
    v[2,2,0]:=x[9];
    multpm(ru,cu,cd,du,dd,ou,od,u,d,a1,oa1,ra1,ca1,da1);
    multpm(rv,cv,cn,dv,dn,ov,on,v,n,a2,oa2,ra2,ca2,da2);
    addpm(ra1,ca1,da1,da2,a1,a2,a,oa,ra,ca,da);}
    {writepm(ru,cu,du,ou,u,'u');
    writepm(rv,cv,dv,ov,v,'v');
    writepm(rd,cd,dd,od,d,'d');
    writepm(rn,cn,dn,on,n,'n');
    writepm(ra1,ca1,da1,oa1,a1,'a1');
    writepm(ra2,ca2,da2,oa2,a2,'a2');
    writepm(ra,ca,da,oa,a,'a');}
    {detpm(ra,da,oa,a,pp,dpp);
    p0[0]:=x[1]; p0[1]:=1.0; dp0:=1;
    p1[1]:=1.0;  dp1:=1;
    for i:=2 to 3 do
    begin
      p1[0]:=x[i]; multp(dp0,dp1,p0,p1,dp0,p0);
    end;
    g[2]:=pp[2]*y1-p0[2];
    g[3]:=pp[1]*y1-p0[1];
    g[4]:=pp[0]*y1-p0[0];
    g[5]:=x[1];
    g[6]:=x[2];
    g[7]:=x[3];}
    {writeln(g[1]:12:6,g[2]:12:6,g[3]:12:6,g[4]:12:6); writeln;}

    f:=f*fac[0];
    for j:=1 to mc do g[j]:=g[j]*fac[j];

end;{evalfg}
{-------------------------------------------------------------}
procedure dervfg(var x,g:vector; var dg:matrix1);
{ This procedure evaluates the gradients of the obj. fun.  }
```

```
{ and the constraints.                                    }
{ g  : gradient of obg. func.                             }
{ dg : the jth col. of the matrix dg is the gradient of the }
{      jth constraint                                     }

var y1,y2,t,t1,h,
    f1,f2,f3,f4 : real;
    i,j         : integer;
    g1,g2,g3,g4 : vector1;
    x1          : vector;
begin

   { Example 3.1
   g[1]:=2.0*(x[1]-0.5); g[2]:=2.0*(x[2]-1.0); g[3]:=2.0*x[3];
   for i:=4 to nv do g[i]:=0.0;

   for i:=1 to 3 do dg[i,1]:=0.0; dg[4,1]:=x[5]+x[7];
   dg[5,1]:=x[4]; dg[6,1]:=0.0; dg[7,1]:=x[4];

   dg[1,2]:=-1.0; dg[2,2]:=-1.0; dg[3,2]:=0.0;
   dg[4,2]:=0.5*(x[5]-x[7]); dg[5,2]:=0.5*x[4]+5.0*x[6];
   dg[6,2]:=5.0*x[5]-2.0*x[7]; dg[7,2]:=-0.5*x[4]-2.0*x[6];

   dg[1,3]:=-x[2]; dg[2,3]:=-x[1]; dg[3,3]:=-2.0*x[3];
   dg[4,3]:=-(3.0*x[5]+x[7]);   dg[5,3]:=-3.0*x[4]+x[6];
   dg[6,3]:=x[5]-8.0*x[7];      dg[7,3]:=-x[4]-8.0*x[6];

   dg[1,4]:=x[3]; dg[2,4]:=-x[3]; dg[3,4]:=x[1]-x[2];
   for i:=4 to nv do dg[i,4]:=0.0;

   dg[1,5]:=1.0; for i:=2 to nv do dg[i,5]:=0.0;
   dg[1,6]:=0.0; dg[2,6]:=1.0; for i:=3 to nv do dg[i,6]:=0.0;}


   {using finite diff. to find derivatives - three point formula}
   h:=0.01;  t:=50.0;
   for i:=1 to nv do x1[i]:=x[i];
   for i:=1 to nv do
   begin
     x1[i]:=x1[i]+h;
     evalfg(x1,f1,g1);
     x1[i]:=x1[i]-h-h;
     evalfg(x1,f2,g2);
     g[i]:=t*(f1-f2);
     for j:=1 to mc do dg[i,j]:=t*(g1[j]-g2[j]);
     x1[i]:=x1[i]+h;
   end;


   {using finite diff. to find derivatives - five point formula}
   {h:=0.01;  t:=8.3333333333333334; t1:=66.6666666666666667;
```

```
  for i:=1 to nv do x1[i]:=x[i];
  for i:=1 to nv do
  begin
    x1[i]:=x1[i]+h;
    evalfg(x1,f1,g1);
    x1[i]:=x1[i]+h;
    evalfg(x1,f3,g3);
    x1[i]:=x1[i]-h-h-h;
    evalfg(x1,f2,g2);
    x1[i]:=x1[i]-h;
    evalfg(x1,f4,g4);
    g[i]:=t1*(f1-f2)-t*(f3-f4);
    for j:=1 to mc do dg[i,j]:=t1*(g1[j]-g2[j])-t*(g3[j]-g4[j]);
    x1[i]:=x1[i]+h+h;
  end;}


end;{dervfg}
{-------------------------------------------------------}
procedure scale(var x:vector);
{This procedure evaluates scales factors for the objective }
{function and the constraints}

var i,j,k :integer;
    f,t   :real;
    g     :vector1;
begin
  evalfg(x,f,g);
  t:=abs(f); if t>0 then fac[0]:=1.0/t;
  for j:=1 to mc do
    begin t:=abs(g[j]); if t>0 then fac[j]:=1.0/t;end;
  writeln('Scale factors:');
  for j:=0 to mc do writeln('factor(',j:1,')',fac[j]);
end;{scale}
{-------------------------------------------------------}
procedure maxConstViol(var x1:vector;
                var f:real; var g,gh:vector1; var a1:real);
{ This procedure evaluates the max. constraint violation    }
{ in absolute value.                                        }
var i:integer;
begin
  evalfg(x1,f,g);
  for i:=1 to ec do gh[i]:=abs(g[i]);
  for i:=ec+1 to mc do if g[i]<0 then gh[i]:=abs(g[i])
                                 else gh[i]:=0.0;
  a1:=-1.0;
  for i:=1 to mc do if gh[i]>a1 then a1:=gh[i];
end; {maxConstViol}
{-------------------------------------------------------}
procedure la(var x,d:vector;var g:vector1; var x1,l,f:real);
```

```
{ This function evaluates the aug. Lag. at : x-x1*d        }
var i     : integer;
    t     : real;
    y     : vector;
begin
  for i:=1 to nv do y[i]:=x[i]-x1*d[i];
  evalfg(y,f,g);
  l:=f;
  for i:=1 to ec do l:=l+(0.5*seg[i]*g[i]-lam[i])*g[i];
  for i:=ec+1 to mc do
  begin
    t:=seg[i]*g[i];
    if (lam[i]>=2.0*t)
    then l:=l+(t-lam[i])*g[i]
    else l:=l-(lam[i]*lam[i])/(4.0*seg[i])
  end;
end; {La}
{-------------------------------------------------------}
procedure gradla(var x,d:vector);

{ This procedure evaluates the gradient vector of the      }
{ Lagrangian function. x is the variables vector and       }
{ d is the gradient vector.                                }

var i,j : integer;
    g1  : vector1;
    dg  : matrix1;
    f,t : real;
begin
  dervfg(x,d,dg);
  evalfg(x,f,g1);

  {write('d =');for i:=1 to nv do write(d[i]); writeln;}
  {for i:=1 to nv do
  begin
    write(d[i]:10:3);
    for j:=1 to mc do write(dg[i,j]:10:3);
    writeln;
  end;
  writeln;
  write(f:10:3);
  for j:=1 to mc do write(g1[j]:10:3);
  writeln;}

  for j:=1 to ec do
  begin
    t:=lam[j]-seg[j]*g1[j];
    for i:=1 to nv do d[i]:=d[i]-t*dg[i,j];
  end;
  for j:=ec+1 to mc do
  begin
```

```
    t:=lam[j]-2.0*seg[j]*g1[j];
    if t>0.0 then for i:=1 to nv do d[i]:=d[i]-t*dg[i,j];
   end;
   {write('dl=');for i:=1 to nv do write(d[i]); writeln;}

end; {gradla}
{-------------------------------------------------------}
procedure solve(n: integer; var a:matrix; var h,x:vector;
                var solution: boolean);
{This procedure solves a linear system of equations:  ax=h  }

 var i,j,k,n1    : integer;
    w          : matrix1;
    temp,temp1,eps  : real;
 begin
    {  save w =  a|h }
    n1:=n+1;  eps:=1E-12;
    for i:=  1 to n do
    begin
      x[i]:=0.0;
      for j:=1 to n do w[i,j]:=a[i,j];  w[i,n1]:=h[i];
    end;

    { Gaussian elemination row operations }
    solution:=true; j:=0;
    while (j<n) and solution do
    begin
      j:=j+1;
      { partial pivoting search for |max| ele. in col. j }
      k:=j;
      temp:=abs(w[j,j]);
      for i:=j+1 to n do
      begin
        temp1:=abs(w[i,j]);
        if temp<temp1 then begin  temp:=temp1; k:=i;  end;
      end;

      { check if the matrix has full rank }
      if temp=0.0
      then solution:=false
      else begin
        { interchange rows }
        if k<>j then
        for i:=1 to n1 do
        begin
          temp:=w[j,i]; w[j,i]:=w[k,i]; w[k,i]:=temp;
        end;

        { upper triangularize }
        for i:=  j+1 to n do
        begin
```

```
          if w[i,j] < > 0.0 then
          begin
           temp: = w[i,j]/w[j,j];
           for k: = j + 1{j} to n1 do
           if w[j,k] < > 0.0 then begin
                              w[i,k]: = w[i,k]-temp*w[j,k];
                              if abs(w[i,k]) < eps then w[i,k]: = 0.0;
                            end;
         end;
        end;
      end;{else}
    end;{while}

    { evaluate the x's }
    if solution then
    for i: = n downto 1 do
    begin
      x[i]: = w[i,n1];
      for j: = n downto i + 1 do x[i]: = x[i]-x[j]*w[i,j];
      x[i]: = x[i]/w[i,i];
      if abs(x[i]) < eps then x[i]: = 0.0;
    end     else writeln(' ??? Singular matrix ');
end;{solve}
{-------------------------------------------------------------}
procedure LineSearch(var x,d:vector;var x1:real);
{ This procedure performs minimization of a function in one }
{ variable. It min. the aug. lag. la(x-x1*d) with respect  }
{ to x1. The procedure utilizes a Quadratic Fit (QF)       }
{ algorithm to find the min. Before the QF iterations the  }
{ procedure performs bracketting to find three points:     }
{ x1,x2 and x3, such that la(x1),la(x2) and la(x3) form    }
{ concave shape.                                           }

var k,k1,i,j       : integer;
   x2,x3,xm,y1,y2,y3,ym,
   w1,h,f,l,t      : real;
   solution        : boolean;
   b,c             : vector;
   g               : vector1;
   a               : matrix;
   flag            : boolean;
begin
  h: = 0.1; t: = 1.2;
  x1: = 1.0; x2: = 1.0; x3: = 1.0;
  { bracketting process}
  la(x,d,g,x1,y1,f);
  {writeln('  alfa = ',x1:10:5,'  La = ',y1:16:8,f:16:8);}
  h: = h*t;
  x2: = x1 + h;
  la(x,d,g,x2,y2,f);
  {writeln('  alfa = ',x2:10:5,'  La = ',y2:16:8,f:16:8); }
```

```
k1:=0;
if y1>y2
then begin
       x3:=x2+h;
       la(x,d,g,x3,y3,f);
       {writeln('  alfa= ',x3:10:5,'  La= ',y3:16:8,f:16:8);}
       while ((y2>=y3) and (k1<nl1)) do
       begin
         x1:=x2; y1:=y2; x2:=x3; y2:=y3;
         h:=h*t; x3:=x3+h;
         la(x,d,g,x3,y3,f);
         k1:=k1+1;
         {for i:=1 to nv do writeln(x[i],d[i]);}
         {writeln('  alfa= ',x3:10:5,'  La= ',y3:16:8,f:16:8);}
       end;
    end
else begin
       x3:=x2; y3:=y2; x2:=x1; y2:=y1; x1:=x1-h;
       la(x,d,g,x1,y1,f);
       {writeln('  alfa= ',x1:10:5,'  La= ',y1:16:8,f:16:8);}
       while ((y1<=y2) and (k1<nl1)) do
       begin
         x3:=x2; y3:=y2; x2:=x1; y2:=y1;
         h:=h*t; x1:=x1-h;
         la(x,d,g,x1,y1,f);
         {for i:=1 to nv do writeln(x[i],d[i]);}
         {writeln('  alfa= ',x1:10:5,'  La= ',y1:16:8,f:16:8);}
         k1:=k1+1;
       end;
    end;
    {writeln('x1,x2,x3=',x1:14:6,x2:14:6,x3:14:6);}

{fitting Quadratic poly.}
k:=0;  w1:=1.0;  flag:=true;
while ((k<nl0) and (w1>epsl) and flag) do
begin
  k:=k+1;
  a[1,1]:=1.0; a[1,2]:=x1; a[1,3]:=x1*x1; b[1]:=y1;
  a[2,1]:=1.0; a[2,2]:=x2; a[2,3]:=x2*x2; b[2]:=y2;
  a[3,1]:=1.0; a[3,2]:=x3; a[3,3]:=x3*x3; b[3]:=y3;
  solve(3,a,b,c,solution);
  if c[3]=0.0 then begin xm:=x2; flag:=false; end else
  begin
    xm:=-c[2]/(2.0*c[3]);
    la(x,d,g,xm,ym,f);
    w1:=abs(c[1]+c[2]*xm+c[3]*xm*xm-ym);
    if xm>x2 then begin if ym<y2
                  then begin
                         x1:=x2; x2:=xm; y1:=y2; y2:=ym;
                       end
                  else begin x3:=xm; y3:=ym; end;
```

```
                end
            else begin if ym < y2
                  then begin
                       x3:=x2; x2:=xm; y3:=y2; y2:=ym;
                       end
                  else begin x1:=xm; y1:=ym; end;
                end;
    end;{else}
  end;{while}
  x1:=xm;
  {if k=n0 then writeln('Number of iterations exceeded')
        else writeln('Minimum= ',x1:10:5);}
end; {LineSearch}
{-------------------------------------------------------------}
procedure BFGS_Quasi_N(var x1,x2:vector;var s1:matrix);
{ This procedure solves the unconstrained minimization of   }
{ the augmented Lagrangian function with respect to X.      }
{ A quasi Newton method is used which utilizes the Davidon_ }
{ Fletcher_Powel update of the inverse of the Hessian.      }
{ x1,x2: initial and final approximation of the solution.   }
{ lam,seg: maltipliers and penalty vectors                  }
{ s1 : approx. of the inverse of the Hessian.               }

var i,j,k              :integer;
    w1,t1,t2,alf,f,l   : real;
    g1,g2,q,p,d,sq,qs  : vector;
    r1,r2              : matrix;
    g                  : vector1;
begin
  k:=0; w1:=1.0;  alf:=0.5;

  {write headings and first iteration}
  for i:=1 to nv do d[i]:=0.0;
  la(x1,d,g,alf,l,f);
  write('k       La        f      ||x||   ');
  for i:=1 to mc do write('  g(',i:1,')   ');
  for i:=1 to nv do write('  x(',i:1,')   ');writeln;
  write(k:1,l:13:6,f:10:6,w1:10:6);
  for i:=1 to mc do write(g[i]:10:6);
  for i:=1 to nv do write(x1[i]:10:6); writeln;

  gradla(x1,g1);
  multmv(s1,g1,d);
  LineSearch(x1,d,alf);
  for i:=1 to nv do p[i]:=-alf*d[i];
  for i:=1 to nv do x2[i]:=x1[i]+p[i];

  {writeln('k=',k:2);
  write('x =');for i:=1 to nv do write(x1[i]); writeln;
  write('s =');for i:=1 to nv do write(s1[1,i]); writeln;
  write('  ');for i:=1 to nv do write(s1[2,i]); writeln;
```

```
  write('g = ');for i:= 1 to nv do write(g1[i]); writeln;
  write('d = ');for i:= 1 to nv do write(d[i]); writeln;
  writeln('alfa= ',alf);
  write('p = ');for i:= 1 to nv do write(p[i]); writeln;
  write('x = ');for i:= 1 to nv do write(x2[i]); writeln;
  writeln;}

  while ((k < ni0) and (w1 > epsi)) do
  begin
    gradla(x2,g2);
    for i:= 1 to nv do q[i]:= g2[i]-g1[i];
    multmv(s1,q,sq);
    multvm(q,s1,qs);
    multvvs(q,sq,t1);
    multvvs(p,q,t2);
    multvvm(sq,qs,r1);
    multvvm(p,p,r2);
    for i:= 1 to nv do for j:= 1 to nv do
    s1[i,j]:= s1[i,j]+r2[i,j]/t2-r1[i,j]/t1;
    multmv(s1,g2,d);
    maxabs(d,w1);
    if w1 > epsi then LineSearch(x1,d,alf);
    for i:= 1 to nv do p[i]:= -alf*d[i];
    for i:= 1 to nv do x2[i]:= x1[i]+p[i];
    la(x1,d,g,alf,l,f);
    maxabs(p,w1);
    k:= k+1;
    copyv(x2,x1);  copyv(g2,g1);

    {writeln('k= ',k:2);
    write('x = ');for i:= 1 to nv do write(x1[i]); writeln;
    write('s = ');for i:= 1 to nv do write(s1[1,i]); writeln;
    write('   ');for i:= 1 to nv do write(s1[2,i]); writeln;
    write('   ');for i:= 1 to nv do write(s1[3,i]); writeln;
    write('q = ');for i:= 1 to nv do write(q[i]); writeln;
    write('g = ');for i:= 1 to nv do write(g1[i]); writeln;
    write('d = ');for i:= 1 to nv do write(d[i]); writeln;
    writeln('alfa= ',alf);
    write('p = ');for i:= 1 to nv do write(p[i]); writeln;
    writeln;}

    write(k:1,l:13:6,f:10:6,w1:10:6);
    for i:= 1 to mc do write(g[i]:10:6);
    for i:= 1 to nv do write(x1[i]:10:6); writeln;
  end; {while}

  write('-----------------------------------------');
  writeln('----------------------------------');
end; {BFGS_Quasi_N}
{-------------------------------------------------------}
procedure alagpf( var x1:vector;var f:real);
```

```
{ This procedure solves a constrained program using the    }
{ augmented Lagrangian function with penalty parameters.   }
{ x1   : solution vector. f is min. value of obj. func.    }
{ lam,seg: multipliers and penalty vectors.                }

var i,j,k  : integer;
w1,a1,a2,t : real;
x2,x3      : vector;
g,gh       : vector1;
s1         : matrix;

begin
    {initialization}
    k:=0;  w1:=1.0;
    for i:=1 to nv do
    begin
      for j:=1 to nv do s1[i,j]:=0.0; s1[i,i]:=1.0;
    end;
    for i:=1 to mc do
    begin
      lam[i]:=0.0; seg[i]:=1.0; fac[i]:=1.0;
    end; fac[0]:=1.0;

    {evaluate scale factors(fac) for the const.}
    scale(x1);
    MaxConstViol(x1,f,g,gh,a1);

    {outer iteration}
    while (((a1>epso) or (w1>epsi)) and (k<no0)) do
    begin
      write('N= ',k:2);
      writeln('    PENALTY FACTORS          Lag. Multipliers');
      for i:=1 to mc do writeln('    ',seg[i],'    ',lam[i]);
      writeln('          Max. const. Viol.= ',a1:14:8);

      {inner iteration}
      copyv(x1,x3);
      BFGS_Quasi_N(x3,x2,s1);
      for i:=1 to nv do x3[i]:=x2[i]-x1[i];
      maxabs(x3,w1);
      MaxConstViol(x2,f,g,gh,a2);
      k:=k+1;

      { update multipliers and penalty vectors}
      for i:=1 to ec do lam[i]:=lam[i]-seg[i]*g[i];
      for i:=ec+1 to mc do
      begin
       t:=lam[i]-2.0*seg[i]*g[i];
       if t>0.0 then lam[i]:=t
              else lam[i]:=0.0;
      end;
```

```
      for i:=1 to mc do seg[i]:=seg[i]*10.0;

      a1:=a2;
      copyv(x2,x1);
  end; {while}
end; {alagpf}
{-------------------------------------------------------}
procedure multp1(n1,n2:integer; p1,p2:poly; var n3:integer;
            var p3: poly);
{ This procedure multiplis two polynomials p1,p2 of degrees
n1,n2 and sets the value in p3 }

var i,j:integer;
begin
 if (n1=-1) or (n2=-1)then n3:=-1 else
 begin
   n3:=n1+n2;
   for i:=0 to n3 do p3[i]:=0.0;
   for i:=0 to n1 do
   for j:=0 to n2 do p3[i+j]:=p3[i+j]+p1[i]*p2[j];
 end;
end;
{-------------------------------------------------------}
begin { main program}
   assign(in1,'ex.dat'); {assign data file}
   reset(in1);

   {read right coprime fraction (n,d) of the plant}
   readpm(d,od,rd,cd,dd);
   readpm(n,on,rn,cn,dn);

   {initialize and set the order of the ocntroller}
   readpm(u,ou,ru,cu,du);
   readpm(v,ov,rv,cv,dv);

   {initialize the following matrices through readpm}
   readpm(p,op,rp,cp,dp);
   readpm(q,oq,rq,cq,dq);
   readpm(pb,opb,rpb,cpb,dpb);
   readpm(dba,odba,rdba,cdba,ddba);

   {Example 3.1}
   for i:=1 to nv do x1[i]:=0.2;

   alagpf(x1,fm);
   writeln;
   writeln('Min. value = ',fm/fac[0]:14:5); writeln;
   writeln('Solution   = ');
   for i:=1 to nv do writeln('             ',x1[i]:14:5);

end. {min}
```

The vita has been removed from the scanned document