CO:22:46 OCA PAD AMENDMENT - PROJECT HEADER INFORMATION 8/1/20/92

Project #: E-19-696  Cost share #: E-19-358  Rev #: 4  Active
Center #: 10/24-6-R6793-0A0  Center shr #: 10/22-1-F6793-0A0
Contract#: CTS-8902534  Mod #: OPAS-NCE
Prime #:  
Subprojects ?: N
Main project #:  

Project unit: CHEM ENGR  Unit code: 02.010.114
Project director(s): ARKUN Y  CHEM ENGR  (404)894-2871

Sponsor/division names: NATL SCIENCE FOUNDATION  /  GENERAL
Sponsor/division codes: 107  /  000

Award period: 890801  to  920331 (performance)  920630 (reports)

Sponsor amount  New this change  Total to date
  Contract value  0.00  80,000.00
  Funded  0.00  80,000.00
Cost sharing amount  6,024.00

Does subcontracting plan apply ?: N

Title: INDUSTRY-UNIVERSITY COOPERATIVE RESEARCH PROGRAM ON A COMPREHENSIVE APPROACH.

PROJECT ADMINISTRATION DATA

OCA contact: Mildred S. Heyser  894-4820
Sponsor technical contact  Sponsor issuing office
MARIA BURKA  (202)357-9606  MATTHEW SMILAK  (202)357-9626
NATIONAL SCIENCE FOUNDATION  NATIONAL SCIENCE FOUNDATION
1800 G STREET, N.W.  1800 G STREET, N.W.
WASHINGTON, D.C. 20550  WASHINGTON, D.C. 20550

Security class (U,C,S,TS) : U
Defense priority rating : N/A
Equipment title vests with: Sponsor
NONE PROPOSED
Administrative comments -
PERFORMANCE ENDING DATE EXTENDED THROUGH 3/31/92 PER OPAS DATED 1/14/92.
GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

Closeout Notice Date 08/05/92

Project No. E-19-696

Project Director ARKUN Y

Center No. 10/24-6-R6793-0A0

School/Lab CHEM ENGR

Sponsor NATL SCIENCE FOUNDATION/GENERAL

Contract/Grant No. CTS-8902534

Contract Entity GTRC

Prime Contract No.

Title INDUSTRY-UNIVERSITY COOPERATIVE RESEARCH PROGRAM ON A COMPREHENSIVE APPRO

Effective Completion Date 920331 (Performance) 920630 (Reports)

Closeout Actions Required:  

<table>
<thead>
<tr>
<th>Final Invoice or Copy of Final Invoice</th>
<th>Y/N</th>
<th>Date Submitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final Report of Inventions and/or Subcontracts</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Government Property Inventory &amp; Related Certificate</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Classified Material Certificate</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Release and Assignment</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

Comments

Subproject Under Main Project No. ____________

Continues Project No. ____________

Distribution Required:

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Director</td>
<td>Y</td>
</tr>
<tr>
<td>Administrative Network Representative</td>
<td>Y</td>
</tr>
<tr>
<td>GTRI Accounting/Grants and Contracts</td>
<td>Y</td>
</tr>
<tr>
<td>Procurement/Supply Services</td>
<td>Y</td>
</tr>
<tr>
<td>Research Property Management</td>
<td>Y</td>
</tr>
<tr>
<td>Research Security Services</td>
<td>N</td>
</tr>
<tr>
<td>Reports Coordinator (OCA)</td>
<td>Y</td>
</tr>
<tr>
<td>GTRC</td>
<td>Y</td>
</tr>
<tr>
<td>Project File</td>
<td>Y</td>
</tr>
<tr>
<td>Other</td>
<td>N</td>
</tr>
</tbody>
</table>
ANNUAL PROGRESS REPORT
ON
INDUSTRY-UNIVERSITY COOPERATIVE RESEARCH ON A
COMPREHENSIVE APPROACH TO CONTROL STRUCTURE
SYNTHESIS FOR DECENTRALIZED CONTROL

GRANT No. CTS-8902534

by

Dr. Yaman Arkun

School of Chemical Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0100
I. PROPOSED GOALS

This research aims at developing a practical theory towards control configuration design for chemical plants. This includes the development of quantitative measures to analyze and screen large sets of alternative control structures, and establishing a general design methodology for decentralized control. Examples from different industries are to be demonstrated in cooperation with Tennessee Eastman and General Electric Companies.

II. RESEARCH STAFF

Dr. Yaman Arkun is currently assisted in this project by two PhD students:

1. Deborah Edwards Reeves

2. Min-Sen Chiu
   Thesis Title: A Methodology for Robust Decentralized Controller Design.

III. Summary of Research Results and Current Research

The work in the area of the development of practical theory for control structure synthesis resulted in the following publications:


The following works are either being submitted or prepared:


Publication [2] is the result of a summer work that Dr. Arkun took with Tennessee Eastman Company as a part of the Industry-University collaboration component of the research. The last three works [6,7,8] involve Dr. Nett from Aerospace Engineering who moved to Tech from General Electric after the grant was funded, and Dr. Minto who is employed by GE and an adjunct faculty at Georgia Tech. Therefore our research continues to be an University-Industry Cooperative program.

The work in the area of a general methodology for decentralized control resulted in the following publications:


Current and future research will focus on finishing the systematic synthesis methodology for configuration design which will encompass nonsquare structures, illustration on industrial examples, and perfecting the robust decentralized controller design methodology.
PART I - PROJECT IDENTIFICATION INFORMATION

1. Program Official/Org.  Dr. Maria Burka

2. Program Name  Chemical and Thermal Systems

3. Award Dates (MM/YY)  From: 8/1/89  To: 4/30/92

4. Institution and Address  School of Chemical Engineering
Georgia Institute of Technology
Atlanta, GA 30332-011

5. Award Number  CTS-8902534

6. Project Title  A Comprehensive Approach to Control Structure Synthesis for Decentralized Control

This Packet Contains
NSF Form 98A
And 1 Return Envelope
The data requested below are important for the development of a statistical profile on the personnel supported by Federal grants. The information on this part is solicited in response to Public Law 99-383 and 42 USC 1885C. All information provided will be treated as confidential and will be safeguarded in accordance with the provisions of the Privacy Act of 1974. You should submit a single copy of this part with each final project report. However, submission of the requested information is not mandatory and is not a precondition of future award(s). Check the “Decline to Provide Information” box below if you do not wish to provide the information.

Please enter the numbers of individuals supported under this grant. Do not enter information for individuals working less than 40 hours in any calendar year.

<table>
<thead>
<tr>
<th>Category</th>
<th>Senior Staff</th>
<th>Post-Doctors</th>
<th>Graduate Students</th>
<th>Under-Graduates</th>
<th>Other Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Total, U.S. Citizens</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>B. Total, Permanent Residents</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>U.S. Citizens or Permanent Residents²</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>American Indian or Alaskan Native</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Asian</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Black, Not of Hispanic Origin</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Hispanic</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Pacific Islander</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>White, Not of Hispanic Origin</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>C. Total, Other Non-U.S. Citizens</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Specify Country</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1. Taiwan</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>3.</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>D. Total, All participants (A + B + C)</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Disabled³</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

☐ Decline to Provide Information: Check box if you do not wish to provide this information (you are still required to return this page along with Parts I-III).

1Category includes, for example, college and precollege teachers, conference and workshop participants.
2Use the category that best describes the ethnic/racial status for all U.S. Citizens and Non-citizens with Permanent Residency. (If more than one category applies, use the one category that most closely reflects the person's recognition in the community.)
3A person having a physical or mental impairment that substantially limits one or more major life activities; who has a record of such impairment; or who is regarded as having such impairment. (Disabled individuals also should be counted under the appropriate ethnic/racial group unless they are classified as “Other Non-U.S. Citizens.”)

AMERICAN INDIAN OR ALASKAN NATIVE: A person having origins in any of the original peoples of North America, and who maintain cultural identification through tribal affiliation or community recognition.

ASIAN: A person having origins in any of the original peoples of East Asia, Southeast Asia and the Indian subcontinent. This area includes, for example, China, India, Indonesia, Japan, Korea and Vietnam.

BLACK, NOT OF HISPANIC ORIGIN: A person having origins in any of the black racial groups of Africa.

HISPANIC: A person of Mexican, Puerto Rican, Cuban, Central or South American or other Spanish culture or origin, regardless of race.

PACIFIC ISLANDER: A person having origins in any of the original peoples of Hawaii; the U.S. Pacific Territories of Guam, American Samoa, or the Northern Marianas; the U.S. Trust Territory of Palau; the islands of Micronesia or Melanesia; or the Philippines.

WHITE, NOT OF HISPANIC ORIGIN: A person having origins in any of the original peoples of Europe, North Africa, or the Middle East.
I. RESEARCH STAFF

Dr. Yaman Arkun, the Principal Investigator, was assisted in this research project by the following graduate students:

Mrs. Deborah Edwards Reeves (PhD 1991),
Mr. Georgios Charos (PhD 1990),
Mr. Min-Sen Chiu (PhD 1991)

II. SUMMARY OF PROJECT

This research aimed at developing a practical theory towards control configuration design for chemical plants. This includes the development of quantitative measures to analyze and screen large sets of alternative control structures, establishing a general design methodology for decentralized control, and development of a CAD software for technology transfer to industry.

III. TECHNICAL DESCRIPTION OF PROJECT AND RESULTS

Thesis completed:

1. Mrs. Deborah Edwards Reeves
   PhD Dissertation Title: A Comprehensive Approach to Control Configuration Design for Complex Systems
2. Mr. Georgios Charos
   PhD Dissertation Title: Model Predictive Constrained Control: Development, Implementation, and Decentralization
3. Min-Sen Chiu
   PhD Dissertation Title: A Methodology for Robust Decentralized Controller Design

Publications:


Publication [2] is the result of a summer work that Dr. Arkun took with Tennessee Eastman Company as a part of the Industry-University collaboration component of the research. The last product of this research is a general purpose Toolbox for control structure synthesis. Dr. Nett is from Aerospace Engineering who moved to Tech from General Electric after the grant was funded. General Electric was the other industrial collaborator in this research. This Toolbox constitutes the CAD software promised in the proposal. It was made available for distribution to industry and universities and announced in the MATLAB Newsletter. Copies of the above publications are attached.
A GENERAL METHOD TO CALCULATE INPUT-OUTPUT GAINS AND THE RELATIVE GAIN ARRAY FOR INTEGRATING PROCESSES

Y. ARKUN† and J. DOWNS

†School of Chemical Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0100, U.S.A.

(Received 12 March 1990; final version received 19 June 1990; received for publication 26 June 1990)

Abstract—A general method based on singular value decomposition and state-space models is presented to calculate a set of gains between inputs and outputs of processes with integrators. The practical importance of these gains for determining input–output sensitivities and asymptotic properties is illustrated along with the calculation of the relative gain array for integrating systems. The new results are particularly applicable to large chemical processes and can be easily incorporated into flowsheet simulators.

1. INTRODUCTION

For processes with integrators the mathematical relationships between certain state variables and inputs are governed by integrators. These processes are also called non-self-regulating in the sense that if a step change is made in one of the inputs, state variables will not reach any new steady-state. In other words the usual steady-state gain is undefined (Seborg et al., 1989). The simplest example is a liquid storage tank shown in Fig. 1. Assuming that the outlet flow can be manipulated independent of the height of the liquid, the mathematical relationship between the height and the inlet flowrates is expressed by an integral. Consequently, as long as there exists a difference between the two flows, the tank will either overflow or run dry, i.e. it will be non-self-regulating. Chemical plants in general contain many material inventory units similar to the example just given which result in integrating or non-self-regulating processes.

Our interest in integrating processes was motivated by an industrial need to calculate "a meaningful gain" between the input and output variables of such processes. One of the objectives was to be able to use these gains in the relative gain array (RGA) analysis to systematically synthesize control structures for large process flowsheets which contained a subset of integrators. Open-loop steady-state gains or transfer functions for these systems cannot be obtained by traditional simulation (e.g. by step response) unless the integrators are identified first and put under closed-loop control. Therefore one usually approaches the control structure synthesis in the following way. First all the integrating state variables are found and defined as controlled variables. Next an equal number of manipulated variables are chosen, and they are assigned to control the integrating variables. This first phase is called synthesis of inventory control loops. Once the inventory loops are closed, the control structure for the remaining self-regulating variables can be constructed using methods like the RGA. However, splitting the synthesis into two phases is suboptimal because assigning certain manipulated variables in the first phase can adversely deteriorate the performance of the controllers to be developed in the second phase. Therefore one may have to iterate between the two phases and the resulting procedure can become very expensive and time consuming.

Recognizing these problems Woolverton (1980) suggested using the derivatives of the integrating state variables as the controlled variables so that steady-state and relative gains could be defined. Later McAvoy (1983) proved that, for a specific 3 x 3 example, the RGA for the integrating variables and their derivatives are the same. Both of these works, although not proven in general, constitute the starting point of our work. Therefore, the examples from Woolverton (1980) and McAvoy (1983) will be revisited here to set the stage for the general method proposed in this paper.

---

†To whom all correspondence should be addressed.

Fig. 1. A liquid storage system.
Method to calculate input-output gains and the RGA

\[ G(0) = \begin{bmatrix} 0 & -1 \\ 0 & 1/2 \end{bmatrix}. \]

Definition 4—In this work input-output gains are defined as the integrator gains \( G_i, i = 1, m \) and the non-integrating gain \( G(0) \). It should be noted that although \( G_i \) is a gain between the output and the integral of the input, due to a lack of a better terminology, we will still classify it as an input-output gain.

### 3. PROBLEM STATEMENT AND THE METHOD TO CALCULATE THE GAINS

We assume that a process flowsheet which contains a certain number of integrating state variables is given and a steady-state operating point is specified along with the inputs and outputs of interest. The mathematical problem is to calculate the input-output gains which are defined as the integrator gains \( G_i, i = 1, m \) and the non-integrating gain \( G(0) \). It should be noted that although \( G_i \) is a gain between the output and the integral of the input, due to a lack of a better terminology, we will still classify it as an input-output gain.

The system matrices \((A, B, C, D)\) are assumed to be available from the usual Jacobian calculation provided by a flowsheet simulator.

When there are integrators, the matrix \( A \) is singular; therefore, the input-output gains cannot be computed from the usual steady-state gain matrix \( G(0) = CA^{-1}B + D \). In Table I some examples of systems with integrators are given. Singularity of \( A \) and existence of integrators is due to zero rows, example (a); zero columns, example (b); combination of zero rows and columns, examples (c) and (d); and dependent non-zero columns or rows of \( A \), example (e). For chemical processes systems of the kind (a–c) with first-order integrators are the most common ones while second and higher order integrators, like the system (d) are not common at all.

We now describe the method used to calculate the gain matrices \( G_i, i = 1, m \) and \( G(0) \) based on the state-space model \((A, B, C, D)\). The results are formally stated for systems with first-order integrators only for the practical reasons mentioned above. However, it should be noted that the method applies to systems with higher order integrators as well. To clarify this point we discuss the necessary changes in the Appendix. The method is based on singular value decomposition (SVD). Every \( n \times n \) matrix \( A \) of rank \( r \) can be written in terms of its SVD (Stewart, 1973):

\[ A = U \Sigma V^T, \]

where \( U \) is an \( n \times n \) orthogonal matrix of the eigenvectors of \( AA^T \), \( V \) is an \( n \times n \) orthogonal matrix of the eigenvectors of \( A^T A \), and \( \Sigma \) is an \( n \times n \) diagonal matrix:

\[ \Sigma = \begin{bmatrix} \Sigma_i & 0 \\ 0 & 0 \end{bmatrix}. \]

The main results can now be stated.

**Result 1**—First-order integrator gain matrix of \( G(s) = C(sI - A)^{-1}B + D \) can be computed from:

\[ G_i = C \begin{bmatrix} I & 0 \\ -U & 0 \end{bmatrix} \begin{bmatrix} I / s & 0 & 1/s(s+1) \\ 0 & 0 & 1/s \\ 0 & 0 & 0 \end{bmatrix} B, \]

where \( V^T U \), is the first \( r \times r \) block of the \( n \times n \) matrix \( V^T U \).

### Table I. Examples of integrating systems

<table>
<thead>
<tr>
<th>( A )</th>
<th>( G(s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{bmatrix} 0 &amp; 0 &amp; 0 \ 1 &amp; -1 &amp; 1 \ 0 &amp; 0 &amp; 0 \end{bmatrix}</td>
<td>\begin{bmatrix} 1/s &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1/s \ 1/s(s+1) &amp; 1/s+1 &amp; 1/(s+1) \end{bmatrix}</td>
</tr>
<tr>
<td>\begin{bmatrix} -0.304 &amp; 0 &amp; -0.38 \ -0.1 &amp; 0 &amp; 0 \ -0.0103 &amp; 0 &amp; -0.054 \end{bmatrix}</td>
<td>\begin{bmatrix} \xi_1(1/s) &amp; 0 &amp; \xi_1(1/s) \ \xi_1(1/s) &amp; \xi_1(1/s) &amp; \xi_1(1/s) \ \xi_1(1/s) &amp; 0 &amp; \xi_2(1/s) \end{bmatrix}</td>
</tr>
<tr>
<td>\begin{bmatrix} 0 &amp; 0 &amp; 0 \ 1 &amp; -1 &amp; 0 \ 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 \end{bmatrix}</td>
<td>\begin{bmatrix} 1/s &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1/s &amp; 0 \ \xi_2(1/s) &amp; 0 &amp; \xi_2(1/s) &amp; 0 \ \xi_2(1/s) &amp; 0 &amp; \xi_2(1/s) &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>\begin{bmatrix} 0 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 1 \ 0 &amp; 0 &amp; 0 \end{bmatrix}</td>
<td>\begin{bmatrix} 1/s &amp; 1/s^2 &amp; 1/s^3 \ 0 &amp; 0 &amp; 1/s^2 \ 0 &amp; 0 &amp; 1/s \end{bmatrix}</td>
</tr>
<tr>
<td>\begin{bmatrix} 1 &amp; 1 \ 2 &amp; 2 \end{bmatrix}</td>
<td>\begin{bmatrix} s - 2/s(s-3) &amp; 1/s(s-3) \ 2/(s-3) &amp; s - 1/s(s-3) \end{bmatrix}</td>
</tr>
</tbody>
</table>

\[ B = C = I, \ D = 0 \] and \( g_i(s) = \begin{bmatrix} \eta_i(s) \\ \eta_i(s) \end{bmatrix} \]
i.e. the integrators are in rows [e.g. see (b) and (c) in Table 1], then,

$$\text{RGA}[G(s)] = \text{RGA} \begin{bmatrix} G_{n1}(s) \\ G_1(s) \end{bmatrix}. \quad (21)$$

It can be also easily proved that if the system belongs to the class:

$$G(s) = \begin{bmatrix} G_{n1}(s) & \frac{1}{s} G_1(s) \end{bmatrix}, \quad G_1 \in \mathbb{R}^{n \times \nu}, \quad (22)$$

i.e. the integrators are in columns [e.g. see (a) and (c) in Table 1], then,

$$\text{RGA}[G(s)] = \text{RGA}[G_{n1}(s)G_1(s)]. \quad (23)$$

The proof is again based on scaling invariance property of RGA but this time with respect to inputs. Note that in (20) and (22) the rows (columns) have been already permuted to separate integrating and non-integrating rows (columns). In the following we first investigate the systems belonging to classes described by equations (20) and (22). Later this is relaxed and the procedure is extended to the general case.

It is clear that for steady-state or “large time” analysis we need the gain matrices $G_{n1}(0)$ and $G_1(0)$. Now it will be shown that these can be obtained from the integrator gain matrix $G_1$ and non-integrating gain matrix $\tilde{G}(0)$ which we know how to compute.

First matrices $G_1$ and $\tilde{G}(0)$ are calculated. The non-zero entries of $G_1$ are identified with the integrators and $G(s)$ is classified as a row (20) or column (22) integrator. Next one constructs a set of numbers $k = [i, j, k \ldots]$ from the number of the rows (columns) of $G_1$ which contain the non-zero entries. One also constructs the complement of this set $\bar{k}$. Then, defining $G(k)$ as the reduced $G$ matrix formed by the rows (columns) whose numbers belong to $k$, one gets:

$$G_1(0) = G_1(k), \quad (24)$$

$$G_{n1}(0) = \tilde{G}(0)(\bar{k}) \quad (25)$$

and

$$\text{RGA}[G(s)](0) = \text{RGA} \begin{bmatrix} \tilde{G}(0)(\bar{k}) \\ G_1(k) \end{bmatrix} \quad \text{or} \quad \text{RGA}[\tilde{G}(0)(\bar{k}) \ G_1(k)]. \quad (26)$$

**Example 4**

Consider Example 3. Inspecting $G_1$, $k = [1, 3]$, $\bar{k} = 2$ and $G_1$ is a column integrator:

$$G_1(k) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix},$$

$$\tilde{G}(0)(\bar{k}) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

$$\text{RGA}[G(s)](0) = \text{RGA} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$
6. PHYSICAL EXAMPLE

Consider the reactor-splitter system shown in Fig. 5. \( F_1 \) is pure A and \( F_2 \) is pure B. The splitter separates \( F_1 \) into \( F_4 \) and \( F_5 \) such that \( F_4 \) is equal to molar flow of A and B in \( F_1 \), and \( F_5 \) is equal to molar flow of C and D in \( F_3 \). The outputs of interest are:

- \( y_1 = X_B \) (mol fraction of component B in reactor),
- \( y_2 \) = inventory (total moles in reactor),
- \( y_3 \) = production rate, \( F_3 \) (mol h\(^{-1}\)).

The manipulated inputs are: \( u_1 = F_1 \), \( u_2 = F_3 \), \( u_3 = F_2 \). The state and output equations, steady-state data and system matrices are given in Table 2.

6.1. Results

Matrix \( A \) is singular and has rank \( r = 3 \). Matrix \( Z_1 = [V^T U] \) is non-singular; therefore, the integrators are first-order. SVD of \( A \) is given in Table 3. The first-order integrator gain matrix is computed from (15):

\[
G_1 = \begin{bmatrix}
-0.05 & 0 & 0.05 \\
-5 & 0 & 5 \\
0 & 0 & 0
\end{bmatrix}
\]

The non-integrating gain matrix is computed from (18):

\[
\tilde{G}(0) = \begin{bmatrix}
-1.2 & 0.16 & 0.8 \\
130 & -16 & -80 \\
6 & 0 & -4
\end{bmatrix}
\]

Inspection of these matrices reveals that there are no pure integrators. Entries (1, 1), (1, 3), (2, 1) and (2, 3) of \( G(s) \) contain first-order integrators plus other stable dynamics. Entries (1, 2), (2, 2), (3, 1), (3, 2) and (3, 3) do not have integrators.

The zero entry (3, 2) of \( \tilde{G}(0) \) suggests that its transfer function can have a zero at the origin. Therefore the gain \([C, R, B] \) (3, 2) is computed and found to be equal to 16, i.e. \( g_{32}(s) = 16s \tilde{g}_{32}(s) \) with \( \tilde{g}_{32}(0) = 1 \).

Now that all the input–output gains are computed.
Using (A2), (A3) becomes:

\[
(sU^TV - \Sigma)^{-1} = \frac{V^TU}{s} - \frac{V^TU}{s} \begin{pmatrix} 0 \\ 1 \end{pmatrix}
\times (sU^TV - \Sigma)^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} V^TU
\]

\[
= \hat{G}(s)
\]  

(A4)

and

\[
\lim_{s \to 0} \left[ \frac{\hat{G}(s) - \frac{G_1}{s}}{s} \right] = -Z_wZ_{11}^{-1}Z_{11}^{-1}Z_w
\]

\[
= -\begin{bmatrix} Z_{11} \\ Z_{21} \end{bmatrix} \begin{bmatrix} Z_{11}^{-1}Z_{11}^{-1}Z_{11}^{-1}Z_{12} \\ Z_{21} \end{bmatrix}
\]

(A20)

2. Modifications for Second and Higher Order Integrators

Assume that \([VTU]\) is singular; then, the previous results cannot be used as given. Therefore we start with (A4) and focus on \((sU^TV - \Sigma)^{-1}\) in which \([VTU]\) is singular.

With first-order integrating systems we expanded the term \((sI - A)^{-1}\) in which \(A\) was singular, now we perform exactly the same calculations for \((sU^TV - \Sigma)^{-1}\). Namealy:

\[
([VTU] - s\Sigma)^{-1} = -V(sU^TV - \Sigma)^{-1}U^T. 
\]

(A23)

\[
[VTU] = U, \Sigma = \begin{bmatrix} \Sigma_0 \\ 0 \end{bmatrix}
\]

\[
\Sigma_0 = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n).
\]

Using the formula (A2) to expand:

\[
(sU^TV - \Sigma)^{-1} = \frac{V^TU}{s} + \frac{V^TU}{s} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} \Sigma_0 \\ 0 \end{pmatrix} V^TU
\]

\[
\times ([VTU] - s\Sigma)^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} V^TU.
\]

(A24)

Note that this expansion is analogous to (A4). Substituting (A24) into (A23) and the resulting expression into the original first-order integrator expansion (A4) gives the complete expansion to second-order integrators:

\[
G(s) = \frac{V^TU}{s} + \frac{V^TU}{s} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} \Sigma_0 \\ 0 \end{pmatrix} V^TU
\]

\[
\times ([VTU] - s\Sigma)^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} V^TU.
\]

(A25)

Then the second-order integrator gain matrix is given by:

\[
G(s) = \frac{V^TU}{s} + \frac{V^TU}{s} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} \Sigma_0 \\ 0 \end{pmatrix} V^TU
\]

\[
\times ([VTU] - s\Sigma)^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} V^TU.
\]

(A26)

(A27)

According to (8), the total expansion to second-order integrators is:

\[
G(s) = G_1 + G_2 + G_3
\]

(A28)

Therefore, having computed \(G_1, G_2, G_3\) follow directly from (A27), i.e.

\[
G_j = \lim_{s \to 0} \left[ \frac{G(s) - G_j}{s} \right]
\]

(A29)

\[
G(0) = \lim_{s \to 0} \left[ \frac{G(s) - G_j}{s} \right]
\]

(A30)
A New Result on Relative Gain Array, Niederlinski Index and Decentralized Stability Condition: 2 × 2 Plant Cases*

MIN-SEN CHIU† and YAMAN ARKUN‡‡

Key Words—Decentralized stability; relative gain array; Niederlinski Index.

Abstract—This note discusses the decentralized stability condition for a 2-channel decentralized control system where each channel is a single-input-single-output (SISO) control loop. Stabilization of each individual loop (with the other loop open) is not required, which had been the case in previous work on the Relative Gain Array (RGA) and the Niederlinski Index (NI). Therefore this new result is a generalization of previous RGA and NI stabilization conditions.

1. Introduction

For a 2-channel decentralized control system where each channel is a single-input-single-output (SISO) control loop, decentralized stability condition can be related to the steady-state gain of the open loop system using the Relative Gain Array (RGA) and the Niederlinski Index (NI) (Grosdidier et al., 1985; Niederlinski, 1971). Both of these quantities have found wide applications in process control in particular. In the derivation of these results (e.g. see Grosdidier et al., 1985) the stability of the individual loop (with the other loop open) is assumed. Although such an assumption can be justified based on practical grounds like reliability, it is not a general theoretical result. Therefore in this work this assumption is relaxed and new stability conditions related to RGA and NI are obtained.

2. Preliminaries

Assumption 1 holds in the following development.

Assumption 1:

• The plant is described by \( G(s) = [g_{ij}(s)] \) for \( i, j = 1, 2 \) and \( G(s) \) is stable, strictly proper or semi-proper.

• Decentralized controller

\[
C(s) = \begin{bmatrix}
\frac{1}{s} c_1(s) \\
\frac{1}{s} c_2(s)
\end{bmatrix}
\]

has two SISO loops and each loop contains integral action.

Under Assumption 1 the decentralized feedback control scheme is shown in Fig. 1. The corresponding decentralized internal model control (IMC) structure is given in Fig. 2 with the following equivalence:

\[
G_m(s) = \text{diag}[g_{i1}(s)] \quad \forall i = 1, 2
\]

\[
Q(s) = \text{diag}[q_{ij}(s)] \quad \forall i = 1, 2
\]

and

\[
q_{ij}(s) = \frac{c_i(s)}{s + g_{ii}(s)c_j(s)} \quad \forall i = 1, 2
\]

where \( q_{ij}(s) \) is allowed to be unstable which implies loop \( i \) (\( i = 1, 2 \)) to be unstable by itself. It is clear from (2) that \( q_{ij}(s) \) has to satisfy

\[
q_{ij}(0) = \frac{1}{g_{ii}(0)}
\]

to ensure the integral action of both loops. Later we make use of this equivalence between the decentralized feedback and IMC structure to prove the results.

The Relative Gain Array (RGA) was developed by Bristol (1966). Given the steady-state gain of \( G(s) \), i.e. \( G(0) = [g_{ij}^0] \) for \( i, j = 1, 2 \) where \( g_{ij}^0 \) is real. RGA is defined as

\[
\Lambda = [\lambda_{ij}] \quad \forall i, j = 1, 2
\]

where

\[
\lambda_{11} = \lambda_{22} = \frac{g_{11}^0 g_{22}^0}{g_{12}^0 g_{21}^0}, \quad \sqrt{\frac{g_{11}^0 g_{22}^0}{g_{12}^0 g_{21}^0}} - \sqrt{\frac{g_{11}^0 g_{22}^0}{g_{12}^0 g_{21}^0}} - \frac{g_{11}^0 - g_{12}^0}{g_{21}^0}
\]

In the \( 2 \times 2 \) plant cases, the Niederlinski Index (NI) (Niederlinski, 1971) is defined to be

\[
\text{NI} = \frac{\text{det}[\Lambda]}{\text{det}^2[\Lambda]} = \left[ \frac{g_{11}^0 g_{22}^0 - g_{12}^0 g_{21}^0}{g_{11}^0 g_{22}^0} \right] = \frac{1}{\lambda_{11}}.
\]

Under the assumption that the individual loop (i.e. loop 1 or 2) is stable when the other loop is open (i.e. loop 2 or 1), it is proved that \( \lambda_{ii} > 0 \) (or \( \text{NI} > 0 \)) is the necessary and sufficient condition for decentralized stability (Grosdidier et al., 1985). However, a priori stabilization of individual loops is not a prerequisite for the stability of the decentralized system. Therefore in this work this requirement is relaxed and the following questions are answered: Is there any restriction on the number of unstable closed-loop poles of loops 1 and 2? If yes, how many unstable closed-loop poles can one assign to loops 1 and 2?

3. Stability conditions

The next proposition states the decentralized stability condition for a \( 2 \times 2 \) plant. Denote

\[
\Delta(s) = 1 - g_{12}(s)g_{21}(s)g_{11}(s)q_{11}(s)
\]

Proposition 1. Let Assumption 1 hold. The necessary and sufficient condition for decentralized stability is that \( \Delta(s) \) does not contain any RHP-zeros.

Proof. Let \( \tilde{Q}(s) \) be the Youla Parametrization as defined in Youla et al. (1976).

\[
\tilde{Q}(s) = [\tilde{q}_{ij}(s)] \quad \forall i, j = 1, 2
\]

where \( \tilde{q}_{ij}(s) \)s (for \( i, j = 1, 2 \)) are stable semi-proper or strictly proper rational functions.
where \( \text{Re} \{ p_i \} > 0 \) for \( i = 3 \sim k + m + 2 \) and \( d_2(s) \), \( d_3(s) \) are polynomials with only LHP-zeros. \( n_3(s), n_4(s) \) and \( n_5(s) \) are the polynomials such that

\[
\frac{n_3(s)}{d_2(s)} \quad \text{and} \quad q_2(s)
\]

are proper. \( q_1(s), q_2(s) \) are designed such that \( \Delta'(s) \) of (7) satisfies Proposition 1. But then the next design also stabilizes the plant

\[
q_1(s) = \frac{n_3(s)}{d_2(s)}
\]

\[
q_2(s) = \frac{n_3(s)}{d_3(s)} \cdot q_2(s)
\]

because \( q_1(s)q_2(s) = q_1(s)q_2(s) \) and \( \Delta'(s) = \Delta''(s) \). However, this contradicts either (2) or (3) of Proposition 2. Hence (2) of Proposition 2 is false and one has to stabilize both loops for achieving decentralized stabilization.

The next example illustrates Theorem 1.

**Example 1.**

\[
G(s) = \frac{1}{1 + 0.11s} \begin{bmatrix} 1 & 0.28 \\ 0.85 & 0.064 \end{bmatrix}
\]

It is checked that \( \text{NI} < 0 \) and three different designs are discussed.

(1) Loops 1 and 2 are stable; for example, choose

\[
q_1(s) = 1
\]

\[
q_2(s) = 250 \frac{1 + 0.11s}{1 + \varepsilon s} \quad \varepsilon > 0
\]

Then the numerator of \( \Delta(s) \) is

\[
\text{num} (\Delta(s)) = e^2 + (9.09e + 1)s - 531.818
\]

It is obvious that (17) has a RHP-zero for any positive \( \varepsilon \) and \( G(s) \) can not be stabilized.

(2) Loop 1 is stable and loop 2 is unstable with two unstable poles. \( q_2(s) \) is the same as (15) and

\[
q_2(s) = 250 \frac{1 + 0.11s}{(-1 + \varepsilon s)(-1 + \phi s)} \quad \phi > 0
\]

It is routine to calculate that

\[
\text{num} (\Delta'(s)) = 0.11e\phi^3 + (\varepsilon\phi - 0.11e - 0.11\phi)s^2 - (\varepsilon + \phi - 0.11)e - 58.5.
\]

Again, \( G(s) \) can not be stabilized by any positive \( \varepsilon \) and \( \phi \).

(3) Loop 1 is stable and loop 2 is unstable with one unstable pole. \( q_2(s) \) is of (15) and

\[
q_2(s) = -250 \frac{1 + 0.11s}{1 + \varepsilon s} \quad \varepsilon > 0
\]

\[
\text{num} (\Delta'(s)) = 0.11e\phi^2 + (\varepsilon - 0.11)e + 58.5.
\]

The necessary and sufficient condition for (21) having LHP-zeros only is

\[
\varepsilon > 0.11.
\]

It is interesting to see the physical interpretation of the stability constraint, (22). From (2), (20) the feedback controller of loop 2 is given by

\[
\frac{1}{s} c_2(s) = \frac{-27.5}{\varepsilon} (1 + \frac{1}{1 + 0.11s}).
\]

Therefore the feedback gain is limited by (22). It should be tuned below 27.5/0.11 = 250 to guarantee the stability of \( G(s) \).

5. Conclusion

A new decentralized stability condition related to RGA and NI without requiring the stabilization of the individual loops is given. This result should be viewed as a generalization of previous work on RGA and NI. The result is limited to \( 2 \times 2 \) plants under multi-loop SISO control and should hopefully motivate further development for larger systems.

Acknowledgement—The authors would like to thank Prof. M. Morari for his helpful comments on the parametrization of stabilizing controllers. The financial support of National Science Foundation is gratefully acknowledged.

References

The same is true with the proposed method since, if the feed flow rate is constant,

$$\Delta D = -\Delta B$$  \hspace{1cm} (15)$$

The $K^{-1}$ matrix is singular for this case and cannot be inverted.

Conclusions

A new method has been presented for the derivation of steady-state process gains. It is a closed loop type of test, and it is particularly useful in addressing the manipulated variable selection problem. For a process with $n$ controlled variables, only $n$ tests are required for any number of sets of manipulated variables.

Nomenclature

- $B =$ bottom flow rate
- $BR =$ boil-up ratio
- $D =$ distillate flow rate
- $K =$ steady-state gain matrix
- $k_{ij} =$ steady-state gain between the $i$ controlled and the $j$ manipulated variables
- $k_{ij} =$ $ij$ element of $K^{-1}$
- $m_i =$ $j$ manipulated variable
- $R =$ reflux flow rate
- $RR =$ reflux ratio
- $S =$ side-stream flow rate
- $T_i =$ temperature on the $i$th tray of the column
- $V =$ vapor boil-up
- $x_B =$ bottoms composition
- $x_D =$ distillate composition
- $x_s =$ side-stream composition

Literature Cited


Decentralized Control Structure Selection Based on Integrity Considerations

Min-Sen Chiu and Yaman Arkun*

School of Chemical Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332-0100

This paper discusses the problem of decentralized control structure selection based on closed-loop integrity considerations. New necessary conditions are derived to achieve decentralized closed-loop stability in the presence of any combination of loop failures. The conditions are scaling invariant, require only steady-state gain information, and apply to multiloop single-input/single-output (SISO) systems or more general block diagonal control systems.

1. Introduction

In decentralized control structures, different subsets of measured outputs are paired with different subsets of manipulated variables and are controlled independently. The most common decentralized control structure is a multiloop SISO control system which pairs one output with one input. In case such fully decentralized controllers give poor stability and performance characteristics, a block decentralized control structure would be preferred, in which groups of inputs are paired with groups of outputs, producing a more general block diagonal control structure with improved closed-loop properties.

For large-scale industrial processes, decentralized control is preferred because multivariable centralized control requires too many control loops with increased cost and complexity of design and difficult implementation, tuning, and maintenance problems. In this paper, we assume that measured and manipulated variables are fixed, and we are only concerned with the pairing problem. The criterion for the selection of the control structure is based on integrity considerations. We require that any decentralized control structure possess closed-loop integrity in the sense that it should be stabilized by a controller having integral action, and it should maintain its nominal stability in the face of failures in its sensors and/or actuators. Therefore, we aim at developing necessary conditions for decentralized closed-loop integrity. We consider block diagonal decentralized control structures in general, and the class of failures is also kept general to include any combination of control loop failures. The results pertain only to stability, and dynamic performance is not addressed here.

Received for review June 16, 1989

Revised manuscript received October 31, 1989

Accepted November 7, 1989
selection procedure are given to achieve DCLI. Finally, the results are compared and put in perspective with recently proposed decentralized integral controllability (DIC) conditions, and the salient conclusions are drawn.

2. Preliminaries

For process control applications, decentralized controllers are very common because of their ease of implementation, design, tuning, and maintenance. However, synthesis of such controllers can be quite complex since for a given plant there are many alternative decentralized control structures to choose from. For example, for a plant with n manipulated inputs and n measured outputs, there are n! different multiloop SISO decentralized control systems. Thus, efficient screening techniques are needed for eliminating the undesirable control structures. In this paper, the criterion considered is that the control structure possesses decentralized close-loop integrity (DCLI). Before we define DCLI, the necessary notations are introduced below.

Let I be the set of integers I = {1, 2, ..., k} and J_i be the subset of I containing l = 1 - l elements of I. Therefore, for a k x k square matrix A, let A_I_j denote the corresponding l x l principal submatrices of A consisting of rows and columns with indices belonging to J_i. Thus, there are k!/[l!(k - l)!] A_I_j's for a given l and A_I_j = A. Furthermore, for a given A_I_j, there are l A_I_j's denoted by A_I_j, A_I_j, ..., A_I_j where the superscript means that the ith (1 <= i <= l) row and column of A_I_j are deleted. Also Re (k) = the real part of a complex number k and r(A) are the eigenvalues of a square matrix A.

For a k-channel decentralized control system, any open-loop stable plant G(s) can be rewritten as G(s) = [G_i(s)] for i = 1 - k and j = 1 - k. The diagonal blocks G_i(s) constitute the subsystems that are under decentralized control and G_i(s) = bd[G_i(s)] for i = 1 - k where bd means the block diagonal in general and reduces to a diagonal in multiloop SISO control systems. The decentralized IMC (DIMC) controller C_D(s) in Figure 1 is given by C_D(s) = bd[G_i(s)] where G_i(s) are stable. The corresponding decentralized feedback (DFB) controller C_F(s) = bd[C_D(s)] is obtained from

C_F(s) = G(s)[l - G(s)]^{-1} (1)

Furthermore,

G_i(s) = G(s) - G_i(s) (2)

and the block relative gains (Manousiouthakis et al., 1986) are given by

BRG_i = [I - G_i(s)G_i(s)^{-1}G_i(s)G_i(s)^{-1}] (3)

where

P_F.G_F = [G_i(s) G_i(s) G_i(s)] (4)

Figure 2. Decentralized feedback control structure.

P_F and P_G are square permutation matrices. In the multiloop SISO control system, the above block relative gains (BRG_i's) reduce to the diagonal elements of the relative gain array (RGA) introduced by Bristol (1966):

RGA[G(0)] = [\lambda_{ij}] (5)

and

\lambda_{ij} = [G(0)]_{ij}[G(0)^{-1}]_{ij} (6)

[A]_{ij} denotes the (i,j)th element of the matrix A. A plant G(s) is said to have positive RGA's if \lambda_{ij} > 0 \forall i = 1 - k. For a given matrix A, the Niederinski index (1971) is defined by

NI(A) = det[A - A_{ij}^{-1}] (7)

where A_{ij} = bd[A].

Both RGA and NI offer important insight into the issue of control structure selection. RGA is often used to measure interactions in a multiloop SISO system, while NI is used as a necessary condition for the closed-loop stability of the decentralized control system. Also, more recently, a DIC (decentralized integral controllability) rule was proposed (Morari and Zafiriou, 1989) and proved to be more comprehensive than either RGA or NI tests. Its definition will be given in section 3 where the comparison between the DIC rule and our result is made.

Here we define that a decentralized control system possesses integrity if the whole system is stable and remains stable in the face of any combination of loop failures. The following gives a more precise definition of the decentralized control system's integrity.

Definition 1. A k-channel stable plant G(s) is said to possess decentralized closed-loop integrity if it can be stabilized by a stable C_D(s) which contains integral action (Figure 2) and if it remains stable after failure occurs in one or more of the feedback loops. It is assumed that the corresponding controller block(s) is switched to manual after the detection and isolation of the loop failures. Therefore, the system's integrity with respect to any combination of loop failures requires G_D(s)C_D(s)[l + G_i(s)C_D(s)] to be stable for l = 1 - k.

3. Main Results

In the derivation of all the results in this section, we will assume the following:

Assumption.

1. G(s) is square, stable, proper, or strictly proper.
2. G(s)C_D(s) or G_D(s)C_D(s) is stable and strictly proper.
3. H_i(s) = G_i(s)G_i(s) = G_i(s)G_i(s)[l + G_i(s)C_D(s)]^{-1} is stable and has vanishing tracking error for asymptotically constant inputs; i.e., H_i(0) = I for i = 1 - k.
4. det [G_i(s)] = 0 for i = 1 - k, and G_i(0)'s are not singular for l = 2 - k.

The following important relationship between NI and BRG (or RGA) sets the foundation for the remaining developments.

Theorem 3.1.

NI[G(0)] = NI[G(0)] det [BRG^{-1}(G(0))] (8)
In a multiloop SISO control system, det [BRGᵢ(Gᵢ))] is replaced by λᵢᵢ. Note that G⁽¹⁾ is the reduced plant matrix after deleting the iᵗʰ row and column of G.

Proof. See Appendix.

Example 1. A µ = (1.006, 0.003, 0.727, 0.005)

0. Stably after the failure of any iᵗʰ loop only if

NI[G(0)] > 0 has one of the following properties. To be stabilized only if

X u [G(0)] > 0

Integrity against single-loop failures.

Further considerations have to be used to determine which one where “which one” is one of the undesirable control system properties, and they must be avoided if integrity considerations are important. In addition, corollary 2 identifies the source of the integrity problem by using λᵤᵢ and NI together.

Corollary 2 also relates closely to the definitions of j-sensor failure sensitive (j-SFS) and j-actuator failure sensitive (j-AFS) introduced by Grosdidier et al. (1985).

Definition 2. A multivariable control system with integral action is j-AFS (or j-SFS) if the complete system is stabilizable, but the reduced system with the jᵗʰ actuator and the jᵗʰ sensor (or the jᵗʰ sensor) removed is not.

Note that j-AFS and j-SFS are equivalent if the multivariable controller is decentralized. The next corollary follows directly from corollary 2 and theorem 3.1.

Corollary 3. Assume that NI[G(0)] > 0. For k > 2, the DFB or DIMC structure is j-AFS (or j-SFS) if either of the following conditions is satisfied: (1) NI[G⁽²⁾(0)] < 0, (2) det [BRG[G(0)][G(0)] < 0 or (λᵤᵢ[G(0)] < 0).

Therefore, NI[G(0)] > 0 itself does not provide complete information about the control system’s integrity either and can lead to undesirable pairing if used alone. The next example shows this fact.

Example 1 demonstrates this relationship.

Example 1. (Luyben (DL Case, 1986)).

\[
\begin{align*}
\text{G(0)} &= \begin{bmatrix}
-1.13 & -2.368 & -9.811 & 0.374 \\
5.24 & 0.422 & 5.984 & -1.986 \\
-0.35 & 0.513 & 2.38 & 0.0594 \\
4.48 & 15.54 & -11.3 & 0.176
\end{bmatrix} \\
\text{RGA: } \lambda_u &= [0.006, 0.003, 0.727, 0.005] \\
\text{NI}[G(0)] &= -490.891
\end{align*}
\]

λᵤᵢ’s are all positive but neither the original 4 × 4 system nor the reduced 3 × 3 system can be stabilized.

The next corollary further establishes the connection between BRG (or RGA) and integrity for the case of single-loop failure and answers the problem posed by Grosdidier et al. (1985, Discussion & Conclusion, p 225): “Further considerations have to be used to determine which one” where “which one” is one of the undesirable properties, and they must be avoided if integrity considerations are important.

Theorem 3.2. The closed-loop system of DFB structure can be stabilized only if

NI[G(0)] > 0.

Proof. See Appendix.

The next corollary relates to decentralized closed-loop integrity against single-loop failures.

Corollary 1. If k > 2 and det [BRG[G(0)] > 0 (or λᵤᵢ[G(0)] > 0) for i = 1 ~ k, the DFB or DIMC structure has one of the following properties.

1. The closed-loop system can be stabilized and remains stable after the failure of any iᵗʰ loop only if

NI[G(0)] > 0.

2. The closed-loop system is unstable and so is the reduced system without any iᵗʰ loop if

NI[G(0)] < 0.

Proof. See Appendix.

Therefore, corollary 1 illustrates why λᵤᵢ > 0 by itself is not sufficient in selecting control structures except in 2 × 2 cases for which NI > 0 is automatically satisfied.

Example 2 (Luyben (DL Case, 1986)).

\[
\begin{align*}
\text{G(0)} &= \begin{bmatrix}
-1.13 & -2.368 & -9.811 & 0.374 \\
5.24 & 0.422 & 5.984 & -1.986 \\
-0.35 & 0.513 & 2.38 & 0.0594 \\
4.48 & 15.54 & -11.3 & 0.176
\end{bmatrix} \\
\text{RGA: } \lambda_u &= [0.006, 0.003, 0.727, 0.005] \\
\text{NI}[G(0)] &= -490.891
\end{align*}
\]

Corollary 4. The closed-loop system of DFB or DIMC structure can be stabilized and remains stable after the failure of any iᵗʰ loop only if (i) NI[G(0)] > 0 and (2) det [BRG[G(0)] > 0 or (λᵤᵢ[G(0)] > 0), ∀ i = 1 ~ k.

Proof. See Appendix.

Corollary 4 requires the combined test of the positivity of the NI and λᵤᵢ. Yu and Luyben (1986) suggested corollary 4 as two of the three rules for eliminating undesirable pairings in multiloop SISO application. However, its unified theoretical justification given here is new.

For 3 × 3 systems, the necessary conditions of corollary 4 also apply to more than one loop failures. This is because the control system will remain stable after the failure of any combination of two loops by appropriate design of the controller (see assumption 3). Therefore, for 3 × 3 systems, the conditions in corollary 4 are necessary for the stability of (i) the whole system and (ii) the reduced system after any combination of loop failures. However, this is not the case for 4 × 4 or larger systems. In its original definition, decentralized closed-loop integrity requires stability in the presence of any combination of loop failures. The next theorem gives the necessary test for this purpose.

Theorem 3.3. The closed-loop system of DFB or DIMC structure possesses integrity only if

NI[Gₜₙ(0)] > 0, ∀ i = 2 ~ k

Proof. Repeatedly apply the NI test to the whole system and the reduced systems after all the possible loop failures occur.

Equation 9 is the property of a given decentralized control structure and is independent of how one designs the control law. Hence, if (9) is violated for a given control
structure, it is impossible to design a decentralized controller to meet the integrity requirement.

In general, there are \(2^n - (k + 1)\) tests in (9) for a \(k\)-channel decentralized control system. Moreover, theorem 3.3 is equivalent to the next result stated in terms of BRG (or RGA).

**Theorem 3.4.** The closed-loop system of DFB or DIMC structure possesses integrity only if

\[
\det \{\text{BRG}(G_i(0))\} > 0, \quad \forall i = 2 \sim k \tag{10}
\]

for any \(i \in [1, 2, ..., l]\). If multiloop SISO controllers are used, \(\det \{\text{BRG}(i)\}\) is replaced by \(\lambda_{\text{ac}}(i)\).

**Proof.** See Appendix.

The next example gives the application of Theorem 3.4.

**Example 4.** Look at \(G(0)\) in example 2 and consider integrity against 2-loop failures. One has to examine the stability of six \(G_i(0)\)’s. For example, if loops (3, 4) fail, where \(G_i(0) = \begin{bmatrix} -11.3 & -2.368 \\ 5.24 & 0.422 \end{bmatrix}\), \(\lambda_{\text{ac}}(G_i(0)) = -0.624\).

Therefore, the closed-loop system will be unstable if the failures of loops (3, 4) occur. In a similar fashion, the other relative gains can be computed in the order of failures of loops (2, 4), (2, 3), (1, 4), (1, 3), and (1, 2):

\[
\lambda_{\text{ac}}(G_i(0)) = |0.893, 6.348, -0.486, -0.002, 2.220|
\]

Hence, the closed-loop system will be unstable also if either loops (1, 4) or loops (1, 3) fail. Therefore, this control structure is not acceptable based on integrity consideration.

### 3.1. New Rules for Selecting Pairings To Achieve DCLI

Since \(\det \{\text{BRG}(i)\}\) (or \(\lambda_{\text{ac}}(i)\)) and NI are scaling invariant, the tests in theorems 3.4 and 3.5 are scaling invariant. Below we propose the rules for control structure selection based on decentralized closed-loop integrity consideration.

**Rule 1.** Select pairings with positive \(\det \{\text{BRG}(G(0))\}\) (or \(\lambda_{\text{ac}}(G(0))\)).

**Rule 2.** Select pairings with positive \(\text{NI}[G(0)]\).

**Rule 3a.** Select pairings with positive \(\det \{\text{BRG}(G_i(0))\}\) (or \(\lambda_{\text{ac}}(G_i(0))\)) for \(l = 2 \sim k - 2\).

**Rule 3b.** Select pairings with positive \(\text{NI}[G_i(0)]\) for \(l = 2 \sim k - 2\).

Control structures passing rules 1 and 2 pass DCLI against single-loop failure as given by corollary 4. Structures passing rule 3a or 3b for a particular value of \(i\) possess DCLI against any combination of \((k - l)\) loop failures.

In a multiloop SISO control system, the important property with rule 1 is that there is no need to recompute \(\lambda_{\text{ac}}(i)\) for an alternative control structure (Bristol, 1986). Therefore, one should eliminate quite a few alternatives just by inspecting rule 1 and then applying rule 2 and rule 3a (or rule 3b) for screening the remaining alternatives.

The following gives the procedure to select a control structure that satisfies DCLI rules.

**Step 1.** Choose \(G(s)\) such that rule 1 is satisfied.

**Step 2.** Check if rule 2 is satisfied. If not, \(G(s)\) is not acceptable, and go to step 1; if yes, integrity against single-loop failure is achieved. Set \(l = k - 2\), and go to step 3.

**Step 3.** If \(l = 0\) or 1, \(G(s)\) is acceptable based on DCLI rules and the screening procedure can terminate here. Otherwise go to step 4.

**Step 4.** Check if rule 3a (or rule 3b) is satisfied. If not, \(G(s)\) is not acceptable, and go to step 1; if yes, integrity against any combination of \((k - l)\) loop failures is achieved. Set \(l = l - 1\), and go to step 3.

The following example demonstrates the above selection procedure.

**Example 5.** Consider \(G(0)\) in example 2. Since \(\text{NI}[G(0)] < 0\), the selection procedure stops at step 2 and goes to step 1 to reconsider the other alternatives. For example, the structure with \((y_1, u_1), (y_2, u_3), (y_3, u_2), (y_4, u_3)\) and \((y_4, u_2)\) pairings of \(G(0)\) in example 2 has positive RGA’s.

**Step 1.** \(\lambda_{\text{ac}}[G(0)] = [1.0063, 1.0935, 0.7264, 0.8860]\).

**Step 2.** \(\text{NI}[G(0)] = 1.1814\), set \(l = 2\), and go to step 3.

**Step 3.** \(l = 0\) or 1; hence, go to step 4.

**Step 4.** \(\lambda_{\text{ac}}[G_i(0)] = [1.0957, 0.8926, 1.0643, 0.9748, 1.0024, 0.8645]\), set \(l = 1\), and go to step 3.

**Step 3.** \(l = 1\), and \(G(0)\) is acceptable based on DCLI rules, and the procedure terminates. The structure \((y_1, u_1), (y_2, u_3), (y_3, u_2), (y_4, u_3)\) possesses DCLI.

### 3.1.1. Relationship to Decentralized Integral Controllability

The following gives the definition of DIC (Morari and Zafiriou, 1989).

**Definition 3.** A plant is decentralized integral controllable if it is possible to design a diagonal controller for this plant which has (1) integral action, (2) yields stable individual loops, (3) is such that the system remains stable when all loops are closed simultaneously, and (4) has the property that each loop gain may be reduced independently with a factor \(\epsilon_i (0 \leq \epsilon_i \leq 1)\) without introducing instability.

Decentralized closed-loop integrity as defined in this paper corresponds to DIC for \(\epsilon_i = 0\) for \(i \in J_i\) and \(1 \leq l \leq k\); i.e., it addresses the stability issue when combination of control loops are taken out of service. However, it does not tackle the problem of detuning controllers (i.e., \(0 < \epsilon_i \leq 1\)) which is addressed in the above definition of DIC. Therefore, a system that has DCLI is not necessarily DIC, but DCLI is a necessary condition for DIC. That is, if a system is DIC, it must also have DCLI. Based on these observations, the DCLI tests developed here can be interpreted and used as additional necessary conditions for DIC to screen control structures.

The following gives the DIC rules for a multiloop SISO control system (Morari and Zafiriou, 1989).

**Rule 1.** Eliminate pairings with negative RGA’s.

**Rule 2.** Eliminate pairings with Re \(\text{RGA}[G^0(S)] < 0\). \(G^0(S)\) is obtained after adjusting \(G(0)\) such that all diagonal elements have positive signs.

**Rule 3.** Eliminate pairings with Re \(\text{RGA}[E(S)] < 1\); \(E = G^0\).

The next subsection gives the comparison between the DCLI and DIC rules.

### 3.1.2. Examples

**Example 6.** Consider \(G(0)\) in example 2 with the pairings \((y_1, u_1), (y_2, u_2), (y_3, u_2), (y_4, u_3)\) which has RGA: \(\lambda_{\text{ac}}[G_i(0)] = [1.0063, 0.0003, 0.0025, 0.1366]\).

Further analysis shows that

**DCLI:** \(\text{NI}[G(0)] = 892\)

\(\lambda_{\text{ac}}[G_i(0)] = [-0.624, 2.152, 0.744, 0.008, 0.049, -1.224]\)

**DIC:** \(\text{r}[\text{RGA}(0)] = [0.266 \pm 1.976, 11.256 \pm 10.958, 4.801, 0.008, -2.405 \pm 12.270]\)

Therefore, both rules conclude that this pairing is not acceptable.

**Example 7** (Morari and Zafiriou, 1989)

\(G(0) = \begin{bmatrix} 8.72 & -15.80 & 2.98 & 2.81 \\ 6.54 & -20.79 & 2.50 & -2.92 \\ -5.82 & -7.51 & -1.48 & 0.99 \\ -7.23 & 7.86 & 3.11 & 2.92 \end{bmatrix}\)

which has positive RGA’s.

RGA: \(\lambda_{\text{ac}}[G(0)] = [0.4142, 0.4372, 0.1712, 0.0011]\).
Further analysis shows that

**DCLI:** 
\[ \text{NI}(G(0)) = 7.05 \]

\[ \lambda_{i1}[G_{i1}(0)] = \{2.326, -2.908, 0.556, 0.621, 1.608, 0.584\} \]

**DIC:** 
\[ r[G^*(0)] = \{27.131, 6.775, 0.002 \pm 5.48j\} \]

\[ r[E(0)] = \{0.876, 1.098, -0.987 \pm 1.338j\} \]

DIC rules will accept this pairing. However, from DCLI rules the closed-loop cannot be made stable if loops 2 and 4 fail, and therefore, this pairing is not acceptable.

At first this conclusion seems to be surprising, since by definition DIC is the best property a decentralized control system can possess and as discussed before should be a tighter requirement than DCLI. However, the above DIC rules are actually derived from integral controllability (IC) results in which all the loop gains are reduced simultaneously and not independently. Clearly this not the case in the DCLI tests in which the loops are taken out of service independently. Therefore, a structure acceptable by the above DIC rules may be rejected by the DCLI rules.

**Example 8.** \(G(0)\) in example 2 with the pairings \((y_1, u_1)\), \((y_2, u_2)\), and \((y_3, u_4)\) has

\[ \lambda_{ib}(G(0)) = \{1.0063, 0.0107, 0.1630, 0.0054\} \]

Further analysis shows that

**DCLI:** 
\[ \text{NI}(G(0)) = 160.61 \]

\[ \lambda_{i1}[G_{i1}(0)] = \{4.172, 0.881, 6.348, 1.486, 0.045, 0.222\} \]

**DIC:** 
\[ r[G^*(0)] = \{2.433, 16.358, -0.409 \pm 4.947j\} \]

\[ r[E(0)] = \{-0.027, 3.785, -1.879 \pm 5.808j\} \]

Hence, DIC rules will not accept this pairing, while DCLI rules will accept it.

Therefore, it is shown that DCLI and DIC rules are independent of each other, and the former should be used as additional necessary conditions for DIC.

4. Conclusions

New necessary conditions to select control structures are given based on control system integrity considerations. They require only steady-state plant gain information and are scaling invariant. Examples show that they are also independent of the recently proposed DIC rules.

Acknowledgment

Financial support from the National Science Foundation is gratefully acknowledged.

Appendix

**Proof of Theorem 3.1.** For a given matrix \(G\),

\[
\text{det}[BRG(G)] = \frac{1}{\text{det}(G_{ei}) \text{det}(G_u) \text{det}(G_{e}) \text{det}(G_{b})} \cdot \frac{\text{det}(I - G_{ei}G_{e}^{-1}G_{b}G_{u}^{-1})}{\text{det}(G_{ei}) \text{det}(G_u) \text{det}(G_{e}) \text{det}(G_{b})}
\]

(11)

where the Schur's lemma is applied to (11). Furthermore, the following important relationship can be obtained from the definition of NI:

\[
\text{NI}(G(0)) = \frac{\text{det}[G(0)]}{\text{det}[G_{e}]} \cdot \frac{\text{det}[G_{b}]}{\text{det}[G(0)]} \cdot \frac{\text{det}[G_{b}]}{\text{det}[G_{e}]} \cdot \frac{\text{det}[G_{b}]}{\text{det}[G(0)]} = \frac{\text{det}[G(0)]}{\text{det}[G_{e}]} \cdot \frac{\text{det}[G_{b}]}{\text{det}[G(0)]} \cdot \frac{\text{det}[G_{b}]}{\text{det}[G(0)]} \cdot \frac{\text{det}[G_{b}]}{\text{det}[G(0)]}
\]

\[ \text{where } G_{b} = bd[G_{d}] \text{. The same procedure holds for a multiloop control system with } \text{det}[BRG_{i}(s)] \text{ replaced by } \lambda_{ib}(s). \]

**Proof of Theorem 3.4.** When \(l = 2\), it is understood that the closed-loop system of any \(2 \times 2\) plant will be stable only if \(\text{det}[BRG_{i}(G_{i}(0))] > 0\). When \(l = 3\), \(\text{NI}[G_{i}(0)] = \text{NI}[G_{i}(0)] / \text{det}[BRG_{i}(G_{i}(0))]\). Thus, \(\text{NI}[G_{i}(0)] > 0\) of theorem 3.3 is equivalent to \(\text{det}[BRG_{i}(G_{i}(0))] > 0\) for any \(i = 1 \sim 3\). The same procedure carries over to \(l = 4 \sim k\).

**Proof of Corollary 1.** If \(\text{det}[BRG_{i}(G_{i}(0))] > 0\), \(\text{NI}(G(0))\) has the same sign of \(\text{NI}[G(0)]\) from theorem 3.1. Hence, from theorem 3.2 and remembering that \(\text{NI}[G(0)]\) is the Niederlinski's index for the reduced system without ith loop, the proof is complete.

**Proof of Corollary 2.** If \(\text{det}[BRG_{i}(G_{i}(0))] < 0\), \(\text{NI}(G(0))\) has the opposite sign of \(\text{NI}[G(0)]\) from theorem 3.1. Therefore, the closed-loop system will be unstable and the reduced system after opening the ith control loop can be made stable if \(\text{NI}(G(0)) < 0\). Moreover, the closed-loop system is not failure tolerant if \(\text{NI}(G(0)) > 0\).

**Proof of Corollary 4.** Condition 1 is the consequence of theorem 3.2. Moreover, the control system can remain stable after the failure of any ith loop only if \(\text{NI}[G(0)] > 0\) for \(i = 1 \sim k\), and under condition 1, this is equivalent to \(\text{det}[BRG_{i}(G_{i}(0))] > 0\) for \(i = 1 \sim k\) from theorem 3.1.

**Literature Cited**


Revised manuscript received October 24, 1989
Accepted November 22, 1989
A Methodology for Sequential Design of Robust Decentralized Control Systems

Min-Sen Chiu and Yaman Arkun

Automatia (in press)

Key Words - Robust control, decentralized control, sequential design

Abstract

A general analysis theory is presented which can incorporate robust stability or robust performance defined in $H^\infty$ or $\mu$ framework for sequential design purposes. As a result, a methodology for sequential design of robust decentralized controllers is proposed. To do so, new formulations of linear fractional transformation of complementary sensitivity function and sensitivity function are given. They are used to derive robustness constraints on individual decentralized controllers. When these robustness constraints are satisfied, the closed-loop system is guaranteed to possess pre-specified robustness properties if and only if the whole system is nominally stable as well.

---

1This work was supported by the National Science Foundation.
2He was with the School of Chemical Engineering, Georgia Institute of Technology, Atlanta, GA 30332. He is now with the International Paper, Georgetown, SC 29442.
3To whom all correspondence should be addressed. Professor, School of Chemical Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0100, 404-894-2871, FAX 404-894-2866
1. INTRODUCTION

For large scale industrial processes decentralized control is preferred because multivariable centralized control requires too many control loops with increased cost and complexity of design, and difficult implementation, tuning and maintenance problems. In order to facilitate decentralized control system design, it is desirable to be able to transform the design constraints (e.g. robust performance) for the whole system to the design constraints on individual decentralized controllers. Toward this end, two classes of design procedures have been reported in the literature. The first class is independent design where each controller element is designed independently of each other (Bernstein and Haddad, 1989; Grosdidier and Morari, 1986; Nwokah, 1987; Skogestad and Morari, 1989; Wu and Mansour, 1988). As described in Skogestad and Morari (1989) the method is potentially conservative since during the design of a particular controller the information on other controllers is not exploited. The second class is sequential design in which controller design is conducted sequentially (Bernstein, 1987; Chiu and Arkun, 1989; Davison and Gesing, 1979; Mayne, 1976; Nett and Uthgenannt, 1988; Tan and Ikeda, 1987; Viswanadham and Taylor, 1988). At each design step one utilizes the information about controllers designed in the previous steps; therefore, the method can be less conservative than independent design.

The robust control problem is solved using different techniques within the independent design framework (Bernstein and Haddad, 1989; Nwokah, 1987; Skogestad and Morari, 1989). Although the authors in (Bernstein, 1987; Viswanadham and Taylor, 1988) claim that their methods can handle robustness, systematic procedures for sequential design of robust decentralized controllers are presently unavailable. In this paper, we present a general analysis theory which can incorporate robust performance defined in $H^\infty$ or $\mu$ framework for sequential design purposes. As a result, a methodology for sequential design of robust decentralized controllers is proposed.

2. PRELIMINARIES

2.1. Notation

Consider the decentralized feedback control system given in Fig. 1. Here the plant $G(s)$ is stable and square. For a $k$–channel decentralized control system $G(s)$ can be rewritten as

$$G = [G_{ij}]_{i,j=1\ldots k}$$

(1)

where $G_{ij}$ is $n_i \times n_j$ transfer function matrix and $n_1 + n_2 + \ldots + n_k = n$. The decentralized controller $C_d(s)$ is given by

$$C_d = bd [C_i]_{i=1\ldots k}$$

(2)
with \(bd\) denotes block diagonal form and \(C_i\) is the \(i\)-th decentralized controller for the \(i\)-th subsystem \(G_{ii}\).

The plant matrix \(G(s)\) can also be expressed by

\[
G \overset{\text{def}}{=} \begin{bmatrix}
G_{11}^l & G_{12}^l \\
G_{21}^l & G_{22}^l
\end{bmatrix} \quad \forall \ l = 1 \sim k - 1
\]

(3)

where the superscript \(l\) denotes the number of subsystems aggregated in \(G_{11}^l\). The corresponding partition of the decentralized controller \(C_d(s)\) is given by

\[
C_d = \begin{bmatrix}
C_1^l & 0 \\
0 & C_2^l
\end{bmatrix} \quad \forall \ l = 1 \sim k - 1
\]

(4)

The complementary sensitivity function and sensitivity function of the whole closed-loop system is given by

\[
H = GC_d(I + GC_d)^{-1}, \quad S = I - H = (I + GC_d)^{-1}
\]

(5)

Similar functions are defined for the aggregated subsystems \(G_{11}^l\) (for \(l=1\sim k-1\)) and the individual subsystems \(G_{ii}\) (for \(i=1\sim k\)):

\[
\bar{H}_1^l = G_{11}^l C_1^l (I + G_{11}^l C_1^l)^{-1}, \quad \bar{S}_1^l = I - \bar{H}_1^l \quad \forall \ l = 1 \sim k - 1
\]

(6)

\[
\bar{H}_i = G_{ii} C_i (I + G_{ii} C_i)^{-1}, \quad \bar{S}_i = I - \bar{H}_i \quad \forall \ i = 1 \sim k
\]

(7)

Also define

\[
\bar{H}_2^l = bd [\bar{H}_i]_{i=1+l}^{i+k-1} \quad \bar{S}_2^l = bd [\bar{S}_i]_{i=1+l}^{i+k-1}
\]

\[
\alpha^l = G_{12}^l G_{22}^l, \quad \beta^l = G_{21}^l C_2^l \bar{S}_1^l
\]

\[
\theta^l = (I - \beta^l \alpha^l)^{-1}, \quad \phi^l = G_{22}^l (bd [G_{ii}]_{i=1+l}^{i+k-1})^{-1}
\]

(8)

for \(l = 1 \sim k - 1\).

2.2. Nominal stability and robust performance

The minimal requirement on the closed-loop system is nominal stability (NS). It is also well known that NS is possible if and only if \(G(s)\) has no unstable decentralized fixed modes (Wang and Davison, 1973). In this case, we say that \(C_d(s)\) stabilizes \(G(s)\) and denote the set of such stabilizing controllers \(S_d(G)\). Robust Performance (RP) is the central design objective considered in this work. We require that \(H^\infty\) performance specification be satisfied by \(G_p\) in the family \(\Pi\) of possible plants:

\[
||W_p S_p||_\infty = \sup_\omega \sigma^*(W_p S_p) < 1 \quad \forall \ G_p \in \Pi
\]

(11)
where \( \sigma^*(\cdot) \) denotes the maximum singular value of \( (\cdot) \) and

\[
S_p = (I + G_p C_d)^{-1}
\]

and \( W_p \) is the performance weight. Thus the RP objective is to design \( C_d(s) \in S_d(G) \) to stabilize all \( G_p \)'s in the family \( \Pi \) and satisfy the performance requirement stated in (11). Here we consider the set \( \Pi \) of possible plants \( G_p \) described by

\[
\Pi = \{ G_p = G(I + l_I \Delta_I), \: \sigma^*(\Delta_I) < 1 \}
\]

where \( l_I \) is the magnitude of the input multiplicative uncertainty. Also let \( \Delta_p \) be an unstructured full matrix with the same dimension of \( S \) and \( \sigma^*(\Delta_p) < 1 \). This is the standard fictitious uncertainty block used to incorporate performance into robustness analysis. Finally \( \mu_\Delta(M) \) will denote the structured singular value of \( M \) with respect to the structure of \( \Delta \) which is block diagonal.

3. SEQUENTIAL DESIGN

In sequential design (SD) controller design is conducted sequentially. At each design step one utilizes the information about the controllers specified in the previous step; therefore, the method can be less conservative than independent design (ID). However, the previous works in SD do not handle RP. Here we give the main theoretical results which lay the foundation for the proposed SD method.

3.1. New formulas for linear fractional transformation (LFT) of \( S \) and \( H \)

The next lemma summarizes the new formulas for LFT of \( S \) and \( H \) which are used in the main theorem that follows.

**Lemma 1.** \( S \) and \( H \) as defined in (5) have the following LFT's:

\[
S = N_{11}^{ss} + N_{12}^{ss} T^s(I - N_{22}^{ss} T^s)^{-1} N_{21}^{ss}
\]

\[
H = N_{11}^{hh} + N_{12}^{hh} T^h(I - N_{22}^{hh} T^h)^{-1} N_{21}^{hh}
\]

where

\[
T^s = \begin{bmatrix} x^s \\ \bar{S}_2^s \end{bmatrix}, \quad T^h = \begin{bmatrix} x^h \\ \bar{H}_2^h \end{bmatrix}
\]

and \( x^s, x^h \) are all bounded matrices with appropriate dimensions and

\[
N_{11}^{ss} = \begin{bmatrix} \bar{S}_1^s (I + \alpha^t \theta^t \beta^t) & -\bar{S}_1^s \alpha^t \theta^t \\ 0 & 0 \end{bmatrix}, \quad N_{12}^{ss} = \begin{bmatrix} 0 & \bar{S}_1^s \alpha^t \theta^t \\ 0 & I \end{bmatrix}
\]
\[ N_{21}^{hh} = \begin{bmatrix} \tilde{S}_i - \tilde{H}_i \alpha^l \beta^l & \tilde{H}_i \alpha^l \beta^l \\ -\phi^{l-1} \beta^l & \phi^{l-1} \beta^l \end{bmatrix}, \quad N_{22}^{hh} = \begin{bmatrix} 0 & -\tilde{H}_i \alpha^l \beta^l \\ 0 & I - \phi^{l-1} \beta^l \end{bmatrix} \]  
(18)

\[ N_{11}^{hh} = \begin{bmatrix} \tilde{H}_i^l & 0 \\ \beta^l & 0 \end{bmatrix}, \quad N_{12}^{hh} = \begin{bmatrix} 0 & \tilde{S}_i \alpha^l \beta^l \\ 0 & \phi^{l-1} \beta^l \end{bmatrix} \]  
(19)

\[ N_{21}^{hh} = \begin{bmatrix} \tilde{S}_i & 0 \\ -\beta^l & I \end{bmatrix}, \quad N_{22}^{hh} = \begin{bmatrix} 0 & \tilde{H}_i \alpha^l \phi^l \\ 0 & I - \phi^{l-1} \phi^l \end{bmatrix} \]  
(20)


Unlike in the case of \( \mathcal{D} \) the above LFT's of \( H \) and \( S \) are dependent on the controllers \( C_i^l \) as illustrated in Fig. 2 for the LFT of \( H \). The matrix \( N^{hh} = \begin{bmatrix} N_{11}^{hh} & N_{12}^{hh} \\ N_{21}^{hh} & N_{22}^{hh} \end{bmatrix} \) is a function of the first \( l \) controllers as seen from (19) and (20), while the rest \( k-l \) controllers are incorporated into \( T^h \). Such dependence leads to the main result presented in the next theorem.

**Main Theorem** Assume that \((G, C_d)\) is nominally \((\Delta_f = 0)\) internally stable. Then RP with input uncertainty is achieved if for \( a \) \( l = 1 \sim k-1 \)

\[ \sigma^*(\tilde{H}_i) < \psi^l_h \quad \forall \quad i = l + 1 \sim k \]  
(21)

where \( \psi^l_h \) solves

\[ \mu_{(\Delta_f \Delta_p \bar{H}_i^l)} \begin{bmatrix} N_{11}^{h} & N_{12}^{h} \\ \psi^l_h N_{21}^{h} & \psi^l_h N_{22}^{h} \end{bmatrix} = 1 \]  
(22)

Or

\[ \sigma^*(\tilde{S}_i) < \psi^l_s \quad \forall \quad i = l + 1 \sim k \]  
(23)

where \( \psi^l_s \) solves

\[ \mu_{(\Delta_f \Delta_p \bar{S}_i^l)} \begin{bmatrix} N_{11}^{h} & N_{12}^{h} \\ \psi^l_s N_{21}^{h} & \psi^l_s N_{22}^{h} \end{bmatrix} = 1 \]  
(24)

for \( \omega \in [0, \infty) \). The matrices in (22) and (24) are computed from

\[ N_{11}^{h} = \begin{bmatrix} 0 & 0 \\ W_p G & W_p I \end{bmatrix} - \begin{bmatrix} l_f G^{-1} \\ W_p I \end{bmatrix} \begin{bmatrix} \tilde{H}_i^l & 0 \\ \beta^l & 0 \end{bmatrix} \begin{bmatrix} G & I \end{bmatrix} \]  
(25)

\[ N_{12}^{h} = -\begin{bmatrix} l_f G^{-1} \\ W_p I \end{bmatrix} \begin{bmatrix} \tilde{S}_i^l \alpha^l \phi^l \\ \theta^{l-1} \phi^l \end{bmatrix} \]  
(26)
\[ \mathcal{N}_{21}^\alpha = \begin{bmatrix} -\beta^l & I \end{bmatrix} \begin{bmatrix} G & I \end{bmatrix}, \quad \mathcal{N}_{22}^\beta = I - \theta^{-1}\phi^l \] (27)

\[ \mathcal{N}_{11}^\alpha = \begin{bmatrix} -l_1 & -l_1G^{-1} \\
0 & 0 \end{bmatrix} + \begin{bmatrix} l_1G^{-1} \\
W_pI \end{bmatrix} \begin{bmatrix} \tilde{S}_1(I + \alpha^l\theta^l\beta^l) & -\tilde{S}_1\alpha^l\theta^l \\
0 & 0 \end{bmatrix} \begin{bmatrix} G & I \end{bmatrix} \] (28)

\[ \mathcal{N}_{12}^\alpha = \begin{bmatrix} l_1G^{-1} \\
W_pI \end{bmatrix} \begin{bmatrix} \tilde{S}_1\alpha^l\theta^l \\
I \end{bmatrix} \] (29)

\[ \mathcal{N}_{21}^\alpha = \begin{bmatrix} -\phi^{-1}\theta^l\beta^l & \phi^{-1}\theta^l \end{bmatrix} \begin{bmatrix} G & I \end{bmatrix}, \quad \mathcal{N}_{22}^\alpha = I - \phi^{-1}\theta^l \] (30)

**Proof:** See Appendix.

Unlike the bounds calculated for \(\mathcal{TD}\), the robust performance bounds \(\psi_h^1, \psi_s^1\) are dependent on the first \(l\) controllers \((C_i^1)\). Fig. 3 shows this dependence. \(N_h^{SP}\) in the figure is the matrix given in (22) which is a function of the first \(l\) controllers. Physically such a dependence delineates the design constraints imposed on the remaining \(k-l\) controllers as the first \(l\) loops are closed. In particular (21) expresses how the complementary sensitivity functions of the remaining \(k-l\) subsystems are constrained by the design of the first \(l\) controllers if one is to achieve \(RP\). In contrast, in \(\mathcal{TD}\) \(N_h^{TD}\) depends on the plant matrix \(G(s)\) only and hence can lead to conservative designs as noted in Skogestad and Morari (1989).

The two constraints (21) and (23) in the Main Theorem can be combined to use over the frequency range just like in \(\mathcal{TD}\). Our experience has shown that (21) is often violated at low frequency range where (23) can usually be satisfied. On the other hand, (21) is more easily satisfied at high frequency range.

4. SEQUENTIAL DESIGN PROCEDURE

In the Main Theorem the \(RP\) design constraints \(\psi_h^1, \psi_s^1\) are dependent on \(l \in \{1, 2, ..., k-1\}\) and such a dependence suggests a *sequential design* method. In the following we will explore this connection. In general, \(SD\) involves two design steps: initial design and redesign of \(C_i\). Without loss of generality, the procedure is presented for design of single-input—single-output decentralized controllers (i.e. \(G_{ii}\) and \(C_i\) for \(i = 1 \sim k\) are scalars). Again without loss of generality each controller is parametrized by the IMC (Internal Model Control) technique (Morari and Zafiriou, 1989) and has one IMC tuning parameter \(\epsilon_i\). Only the salient features are presented below. For details the reader is referred to Chiu (1991).

**Initial Design:**

**Step 1:** Set \(l = 1\) in the Main Theorem. Then we need information on the first controller “\(C_1^1\)” in order to solve the constraints \(\psi_h^1, \psi_s^1\). Therefore one has to decide which subsystem
and $\Delta_f$ in (13) has the structure of $C_d$.

The uncertainty and performance weights are

$$l_f(s) = 0.07 \quad W_{p}(s) = 0.25 \frac{7s + 1}{7s}$$  \hspace{1cm} (32)

The decentralized controllers are chosen to give the following complementary sensitivity functions:

$$\tilde{H}_1(s) = \frac{1}{(\epsilon_1 s + 1)^2}, \quad \tilde{H}_2(s) = \frac{e^{-s}}{(\epsilon_2 s + 1)^2} \quad \epsilon_1, \epsilon_2 > 0$$  \hspace{1cm} (33)

where $\epsilon_i$'s are to be tuned to achieve RP. The corresponding controllers are

$$C_1(s) = \frac{23.49s + 0.60}{\epsilon_1 s (\epsilon_1 s + 2)}, \quad C_2(s) = \frac{2.71s + 0.71}{\epsilon_2 s (\epsilon_2 s + 2)}$$  \hspace{1cm} (34)

It can be easily checked that NS holds for any positive $\epsilon_1$ and $\epsilon_2$. To apply $SD$ for RP we will discuss two different design sequences and show that subsystem $G_{22}$ should be closed first.

(1) **Design sequence** \{G_{11}, G_{22}\}.

**Initial design:** Suppose we close $G_{11}$ first with $\epsilon_1 = 4$ and solve (22) and (24) for $\psi_h^1, \psi_s^1$ as shown in Fig. 4. It turns out that $7.9 < \epsilon_2 < 15.1$ will satisfy the RP constraints. This is illustrated in Fig. 4 for $\epsilon_2 = 10$.

**Redesign - Iteration 1:** The initial design will result in $(4, \epsilon_2^*) = (4, 8)$. Next we proceed to check whether we can design $\epsilon_1$ smaller than 4. Thus we close the second loop using the initial design $\epsilon_2 = 8$ and compute the RP bounds $\psi_h^1, \psi_s^1$ for the first loop as shown in Fig. 5. It turns out that $\epsilon_1 < 2.5$ will satisfy the RP constraints. Fig. 5 demonstrates this for $\epsilon_1 = 0.1$.

Since there is no lower bound on $\epsilon_1$, it can be chosen as small as possible as far as RP for input uncertainty is concerned. However other practical considerations, e.g. actuator limits, will limit the magnitude of gain. Therefore $\epsilon_1 = 0.1$ is used here as the lowest allowable value. Hence iteration 1 yields $(0.1, 8)$ for the tuning parameters $(\epsilon_1, \epsilon_2)$.

**Redesign - Iteration 2:** Now we want to update $\epsilon_2$. Set $\epsilon_1 = 0.1$. Solve the Main Theorem for $\psi_h^1, \psi_s^1$ and design $\epsilon_2$ to satisfy the RP constraints. One gets the feasible range $2.3 < \epsilon_2 < 15$.

Since $\epsilon_1$ can not be tuned below 0.1 there is no need to update $\epsilon_1$, the design ends with two iterations giving $(0.1, 2.4)$ as the best design. Table 1 summarizes the design steps and gives a comparison with the classical $\mu$ design approach based on (35). It shows that as iterations evolve the solution space admitted by $SD$ converges to that of $\mu$ design. Table 2 lists $\mu^*(M^{RP})$ for different designs, $(0.1, 2.4)$ yields the highest peak value among high gain designs.

(2) **Design sequence** \{G_{22}, G_{11}\}.

**Initial design:** Suppose we close $G_{22}$ first with $\epsilon_2 = 4$. Then we solve for $\psi_h^1, \psi_s^1$ and get the feasible values $\epsilon_1 < 0.8$. For the reason stated earlier, we set $\epsilon_1 = 0.1$ and start the redesign.
Redesign: This will give the same result obtained in iteration 2 of the previous design sequence. Hence (0.1, 2.4) is the best design and the design terminates.

Comparing the initial design phase of the two possible sequences it is important to note that \( G_{22} \) should be closed first since it puts less constraint on the design of the remaining loop, i.e. it provides smaller lower bound on the \( e \) of the remaining loop while satisfying \( \text{RP} \). It is also shown that designing \( G_{22} \) first converges faster than designing \( G_{11} \) first.

It should be noted that \( TD \) fails to find controllers (for any positive \( e_1, e_2 \)) to satisfy \( \text{RP} \).

6. CONCLUSIONS

New formulations of linear fractional transformation of the complementary sensitivity function \( (H) \) and sensitivity function \( (S) \) are derived. Their application provides a theoretical basis for the sequential design of robust decentralized controllers.

Acknowledgement – The authors wish to thank Prof. Michael Fan for providing the software to calculate \( \mu \).

REFERENCES


**APPENDIX**

**Proof the Main Theorem:** From Doyle (1982) $\textbf{RP}$ for input uncertainty is achieved if and only if

$$\mu(\Delta_I, \Delta_p) (M^{RP}) < 1 \quad \forall \omega$$

where $M^{RP}$ can be expressed as a LFT of $H$:

$$M^{RP} = \begin{bmatrix} 0 & 0 \\ W_pG & W_pI \end{bmatrix} - \begin{bmatrix} l_I G^{-1} & \tilde{H}_I \\ W_pI & \beta_I \end{bmatrix} H \begin{bmatrix} G & I \end{bmatrix}$$

Now substitute (15) into (36) to get

$$M^{RP} = N_{11}^h + N_{12}^h T^h (I - N_{22}^h T^h)^{-1} N_{21}^h$$

where

$$N_{11}^h = \begin{bmatrix} 0 & 0 \\ W_pG & W_pI \end{bmatrix} - \begin{bmatrix} l_I G^{-1} & \tilde{H}_I \\ W_pI & \beta_I \end{bmatrix} \begin{bmatrix} G & I \end{bmatrix}$$

$$N_{12}^h = \begin{bmatrix} 0 & -\begin{bmatrix} l_I G^{-1} \\ W_pI \end{bmatrix} \begin{bmatrix} \tilde{S}_1 & \phi^I \\ \theta^{-1} & \phi^I \end{bmatrix} \end{bmatrix} \overset{\text{def}}{=} \begin{bmatrix} 0 & N_{12}^h \end{bmatrix}$$

$$N_{21}^h = \begin{bmatrix} [\tilde{S}_1 & 0] \begin{bmatrix} G & I \end{bmatrix} \end{bmatrix} \overset{\text{def}}{=} \begin{bmatrix} n_{21}^h \\ N_{21}^h \end{bmatrix}$$
\[ N_{22}^h = \begin{bmatrix} 0 & \bar{H}_2^1 \alpha^i \phi^i \\ 0 & I - \theta^i \phi^i \end{bmatrix} \overset{\text{def}}{=} \begin{bmatrix} 0_h & n_{22}^h \\ 0 & N_{22}^h \end{bmatrix} \quad (41) \]

One sufficient condition for this inequality to hold can be obtained by applying the theorem given in Skogestad and Morari (1988) to (35) and (37). It gives

\[ \sigma^*(T^h) < \psi_h^l \quad \forall \omega \in [0, \infty) \quad (42) \]

or

\[ \sigma^*(\bar{H}_2^1) < \psi_h^l \quad \forall \omega \in [0, \infty) \quad (43) \]

for \( l = 1 \sim k - 1 \). The bound \( \psi_h^l \) solves

\[ \mu(\Delta_l \Delta_r \bar{A}_l^1) \begin{bmatrix} N_{11}^h & 0 & N_{12}^h \\ \psi_h^l n_{21}^h & 0_h & \psi_h^l n_{22}^h \\ \psi_h^l N_{21}^h & 0 & \psi_h^l N_{22}^h \end{bmatrix} = 1 \quad \forall \omega \quad (44) \]

where \( \Delta_h \) has the same dimension of \( 0_h \).

(44) reduces to (Chiu, 1991)

\[ \mu(\Delta_l \Delta_r \bar{A}_l^1) \begin{bmatrix} N_{11}^h & N_{12}^h \\ \psi_h^l N_{21}^h & \psi_h^l N_{22}^h \end{bmatrix} = 1 \quad \forall \omega \quad (45) \]

This completes the proof of part (1). Part (2) can be proved following the same procedure. Finally, as discussed in (Skogestad and Morari, 1988), (22) and (24) can be combined to use for \( \omega \in [0, \infty) \) and this completes the proof. \( \square \)
Figure 1: Decentralized feedback control structure.

Figure 2: The linear fractional transformation of $H$ with respect to $T^H$.

Figure 3. Sequential design

Figure 4: $SD$ : initial design with $\epsilon_1 = 4$. $\epsilon_2 = 10$ is feasible since $|\tilde{S}_2| < \psi_2^1$ at $\omega < 0.14$ and $|\tilde{H}_2| < \psi_2^1$ at $\omega > 0.121$.

Figure 5: $SD$ : iteration 1 with $\epsilon_2 = 8$. $\epsilon_1 = 0.1$ is feasible since $|\tilde{S}_1| < \psi_2^1 \forall \omega$.

Table 1: Iterations of design sequence $\{G_{11}, G_{22}\}$. $-$ denotes the best design.

Table 2: $\mu^*(M^{RP})$ for different designs.
\begin{table}
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{Iteration} & \textbf{SD} & \textbf{\((\epsilon_1^*, \epsilon_2^*)\)} & \textbf{\(\mu\) Design} \\
\hline
1 & \(\epsilon_1 = 4.0\) & \(7.9 < \epsilon_2 < 15.1\) & \((4.0, 8.0)\) & \(\epsilon_2^\text{mu} < 15.4\) \\
1 & \(\epsilon_2 = 8.0\) & \(\epsilon_1 < 2.5\) & \((0.1, 8.0)\) & \(\epsilon_1^\text{mu} < 7.3\) \\
2 & \(\epsilon_1 = 0.1\) & \(2.3 < \epsilon_2 < 15\) & \((0.1, 2.4)\) & \(2 < \epsilon_2^\text{mu} < 16\) \\
\hline
\end{tabular}
\end{table}

\textit{Table 1}

\begin{table}
\begin{tabular}{|c|c|c|}
\hline
\textbf{Iteration} & \textbf{\(\epsilon_1\)} & \textbf{\(\epsilon_2\)} & \textbf{\(\mu^*(M^{RP})\)} \\
\hline
1 & 4.0 & 8.0 & 0.89 \\
1 & 4.0 & 15 & 0.98 \\
2 & 0.1 & 8.0 & 0.61 \\
2 & 2.4 & 8.0 & 0.80 \\
\hline
\end{tabular}
\end{table}

\textit{Table 2}
\[ C_d \]

\[ H \]

\[ T^h \]

\[ N^{hh} = \begin{bmatrix} N_{11}^{hh} & N_{12}^{hh} \\ N_{21}^{hh} & N_{22}^{hh} \end{bmatrix} \]
Figure 3
Fig 5
A Decentralized Quadratic Dynamic Matrix Control Algorithm

G. N. Charos*
Y. Arkun †
School of Chemical Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0100
July 26, 1992

Abstract

A decentralized version of Quadratic Dynamic Matrix Control algorithm is presented. Taking advantage of this new formulation it is possible to reduce the time required for the solution of otherwise large optimization problems. Examples demonstrate the performance and the CPU time requirements of the proposed algorithm.

1 Introduction

Quadratic Dynamic Matrix Control (QDMC) is a Model Predictive Control (MPC) algorithm which has been widely used in the industry. Its popularity is due to the fact that it can handle multivariable control problems with constraints, delays and control objectives changing over time. It uses discrete step response models which are intuitive to the engineers. QDMC solution is based on a quadratic program which is solved on-line.

*Currently with Amoco Chemical Company, Naperville, Illinois
†To whom all correspondence should be addressed
For problems with many constrained inputs and outputs this can be computationally time consuming. In this paper we give a new formulation of the QDMC algorithm which decomposes the original problem to smaller subproblems which can be solved independently and in parallel, with possible savings in computation time. The paper is structured as follows. Section 2 introduces the notation and presents the necessary background on QDMC. In Section 3 decentralized version is derived and some of its properties are discussed. Finally a numerical example is given in Section 4.

2 Notation and Background

Readers who are not familiar with the notation are referred to Prett and García (1988). The input / output behavior of the plant is modeled by a finite step-response model:

\[ Y(k) = \sum_{i=1}^{N} S^i \Delta U(k-i) + Y^*(k) \]  

\[ Y^*(k) = Y^*(k-1) + S^N \Delta U(k-N-1) \]  

The variables in equations (1) and (2) are defined below. Outputs at the sampling time \( k \):

\[ Y(k) = [y_1(k) \ y_2(k) \ ... \ y_n(k)]^T \]  

Input changes at the sampling time \( k-i \):

\[ \Delta U(k-i) = [\Delta u_1(k-i) \ \Delta u_2(k-i) \ ... \ \Delta u_n(k-i)]^T \]  

where

\[ \Delta u_j(k-i) = u_j(k-i) - u_j(k-i-1) \]
The variable $S^i$ is the $i$-th step response matrix given by

$$
S^i = \begin{bmatrix}
  s_{1,1}^i & s_{1,2}^i & \cdots & s_{1,n_u}^i \\
  s_{2,1}^i & s_{2,2}^i & \cdots & s_{2,n_u}^i \\
  \vdots & \vdots & \ddots & \vdots \\
  s_{n_y,1}^i & s_{n_y,2}^i & \cdots & s_{n_y,n_u}^i
\end{bmatrix}
$$

(6)

where $s_{i,p}^i$ is the $i$-th step response coefficient between output $y_j$ and input $u_p$. Finally $N$ is the total number of step response matrices used to describe the dynamic input/output map.

Equations (1) and (2) give the current values of the outputs from the past values of the inputs. In a similar fashion one can predict the future output values:

$$
\begin{bmatrix}
  Y(k+1) \\
  Y(k+2) \\
  \vdots \\
  Y(k+P)
\end{bmatrix}
= \begin{bmatrix}
  S^1 & 0 & \cdots & 0 \\
  S^2 & S^1 & 0 & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  S^P & S^{P-1} & \cdots & S^{P-M+1}
\end{bmatrix}
\begin{bmatrix}
  \Delta U(k) \\
  \Delta U(k+1) \\
  \vdots \\
  \Delta U(k+M-1)
\end{bmatrix}
= \mathbf{A}\Delta U(k)
$$

(7)

$$
\begin{bmatrix}
  \Delta U(k-1) \\
  \Delta U(k-2) \\
  \vdots \\
  \Delta U(k-N+1)
\end{bmatrix}
$$

$$
\begin{bmatrix}
  Y^*(k+1) \\
  Y^*(k+2) \\
  \vdots \\
  Y^*(k+P)
\end{bmatrix}
= \begin{bmatrix}
  W(k) \\
  W(k) \\
  \vdots \\
  W(k)
\end{bmatrix}
\begin{bmatrix}
  W(k) \\
  W(k) \\
  \vdots \\
  W(k)
\end{bmatrix}
= \mathbf{W}(k)
$$

The number of future predicted outputs is $P$; the vector of past input changes is denoted by $\Delta U(k-1)$; the vector of future input changes is $\Delta U(k)$. It is assumed that the input
remains constant after it changes \( M \) times. For completeness unmeasured disturbance effect \( W(k) \) is also included. In general it is assumed that the disturbance remains constant into the future i.e.

\[
W(k + i) = W(k) \quad i = 1, 2, \ldots, P
\]

where the current disturbance value is estimated from the difference between the plant and model outputs:

\[
W(k) = Y_p(k) - Y(k)
\]

QDMC arrives at the control inputs through the solution of the following optimization problem (QP):

\[
\min_{\Delta U(k)} J = ((Y_s(k + 1) - Y(k + 1))^T \Gamma^T \Gamma + \Delta U^T(k) \Lambda^T \Lambda \Delta U(k))
\]

s.t.

\[
Y(k + 1) = A\Delta U(k) + B\Delta U(k - 1) + Y^*(k) + W(k)
\]

and

\[
U_{\min}(k + l) \leq U(k + l) \leq U_{\max}(k + l) \quad l = 0, \ldots, M - 1
\]

\[
\Delta U_{\min}(k + l) \leq \Delta U(k + l) \leq \Delta U_{\max}(k + l) \quad l = 0, \ldots, M - 1
\]

\[
Y_{\min}(k + 1) \leq Y(k + 1) \leq Y_{\max}(k + 1) \quad m = 1, \ldots, P
\]

\( \Gamma \) and \( \Lambda \) are diagonal weighting matrices, and \( Y_s \) is the vector of future setpoints. Thus QDMC tries to find \( M \) future inputs to minimize the sum of the squares of the weighted output errors plus the weighted control effort. Once the optimal \( \Delta U(k) \) is computed only the first change \( \Delta U(k) \) is implemented and the calculation is repeated at the next sampling time.

It can be easily shown that the above objective function is quadratic in \( \Delta U \) and is solved by Quadratic Programming.
3 Decentralization of QDMC

The objective function (10) is separable i.e.

$$J = \sum_{i=1}^{n} J_i$$  \hspace{1cm} (15)

where

$$J_i = (Y_{s,i} - Y_i)^T \Gamma_i^T \Gamma_i (Y_{s,i} - Y_i) + \Delta U_i^T \Lambda_i^T \Lambda_i \Delta U_i$$  \hspace{1cm} (16)

Here $Y$ and $U$ are partitioned into $n$ subvectors of outputs and inputs respectively:

$$Y^T = [Y_1 \ Y_2 \ ... \ Y_n]^T \quad U^T = [U_1 \ U_2 \ ... \ U_n]^T$$  \hspace{1cm} (17)

Then $J_i$ denotes the objective function for the subsystem $(Y_i, U_i)$. Although the overall objective function is separable, the optimization itself cannot be expressed by $n$ independent smaller optimizations. This is due to the prediction eqn (11) which introduces the coupling between the subproblems. In the following a particular decomposition is given to partially eliminate this coupling. It is shown that the QP can be solved by $n$ smaller optimizations using this approach. The derivation starts from equation (11).

The Model Prediction Equation Revisited

For the sake of clarity and without loss of generality let us assume that $n_y = n_u = n$ and dim $Y_i = \text{dim} U_i = 1$. That is outputs and inputs are partitioned into $n$ scalar subsystems. Then for each output $q = 1, \ldots, n$ we get:

$$
\begin{bmatrix}
    y_q(k+1) \\
    y_q(k+2) \\
    \vdots \\
    y_q(k+P)
\end{bmatrix}
= \begin{bmatrix}
    s^1_{q,q} & 0 & \cdots & 0 \\
    s^2_{q,q} & s^1_{q,q} & 0 & 0 \\
    \vdots & \vdots & \vdots & 0 \\
    s^P_{q,q} & s^{P-1}_{q,q} & \cdots & s^{P-M+1}_{q,q}
\end{bmatrix}
\begin{bmatrix}
    \Delta u_q(k) \\
    \Delta u_q(k+1) \\
    \vdots \\
    \Delta u_q(k+M-1)
\end{bmatrix}
$$

\hspace{1cm} \text{where } A_{q,q}
If we identify subsystem $q$ with the input-output pair $(u_q, y_q)$, the terms in eqn (19) are interpreted as follows:

i. is the effect that the future input to subsystem $q$ will have on output $q$. 

Or

\[ Y_q(k + 1) = A_{q,q} \Delta U_q(k) + B_{q,q} \Delta U_q(k - 1) + \sum_{j=1, j \neq q}^{n} A_{q,j} \Delta U_j(k) \]

\[ + \sum_{j=1, j \neq q}^{n} B_{q,j} \Delta U_j(k - 1) + Y_{q,q}^*(k + 1) + \sum_{j=1, j \neq q}^{n} Y_{q,j}^*(k + 1) + W_q(k) \]
ii. is the effect that the past input to subsystem q will have on output q.

iii. is the effect that the future inputs to all other subsystems will have on output q.

iv. is the effect that the past inputs to all other subsystems will have on output q.

v. is the effect that the initial condition of subsystem q will have on the output of subsystem q.

vi. is the effect that the initial conditions of all other subsystems will have on the output of subsystem q.

vii. is the effect that the disturbance will have on the output of subsystem q.

The fundamental assumption made in our formulation of decentralized QDMC is the following:

Each subsystem q assumes that each other subsystem i (i ≠ q) will keep the inputs which it has implemented at the previous sampling time constant for the next prediction horizon.

This assumption implies that part (iii) of eqn (19) is identically zero which results in the decomposition of QP into n suboptimizations:

\[
\min_{\Delta U_q} J_q \quad q = 1, \ldots, n
\]  

s.t.

\[
Y_q(k+1) = A_{q,q} \Delta U_q(k) + Y_q^{past}(k+1) + W_q(k)
\]  

where

\[
Y_q^{past}(k+1) = \sum_{j=1}^{n} B_{q,j} \Delta U_j(k - 1) + \sum_{j=1}^{n} Y_{q,j}^{past}(k + 1)
\]  

and

\[
U_{q,\text{min}}(k + l) \leq U_q(k + l) \leq U_{q,\text{max}}(k + l) \quad l = 0, \ldots, M - 1
\]  

\[
\Delta U_{q,\text{min}}(k + l) \leq \Delta U_q(k + l) \leq \Delta U_{q,\text{max}}(k + l) \quad l = 0, \ldots, M - 1
\]  

\[
Y_{q,\text{min}}(k + 1) \leq Y_q(k + 1) \leq Y_{q,\text{max}}(k + 1) \quad m = 1, \ldots, P
\]
The term $J_q^{past}$ predicts the effect of past inputs on future outputs. Therefore it can be easily computed before each optimization is performed. Similarly $W_q$ can be computed from the plant measurement and past inputs. Once these two quantities are computed, suboptimizations for the future control inputs are carried out in a completely independent fashion. This is illustrated in Figure 1 where each subsystem implements the following strategy:

At sampling time $k$ collect output measurement $y_q$ and receive information about the inputs implemented by all subsystems at $k-1$; compute $J_q^{past}(k+1)$ and $W(k)$; solve the optimization for $\Delta U_q$ using a QP and implement $\Delta u_q(k)$. Finally make this value available to all other subsystems and go to the next sampling time.

It is important to note that our decentralization is in the prediction of the effect of future inputs. For this purpose the fundamental assumption is invoked and the computation of the optimal future inputs is decentralized. The terms $J_q^{past}$ and $W_q$ are easily calculated from a full model without any need for decentralization and can be viewed as constant inputs to the optimization.

We will now address some of the properties of decentralized QDMC. This will be done using the IMC structure in Figure 2 without including the constraints. The objective here is to give some insight into the operation of the algorithm without complicating the analysis. Later simulation results will include constraints. First we want to derive the underlying transfer functions starting with the model in Fig. 2. The effect of the $j$-th input on the $i$-th input can be expressed in terms of the impulse response coefficients:

$$y_{i,j}(k+1) = h_{1,j}^i u_j(k) + h_{2,j}^i u_j(k-1) + h_{3,j}^i u_j(k-2) + \ldots + h_{N,j}^i u_j(k-N+1)$$

or in z-domain:

$$z y_{i,j}(z) = (h_{1,j}^i + h_{2,j}^i z^{-1} + h_{3,j}^i z^{-2} + \ldots + h_{N,j}^i z^{-N}) = H_{i,j}(z)$$

where $H_{i,j}(z)$ denotes the individual transfer function. Thus the model simulated in parallel
with the plant is given by

\[ Y(z) = z^{-1} H(z) U(z) \]  

(28)

As noted earlier, since this model can be easily used to calculate \( \mathcal{Y}^{\text{past}} \) and \( \mathcal{W} \), it is not decentralized i.e \( H(z) \) is full and not diagonal. In this respect calculation of \( \mathcal{Y}^{\text{past}} \) and \( \mathcal{W} \) is identical to that of classical centralized QDMC.

Now let's focus on the predictive model used inside the controller. Note that in the general case there will be a delay between \( y_i \) and \( u_j \) say \( \kappa_{i,j} \). This implies that the first \( \kappa_{i,j} \) impulse coefficients will be zero and eqn (27) will be written as:

\[ z y_{i,j}(z) = z^{-\kappa_{i,j}} \left( \sum_{n=0}^{\infty} h_{i,j}^{\kappa_{i,j}+n} z^{-n} \right) u_j(z) \]  

(29)

The output of each subsystem \( i \) is then given by

\[ z^{\kappa_{i,i}+1} y_i(z) = H_{i,i} u_i + \sum_{j=1,i \neq j}^n z^{\Delta \kappa_{i,j}} H_{i,j} u_j \]  

(30)

where

\[ \Delta \kappa_{i,j} = \kappa_{i,i} - \kappa_{i,j} \]  

(31)

The summation expresses the coupling with the other subsystems. We will now show how this is modified by the fundamental assumption we imposed earlier. Each term under the summation is given by

\[ z^{\Delta \kappa_{i,j}} H_{i,j} u_j(z) = z^{\kappa_{i,i}+1} y_{i,j} \]

\[ = (h_{i,j}^{(\kappa_{i,j}+1)} z^{(\kappa_{i,j})} + h_{i,j}^{(\kappa_{i,j}+2)} z^{(\Delta \kappa_{i,j}+1)}) z^{(-1)} + \cdots + h_{i,j}^{(\kappa_{i,j}+\Delta \kappa_{i,j}+1)} z^{(-2)} + \cdots + u_j(z) \]  

(32)

When \( \Delta \kappa_{i,j} \) is positive, the first \( \Delta \kappa_{i,j} + 1 \) terms (up to the \( z^{-1} \) term) will require the future inputs \( u_j(k), u_j(k+1), \ldots, u_j(k+\Delta \kappa_{i,j}) \) to predict \( y_{i,j}(k+\kappa_{i,i}+1) \). The fundamental assumption sets all these future inputs to their last implemented values \( u_j(k-1) \) for all
\( j \neq i \). This means that eqn (32) is modified to:

\[
z^{\kappa_{i,j}+1} y_{i,j} = z^{\Delta \kappa_{i,j}} H_{i,j} - z^{\Delta \kappa_{i,j}} \sum_{l=1}^{\Delta \kappa_{i,j}+1} h_{i,j}^{\kappa_{i,j}+1} z^{-1+l} + \left( \sum_{l=1}^{\Delta \kappa_{i,j}+1} h_{i,j}^{\kappa_{i,j}+1} \right) z^{-1} u_j \tag{33}
\]

\[
eq \hat{H}_{i,j}(z) u_j \tag{34}
\]

This modification basically replaces the first \( \Delta \kappa_{i,j} + 1 \) terms in eqn (32) by the last summation in (33) which requires knowledge of the past input values only and no future values. For delay free systems, only the sampling time has to be taken care of. Therefore \( \Delta \hat{H}_{i,j} = (-1 + z^{-1}) h_{i,j}^{\kappa_{i,j}+1} \). In the case of equal delays \( \Delta \kappa_{i,j} = 0 \) and \( \Delta \hat{H}_{i,j} = (-1 + z^{-1}) h_{i,j}^{\kappa_{i,j}+1} \).

Finally if \( \Delta \kappa_{i,j} < 0 \), \( \Delta \hat{H}_{i,j} = 0 \) and the original transfer function remains unchanged. Substituting eqn (34) into eqn (30) one gets the final predictive model used in the decentralized QDMC calculations:

\[
\begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_n
\end{bmatrix} =
\begin{bmatrix}
z^{-(\kappa_{1,1}+1)} & 0 & \cdots & 0 \\
0 & z^{-(\kappa_{2,2}+1)} & \cdots & \cdots \\
\vdots & \vdots & \ddots & \vdots \\
0 & \cdots & \cdots & z^{-(\kappa_{n,n}+1)}
\end{bmatrix}
\begin{bmatrix}
\hat{H}_{1,1} \\
\hat{H}_{1,2} \\
\vdots \\
\hat{H}_{n,1} \end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
\vdots \\
u_n
\end{bmatrix} = diag \left( z^{-(\kappa_{i,i}+1)} \right) H_{dec}(z) U(z) \tag{35}
\]

which is in the delay factorization form (Ricker, 1985). In general factored delays \( \kappa_{i,i} \) do not have to be the minimum delays.

It is instructive to compare this decentralized QDMC with the classical centralized
QDMC. Without decentralization the predictive model consists of equations (30). Or

\[
\begin{bmatrix}
  y_1 \\
  y_2 \\
  \vdots \\
  y_n
\end{bmatrix} = \begin{bmatrix}
  z^{-(\kappa_{1,1}+1)} & 0 & \ldots & 0 \\
  0 & z^{-(\kappa_{2,2}+1)} & \ldots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & \ldots & \ldots & z^{-(\kappa_{n,n}+1)}
\end{bmatrix} \begin{bmatrix}
  \hat{H}_{1,1} & z^{\Delta \kappa_{1,2}} \hat{H}_{1,2} & \ldots & z^{\Delta \kappa_{1,n}} \hat{H}_{1,n} \\
  \hat{H}_{2,2} & \ldots & \ldots & \ldots \\
  \vdots & \vdots & \ddots & \vdots \\
  \hat{H}_{n,n} & \ldots & \ldots & \hat{H}_{n,n}
\end{bmatrix} \begin{bmatrix}
  u_1 \\
  u_2 \\
  \vdots \\
  u_n
\end{bmatrix}
\]

\[= \text{diag} \left( z^{-(\kappa_{1,1}+1)} \right) H_{cen}(z)U(z) \tag{38}\]

If one can pair the inputs and the outputs so that minimum delays can be factored out for each output, the above two models (35) and (37) become identical. This is due to the fact that in this case \( \Delta \kappa_{i,j} < 0 \) and by construction \( \Delta \hat{H}_{i,j} = 0 \). It then follows that \( \hat{H}_{i,j} = z^{\Delta \kappa_{i,j}} \hat{H}_{i,j} \) and \( H_{cen}(z) = H_{dec}(z) \). In addition, if one uses for the tuning parameters \( P = M \) and \( \Lambda = 0 \), the controllers are the inverse controllers which are identical i.e \( G_c = H_{cen}^{-1}(z) = H_{dec}^{-1}(z) \). Systems which can be decentralized with minimum delay factorization are called strictly balanced systems (García and Morari, 1985). The above analysis shows that our decentralized QDMC and the classical centralized QDMC can give identical results for such systems. For unbalanced systems and for systems where minimum delay factorization is not sought, decentralized QDMC still applies with its predictive model in its most general form (35). Like its centralized counterpart, decentralized QDMC provides zero steady state offset in general since \( \Delta \hat{H}_{i,j}(z = 1) = 0 \).

4 Examples

The first example is a small system which will be used to verify some of the theory discussed and give comparisons with other MPC algorithms. The second example will treat a larger system and include results on CPU requirements.

**Example 1**

Consider the Wood and Berry distillation column (Wood and Berry, 1973). The top composition is paired with reflux and bottoms composition is paired with steam flowrate. This system is strictly balanced and its transfer function has the following structure where only
delays are explicitly shown:
\[
\begin{bmatrix}
  z^{-2} f_{1,1} & z^{-4} f_{1,2} \\
  z^{-8} f_{2,1} & z^{-4} f_{2,2}
\end{bmatrix}
\]  \hspace{1cm} (39)

Figure 3 shows the effect of prediction horizon on set point response. Since the system is strictly balanced, with \( P = M \) one gets perfect tracking in the absence of model/plant mismatch as one would get in centralized QDMC. Also as \( P \) increases, the response becomes slower as expected. The effects of other tuning parameters are similar to those observed in classical centralized QDMC and not shown here.

It is also interesting to compare our decentralized QDMC with a fully decentralized IMC controller. To avoid confusion full decentralization here means the model used in parallel with the plant i.e. \( H(z) \) and the predictive model used to compute the control action i.e. \( H_{cen}(z) \) are both kept diagonal by setting their off-diagonal elements to zero. This is the scheme known as decentralized IMC or predictive controller reported in the literature. Note that this controller is very different than the one we propose here in that our \( H(z) \) (see eqn (28)) and predictive model (see eqn (35)) are both full matrices. Note that despite the full structure of these matrices the computation of future control moves can still be performed in a decentralized fashion as described earlier. Figures 4 and 5 compare the two controllers. As expected decentralized QDMC performs better. Looking at the system's delay structure above, it is clear that with full decentralization, the controller which manipulates the reflux will react to the setpoint change of tops composition at time zero as shown. However the bottoms controller will not react until it realizes the disturbance caused by the first controller through feedback 8 sampling times later (the time delay entry \((2,1)\) in the above transfer matrix). However, for the decentralized QDMC, steam will start reacting the following sampling time (it takes only one sampling time to receive information that top composition loop has acted). This explains the apparent difference in performance between the two control algorithms. Due to the particular tuning chosen for this simulation, the steam reacts very cautiously for the first three sampling times; its moves are hardly detectable in Figure 4. However it starts reacting rather vigorously after the third sampling time.
Example 2

The continuous dynamic model of the fluid catalytic cracker unit used in this section was originally presented by Grosdidier et al. (1988) and it is given below in its discretized form with \( T = 1 \) min. The variables are defined in Table 2.

\[
\begin{bmatrix}
    y_1(z) \\
    y_2(z) \\
    y_3(z)
\end{bmatrix} = \begin{bmatrix}
    0.0127z^{-3} & -0.0013z^{-3} & -0.0012z^{-3} \\
    1-0.8187z^{-1} & 1-0.8669z^{-1} & 1-0.8825z^{-1} \\
    -0.0917z^{-3} & -0.0202z^{-9} & 0.0019z^{-1} \\
    1-0.8825z^{-1} & 1-0.9775z^{-1} & 1-0.9048z^{-1} \\
    -0.0713z^{-37} & -0.0198z^{-1} & 0.0256z^{-3} \\
    1-0.9131z^{-1} & 1-0.9802z^{-1} & 1-0.9200z^{-1}
\end{bmatrix}
\begin{bmatrix}
    u_1(z) \\
    u_2(z) \\
    u_3(z)
\end{bmatrix}
\] (40)

The pairings are chosen as \((y_1, u_1), (y_2, u_2), (y_3, u_3)\). Among the possible pairings with positive Niederlinski Index, these pairings have the most reasonable time delays. Note that this is not a strictly balanced system.

The simulation run is shown in Figure 6; the tuning parameters used for this simulation are given in Table 3 and the time statistics are given in Table 4. This simulation run shows a step setpoint change of 10 °F to the regenerator temperature subject to constraints on the manipulated variables given in Table 3.

The primary motivation behind the decentralized QDMC implementation has been to reduce the long time required for the solution of the centralized problem in the presence of a large number of constraints. The quality of the QP computer code as far as computational robustness, reliability and accessibility are concerned is of equal importance in an on-line implementation. For this reason, in this work we used the QPSOL package developed at Stanford University. We feel that the types of time savings observed in this study with the QPSOL package due to decentralization can be also observed when other optimization packages and/or algorithms are used.

This example is intended to attract attention to the CPU time required to solve the optimization problems. These results are presented in Table 4. The table entries under the columns "Optimization" are the maximum CPU times in seconds that it took the optimizer to reach a solution at one sampling time. All the simulations were performed on a VAXstation II computer. The maximum time it took to solve the centralized optimization problem is always considered to be the reference speed for the particular simulation and it is assigned the value of 1. The entries under the "speed" column of the other systems
are calculated by dividing the time of the centralized configuration by the maximum time among the times it took any of the subsystems according to the expression:

\[
\text{speed}_{\text{configuration}} = \frac{(CPU)_{\text{centralized}}}{\max_j(CPU)_{\text{configuration}}} \quad j = 1, \ldots, NS
\]  

This entry actually indicates how fast the decentralized subsystems can reach their optimum as compared to the centralized problem.

In our current implementational environment all the decentralized calculations are performed sequentially on one computer. In particular the "driver" program requests solutions of the subsystems' optimization problems in a prearranged order. Nevertheless, the only information shared by the subsystems is the past inputs as has been already described in detail. In a real process environment, provided there exists a dedicated communications network for data collection and archival, and provided that such a network allows different computers to exchange information in real time, it is possible to separate the subsystem optimization problems and install them on different computers, thus utilizing a parallel computing architecture.

The simulations show that decentralized QDMC provides tractable dynamic behaviour with less CPU time than centralized QDMC. As the number of constraints increase the time savings also increase.

5 Conclusion

In this paper we have presented a decentralized version of the Quadratic Dynamic Matrix Control algorithm. It is shown that the original quadratic programming can be decomposed to smaller QP's which can solved independently and in parallel.

Acknowledgements- The financial support of National Science Foundation is gratefully acknowledged.

References

[1] C. García and M. Morari. Internal Model Control. 3. Multivariable control law com-


<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Operating Steady State</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1$</td>
<td>Flue Gas O$_2$</td>
<td>2.5 %</td>
</tr>
<tr>
<td>$y_2$</td>
<td>Regen. bed. Temp</td>
<td>1320°F</td>
</tr>
<tr>
<td>$y_3$</td>
<td>Flue gas temp.</td>
<td>1355°F</td>
</tr>
<tr>
<td>$u_1$</td>
<td>Combustion air flow</td>
<td>300 MLB/HR</td>
</tr>
<tr>
<td>$u_2$</td>
<td>Recycle oil flow</td>
<td>80 BPH</td>
</tr>
<tr>
<td>$u_3$</td>
<td>Feed flow</td>
<td>2000 BPH</td>
</tr>
</tbody>
</table>

Table 2: The fluid catalytic cracker unit variable summary.

---

Table 1: Tuning parameters and constraints for Example 1

<table>
<thead>
<tr>
<th>Tuning Parameters</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 100$</td>
<td>$P$</td>
</tr>
<tr>
<td>Output 1</td>
<td>30</td>
</tr>
<tr>
<td>$\Delta$ Input 1</td>
<td>-</td>
</tr>
<tr>
<td>Input 1</td>
<td>-</td>
</tr>
<tr>
<td>Output 2</td>
<td>30</td>
</tr>
<tr>
<td>$\Delta$ Input 2</td>
<td>-</td>
</tr>
<tr>
<td>Input 2</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: Tuning parameters and constraints for Example 1
Tuning Parameters Constraints

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>M</th>
<th>$\Gamma$</th>
<th>$\Lambda$</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output 1</td>
<td>50</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>$-\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>-</td>
<td>20</td>
<td>-</td>
<td>0.8</td>
<td>$-\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>Input 1</td>
<td>-</td>
<td>20</td>
<td>-</td>
<td>0.8</td>
<td>$-\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>Output 2</td>
<td>50</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>$-\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>-</td>
<td>20</td>
<td>-</td>
<td>0.8</td>
<td>-2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Input 2</td>
<td>-</td>
<td>20</td>
<td>-</td>
<td>0.8</td>
<td>-10.0</td>
<td>$\infty$</td>
</tr>
<tr>
<td>Output 3</td>
<td>50</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>$-\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>-</td>
<td>20</td>
<td>-</td>
<td>0.8</td>
<td>-5.0</td>
<td>5.0</td>
</tr>
<tr>
<td>Input 3</td>
<td>-</td>
<td>20</td>
<td>-</td>
<td>0.8</td>
<td>-40.0</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>

Table 3: Tuning parameters and constraints for Example 2.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Optimization for subsystem 1 CPU sec.</th>
<th>Optimization for subsystem 2 CPU sec.</th>
<th>Optimization for subsystem 3 CPU sec.</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized</td>
<td>13.1</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Decentralized</td>
<td>0.9</td>
<td>2.0</td>
<td>0.8</td>
<td>6.6</td>
</tr>
</tbody>
</table>

Table 4: Time requirements for Example 2
Figure 1: Architecture of Decentralized QDMC

Figure 2: IMC Structure
Figure 3: Example 1. Effect of $P$. $\lambda = 0$, $M = 1$, No constraints
Figure 4: Example 1. Decentralized QDMC using the data in Table 1.

Figure 5: Example 1. Fully Decentralized QDMC using the data in Table 1.
Figure 6: Example 2. Decentralized QDMC using the data in Table 3.
1 Tutorial

1.1 Introduction

Control configuration design (ccd) addresses the issues of (1) selecting measurements and manipulations for closed-loop control and (2) partitioning the feedback of the selected measurements to the selected manipulations. Effective synthesis of control configurations requires a systematic methodology. The CCD TOOLBOX adopts the indirect synthesis method of Reeves, Nett, and Arkun [1, 2].

Indirect synthesis of control configurations requires that configurations be postulated and then analyzed using some established criteria. The criteria employed by the toolbox are based on the practical theory contained in [1].

This tutorial will guide you through the essential steps of the indirect synthesis method of control configuration design. The tutorial is therefore divided into sections that describe each step individually. The theory behind each step will be addressed briefly but the emphasis will be on explaining the available functions and the relevant notation. The tutorial concludes with a section that leads you through a typical control configuration design session.

Before proceeding, some general notational comments should be made. Ordinarily, uppercase letters will denote matrices while lowercase letters will be used for both vectors and scalars. Unless stated otherwise, vectors are assumed to be entered as row vectors. Text that appears in block letters, such as select and partit, will indicate MATLAB variables, commands, and function names.

Many of the functions require that the inputs be entered in a very specific form. Most of the functions also return carefully formatted results. You should therefore be familiar with what the CCD TOOLBOX denotes as ccd convention for formatting. Any ccd conventions employed by a function will be described in the appropriate section of this tutorial. The location of these descriptions is indicated in the text by the boldface phrase ccd convention.

On-line help can be obtained for any of the CCD TOOLBOX functions by entering help function name.

---

1The user is referred to previous chapters for an in-depth treatment of the theory behind the indirect synthesis method.
1.2 System Representation

The control configuration design toolbox is designed for application to finite-dimensional, linear, time-invariant (FDLTI) systems that are either represented in the state-space form:

\[\begin{align*}
\dot{x} &= Ax + Bu \\
y &= Cx + Du
\end{align*}\]  

(1) (2)

or as a transfer function matrix:

\[y = G(s)u\]  

(3)

where in each case the variables \(y\) and \(u\) denote vectors of candidate measured and manipulated variables, respectively. Limited application is possible if only the steady-state gain matrix, \(G(0)\), is available.

The fundamental operating element in the toolbox is a “system matrix,” evaluated at a particular frequency, \(\omega\). Typically the system matrix will be the transfer function matrix, computed via the relationship

\[G = C(sI - A)^{-1}B + D\]  

(4)

where \(s = j\omega\). The function

\[\text{tfw}(A,B,C,D,w)\]

computes \(G\), given the state-space matrices \(A,B,C,D\) and the frequency \(w\).

The notation \(yx\) will typically indicate a row vector that contains information about the process measurements. Similarly, \(ux\) will usually denote a row vector that contains information about the process manipulations. The information that the vectors contain will depend on the particular function being employed.
1.3 Scaling

Much of the theory available for control configuration design relies upon the induced 2-norm of the system matrix, interpreted as a physical gain:

\[
\sigma(G) = \max_{y \neq 0} \frac{\|y\|_2}{\|u\|_2} = \max_{u \neq 0} \frac{\|Gu\|_2}{\|u\|_2}
\]

(5)

But this norm is quite dependent upon the scaling of \(y\) and \(u\). Therefore, before applying any criteria that incorporate \(\sigma(G)\), the system matrix must be scaled appropriately. Otherwise, any results obtained from these criteria will be highly suspect.

Scaling corresponds to designating units for the system variables. If the unscaled variables of the system are contained in \(u\) and \(y\), then the scaled variables can be represented as

\[
u_s = S_u u \quad y_s = S_y y
\]

(6)

where the matrices \(S_u\) and \(S_y\) are real, positive, and diagonal. The scaled system matrix will then be

\[
G_s = S_y G S_u^{-1}
\]

(7)

The CCD TOOLBOX provides two functions to aid the designer in scaling the system matrix. The function

\[
scale(G, y_s, u_s)
\]

returns the scaled system matrix, given \(G\) and vectors that describe the scaling on the measurements \((y_s)\) and manipulations \((u_s)\). The elements of \(y_s\) and \(u_s\) should be the Smallest Significant Deviations of Interest for the corresponding variables, as described in [2]. The diagonal elements of \(S_y\) and \(S_u\) above will be the inverse of the elements of these vectors.

If the proper scaling for all of the manipulations is not known, the function

\[
us = ussdoi(G, y_d, u_d)
\]

provides the designer with a rough first estimate. The function \(ussdoi\) is based on a heuristic approach to the Smallest Significant Deviation of Interest (SSDOI) Method.

\[^2\text{Note that the state variables } x \text{ of the system can also be scaled, but this scaling cancels and does not appear in the system matrix.}\]
of scaling, as described in [2]. You must specify the steady-state system matrix \((G)\) and the deviation vectors for the measurements \((yd)\) and the manipulations \((ud)\). Ccd convention dictates that these vectors contain any SSDOI's that are known and \(\text{Inf}\) in the location of those SSDOI's that must be obtained. At the very least, one element of \(yd\) must be finite. However, the more measurements SSDOI's that are supplied to the function, the more reliable the results it produces. The function returns a vector \(us\) that has the same form as \(ud\), but with the \(\text{Inf}\)'s replaced by estimated SSDOI's. Note that \(ussdoi\) should only be considered a helpful guide and \textit{never} the final word on scaling system manipulations. That is, the results of \(ussdoi\) should always be checked against the designer's physical understanding of the system.

If some of the measurement SSDOI's are unknown, you are encouraged to follow the procedure mentioned in [2]. Examine the elements of each row of the matrix formed by multiplying \(G\) and \(ud\) (or \(us\) if \(ussdoi\) was employed). These elements should give an indication of a typical range that the SSDOI for the corresponding measurement should lie within. A final judgement with respect to the unknown measurement SSDOI's must be made in the context of the physical system.

1.4 System Size Reduction

If either or both of the dimensions \(m\) and \(n\) are quite large (usually greater than 10), then an exhaustive search through all possible subsystems and partitions may be prohibitive. In this case, the ccd systematic methodology recommends that the dimension(s) of the system be reduced via one or more of several proposed heuristics. All of the heuristics use either the condition number or the RGA as a guide for dimension reduction. The justification for using these quantities follows from the theory for measurement/manipulation selection set forth in [1]. Note that the properties of the heuristics vary, so that one may be better suited than another for a particular system. The order of presentation implies nothing about the relative merits of the heuristics — you are encouraged to study each of them in the context of your own system before implementing any one of them.

Note that reducing the system size in preparation for an exhaustive application of the selection criteria may preclude the identification of an optimal selected subset of measurements and manipulations. Instead the designer may be led to a suboptimal selected subsystem. This is unfortunate, but unavoidable given present theoretical
and computational limitations. Note also that system size reduction should only be conducted in the context of the control objectives for the system under investigation. If necessary, the heuristics should be constrained to retain certain variables essential for satisfying the control objectives.

1.4.1 Heuristic 1

The first heuristic is based directly on one of the viability criteria for measurement/manipulation selection given in [1]. This heuristic is fairly versatile, since it may be applied to nonsquare systems and the amount of searching involved would rarely be prohibitive. However, you should be rather sure of your system scaling before employing this heuristic, since it uses the scale-dependent condition number.

The selection criteria of [1] employ necessary conditions for robust stability. One necessary condition requires that the condition number of the system lie below an upper bound specified by the control configuration designer. The heuristic follows a procedure for reducing the system size where only one measurement or manipulation is discarded after each step, based on the condition number of the resulting reduced system. A step can be summarized as follows:

Suppose an $M \times N$ system must be reduced to an $m \times n$ system. If $(M - m) \geq (N - n)$, then evaluate the measure for each of the $M$ subsystems with dimension $(M - 1) \times N$. These subsystems are obtained by deleting one row at a time from the system matrix. Choose to discard the row (corresponding to a measurement) that produces the subsystem with the lowest condition number. The system size will now be $(M - 1) \times N$.

If $(M - m) < (N - n)$, then evaluate the measure for each of the $N$ subsystems with dimension $M \times (N - 1)$. These subsystems are obtained by deleting one column at a time from the system matrix. Choose to discard the column (corresponding to a manipulation) that produces the subsystem with the lowest condition number. The system size will now be $M \times (N - 1)$.

Reduction from a $k \times l$ system to an $m \times n$ system will require $(k - m) + (l - n)$ steps.
The dimensions of a system can be reduced using this heuristic by invoking the function

\[ [yr, ur] = \text{rdcon}(G, m, n) \]

You must supply the system matrix \( G \) and the desired reduced dimensions, \( m \) and \( n \). The function \text{rdcon} repeatedly calls two other functions, \text{concol} and \text{conrow}, that eliminate one column or row at a time from the system matrix. The function returns the two vectors \( yr \) and \( ur \) which contain the indices of the measurements and manipulations contained in the reduced system.

If certain measurements and manipulations must be included in the reduced system, the row/column indices of these variables may be specified in the bias vectors \( yb \) and \( ub \) and the function

\[ [yr, ur] = \text{rdcon}(G, m, n, yb, ub) \]

employed. In this case, \( yr \) will contain \( yb \) and \( ur \) will contain \( ub \).

1.4.2 Heuristic 2

The second heuristic is also based directly on one of the criteria of [1]. While this heuristic has the advantage of being based on a scale-independent measure, it is restricted in application to square systems where a square reduced system is also desired.

A necessary condition for robust stability requires that the quantity

\[ 2 \max\{\|\Lambda(G)\|_1, \|\Lambda(G)\|_\infty\} - 1 \]

lie below a designer-specified upper bound, where \( \Lambda(G) \) is the RGA of the system matrix. The heuristic uses this RGA quantity as a basis for eliminating one measurement and one manipulation in each step of the procedure. A step can be summarized as follows:

Suppose a size \( N \) system must be reduced to a size \( n \) system. Evaluate the RGA quantity for each of the \( N^2 \) size \( N - 1 \) subsystems obtained by deleting one row and one column of the system matrix. Choose to discard the row (corresponding to a measurement) and column (corresponding to a
manipulation) that produce the subsystem with the lowest RGA quantity. The system size will now be $N - 1$.

Reduction from a size $m$ system to a size $n$ system will require $(m - n)$ steps.

The dimensions of a system can be reduced using this heuristic by invoking the function

$$[y_r, u_r] = r\text{drga}(G, n)$$

You must supply the system matrix $G$ and the desired reduced dimension $n$. The function returns the two vectors $y_r$ and $u_r$ which contain the indices of the measurements and manipulations contained in the reduced system.

If certain measurements and manipulations must be included in the reduced system, the row/column indices of these variables may be specified in the bias vectors $y_b$ and $u_b$ and the function

$$[y_r, u_r] = r\text{drga}(G, n, y_b, u_b)$$

employed. In this case, $y_r$ will contain $y_b$ and $u_r$ will contain $u_b$.

### 1.4.3 Heuristic 3

The third heuristic employs the relative gain array of the system matrix. The advantage of this method stems from the scale-independence of the RGA. The results of employing this heuristic will be the same whether the system has been scaled properly or not. The disadvantage of this method lies in the fact that application is restricted to square systems. This restriction is mandated by the theoretical justification for the heuristic, which will be outlined here briefly.

It has recently been shown [3, 4] that a square system matrix $G$ will become singular if a single element $g_{ij}$ is perturbed to be $g_{ij}(1 - \frac{1}{\lambda_{ij}})$, where $\lambda_{ij}$ is the RGA element that corresponds positionally to $g_{ij}$. Thus a large RGA element implies that only a small relative perturbation in the corresponding element of the system matrix will lead to system singularity. This in turn leads to concern about system performance in the presence of uncertainty. Consequently, a method for reducing system size that focuses on eliminating system elements that correspond to large RGA elements could be quite prudent.
A heuristic based on these facts has therefore been formulated. The following steps should be repeated until the desired reduced system size is reached:

Calculate the RGA of the system matrix. Identify the row and column indices of the largest RGA element. Strike the row and column with corresponding indices from the system matrix and consider the reduced matrix to be the system matrix.

Note that the searching involved in this procedure is minimal: each iteration requires that only one RGA be computed and its elements examined.

The function

\[ [y_r, u_r] = rgelem(G, n) \]

implements this heuristic on a system matrix, given \( G \) and the reduced dimension \( n \). The output of this function is also two vectors, \( y_r \) and \( u_r \), containing the indices of the measurements and manipulations in the reduced system. As with the first two heuristics, if certain measurements and manipulations must be included in the reduced system, the row/column indices of these variables may be specified in the bias vectors \( y_b \) and \( u_b \) and the function

\[ [y_r, u_r] = rgelem(G, n, y_b, u_b) \]

employed.

1.5 Selection

The selection problem focuses on choosing measured and manipulated variables for closed-loop control. Given an \( m \times n \) system matrix \( G \), a square subsystem with dimension less than or equal to \( \min(m, n) \) will ultimately be selected.

The initial stages of the selection problem employ necessary conditions for viability in order to eliminate subsystems that will not be viable for the stated performance requirements. This approach requires that distinct selection subsets of measurements and manipulations be identified. The most straightforward method of doing this identifies subsets by the row and column indices of the system matrix \( G \) that correspond to the selected measurements and manipulations. Therefore, the ccd convention for representing a \( k \)-dimensional selected subset is a \( 2 \min(m, n) \)-dimensional row vector of the form
The first $k$ elements of the $\min(m,n)$-dimensional row vector $yi$ specify the row indices of the selected measurements. The first $k$ elements of the $\min(m,n)$-dimensional row vector $ui$ specify the column indices of the selected manipulations. Zeros should fill out the last $\min(m,n) - k$ elements of $yi$ and $ui$. Note that the ordering of the row/column indices is not significant for selection.

1.5.1 Generating Subsystems

Once the system size is within range for an exhaustive application of the criteria, all of the distinct square subsystems should be generated and evaluated. The function

$$S = \text{select}(sz)$$

generates all distinct square subsystems of a size $sz= [m \ n]$ system, where $m$ is the row dimension and $n$ is the column dimension. (If $m=n$ then $sz$ can be a scalar.) The function

$$S = \text{selsub}(k, sz)$$

generates only those distinct square subsystems with dimension $k$. If certain measurements and manipulations must be included in every selected subsystem, the row/column indices of these variables may be specified in the bias vectors $yb$ and $ub$ and appended as optional arguments to the above functions.

Each of these functions follows ccd convention in its representation of the subsystems. Specifically, they each return a matrix where each row has the form $[yi \ ui]$.

1.5.2 Evaluating Subsystems

All of the subsystems in a given set $S$ may be evaluated using the criteria of [1] by invoking the function

$$\text{rstab}(G, crit, bd, S)$$

You must supply the system matrix $G$ as well as the set $S$. Note that $S$ must follow ccd convention in its representation of the subsystems. Each row should have the form $[yi \ ui \ x]$. (The variable $x$ represents an optional additional column of
information that may be appended to the matrix, but will be ignored by the function.)
The variable crit specifies which criteria should be applied: crit=1 for the scale-independent RGA criterion, or crit=2 for the scale-dependent condition number criterion. The upper bound required by each criterion should be entered as bd. The function will return a matrix containing those subsystems from S that pass the specified criterion. Each row will have the form \([yi \; ui \; x]\), where \(x\) will be the quantity evaluated by the criterion.

Note that the set \(S\) may in fact be the output from a previous application of the function \(rstab\). By adopting this strategy, you may apply the criteria at a new frequency, and/or for a different value of bd, only to those subsystems that passed under the previous conditions. Similarly, you may choose to apply \(crit=2\) only to those subsystems that passed for \(crit=1\). This allows you to gauge the relative effectiveness of the two criteria without wasting unnecessary computations by using the full set \(S\) for both applications of \(rstab\).

If you wish to apply the criteria to all possible subsystems of the system matrix, without explicitly calling the function \(select\) in advance, then the argument \(S\) may be omitted. The function \(rstab(G, crit, bd)\) will then call \(select\) before proceeding to apply the specified criteria.

### 1.5.3 Filtering Subsystems

Once the criteria have been applied using \(rstab\), you may wish to filter the subsystems that pass. That is, you may inquire "which of the subsystems that passed included measurement a and manipulation b?". The function

\[
sfilter(S, rc, ind, num, fl)
\]

provides the answer to such queries. You supply the set of subsystems \(S\) and specify whether the filtering will be done by rows (\(rc=1\)) or columns (\(rc=2\)). (If filtering with respect to both rows and columns is desired, you may simply apply \(sfilter\) twice.)

The argument \(ind\) specifies the filter indices — those indices that you wish to either include or exclude, as determined by \(num\) and \(fl\). How many of these indices should be grouped together in each subsystem is specified by \(num\). Finally, the filter logic is set by \(fl\). A value of \(fl=1\) means that at most \(num\) of the indices should appear in each subsystem. A value of \(fl=0\) requires that exactly \(num\) of them
should appear, and \( f_1 = -1 \) indicates that at least \( \text{num} \) should appear. Therefore, if you choose \( \text{num} = 0 \) and set \( f_1 \) to be either 1 or 0, the function will identify only subsystems that exclude the specified filter indices.

Note that \textit{sfilter} is not the only function useful for filtering data. While \textit{sfilter} was structured particularly for the selection problem, \textsc{Matlab} contains a great variety of built-in functions that can aid with filtering the data for analysis.

1.6 Partitioning

The partitioning problem involves specifying how selected measurements should be fed back to selected manipulations for closed-loop control. It presupposes that an \( n \)-dimensional square subset of measurements and manipulations have been selected. This subset will be now be denoted \( G \), since it is the system matrix for partitioning. As in the selection problem, the initial stages of partitioning will employ necessary conditions for viability to eliminate nonviable partitions. Thus all distinct partitions must be identified using some standard convention.

A partition requires both a block structure and a variable rearrangement. Ccd convention will therefore dictate that a partition be represented by a \( 3n \)-dimensional vector of the form

\[
[b \ y_i \ u_i]
\]

The \( n \)-dimensional vector \( b \) specifies the block structure. Each element of \( b \) should be an integer corresponding to a diagonal block size. The block sizes must be consistently arranged in a nonincreasing order. Since the number of blocks will frequently be less than \( n \), the last few elements of \( b \) may be zeros. The sum of the elements of \( b \) must equal \( n \).

The \( n \)-dimensional row vectors \( y_i \) and \( u_i \) specify the variable rearrangement within the block structure. Each element of \( y_i \) is an integer corresponding to the row index of a measurement. The ordering of the integers 1 through \( n \) in \( y_i \) specifies a re-ordering of the rows of the system matrix \( G \). Similarly, each element of \( u_i \) is an integer corresponding to the column index of a manipulation. The ordering of the integers 1 through \( n \) in \( y_i \) specifies a re-ordering of the columns of the system matrix \( G \).
1.6.1 Generating Partitions

The CCD TOOLBOX supplies several functions to aide the designer in generating partitions. The function

\[ P = \text{partit}(n) \]

generates all distinct partitions of an \( n \)-dimensional system. The function

\[ P = \text{partsub}(b) \]

generates only those distinct partitions that have the block structure \( b \). Each of these functions follows ccd convention in its representation of the partitions. Specifically, they each return a matrix where each row has the form \([b \ yi \ ui]\).

These functions call several other functions that you may occasionally wish to employ. The first,

\[ \text{block}(n) \]

generates all of the distinct block structures that can be imposed on an \( n \)-dimensional system. A related function which is used by block is

\[ \text{blocksub}(n,b) \]

This function generates distinct block structures that have leading block sizes specified by the vector \( b \).

In order to perform the row and column rearrangements within a block structure that generate distinct partitions, \text{partit} and \text{partsub} call the functions

\[ \text{vrrow}(b) \]

and

\[ \text{vrcol}(b) \]

where the vector \( b \) specifies the block structure.
1.6.2 Evaluating Partitions

All of the partitions in a given set \( P \) may be evaluated using the criteria of [1] by invoking the function

\[
\text{cfpd}(G, \text{crit}, \text{dh}, P)
\]

This function applies necessary conditions for low cross-feed performance degradation to the partitions of \( G \) contained in the set \( P \). The variable \( \text{crit} \) specifies which criteria should be applied: \( \text{crit}=1 \) for the scale-independent RGA criterion, or \( \text{crit}=2 \) for the scale-dependent \( V(G) \) criterion. The maximum allowable cross-feed performance degradation should be entered as \( \text{dh} \).

Note that \( P \) must follow ccd convention in its representation of the partitions. Each row should have the form \([b \, yi \, ui \, x]\). (The variable \( x \) represents an optional additional column of information that may be appended to the matrix, but will be ignored by the function.) The function will return a matrix containing those subsystems from \( P \) that pass the specified criterion. Each row will have the form \([b \, yi \, ui \, x]\), where \( x \) will be the quantity evaluated by the criterion.

The set \( P \) may be the output from a previous application of the function cfpd. Thus you may apply the criteria at a new frequency, for a different value of \( \text{dh} \), or with a different setting for \( \text{crit} \) only to those partitions that passed under the previous conditions.

If you wish to apply the criteria to all possible partitions of the system matrix, without explicitly calling the function \( \text{partit} \) in advance, then the argument \( P \) may be omitted. The function \( \text{cfpd}(G, \text{crit}, \text{dh}) \) will then call \( \text{partit} \) before proceeding to apply the specified criteria. Be aware that the number of all possible partitions for a system may be quite large, so that the argument \( P \) should only be omitted if the dimension of the system is rather small.

The function \( \text{cfpd} \) calls one of two other functions when applying the criteria, depending on the value of \( \text{crit} \).

\[
\text{cprs}(b, yi, ui, L)
\]
computes the complementary partial row sums of a matrix \( L \) that is partitioned with block structure \( b \) and variable rearrangements \( yi \) and \( ui \). Within \( \text{cfpd} \), the function is applied to the RGA of \( G \) when \( \text{crit}=1 \). If \( \text{crit}=2 \), then \( \text{cfpd} \) calls
normv(b, y, u, G)

to calculate the norm of the matrix quantity $V(G)^3$, given the partition described by $b$, $y$, and $u$.

1.6.3 Filtering Partitions

You will typically wish to filter the partitions that pass the criteria with respect to certain common characteristics. For instance, you may wish to isolate all partitions that have a certain block structure or that group a subset of the measurements/manipulations together in a single block. The function

\[ \text{pfilterb}(P, b) \]

helps with the first task. You supply the set of partitions $P$ and the desired block structure $b$. The function returns those rows of $P$ that have the desired block structure.

The function \[ \text{pfilterv}(P, r, i, d, b) \]

filters the partitions in $P$ with respect to variable rearrangement. The argument $r$ specifies whether filtering will be done on row ($r=1$) or column ($r=2$) indices. (If filtering with respect to both rows and columns is desired, you may simply apply $\text{pfilterv}$ twice.) The indices of interest are specified in the vector $i$. These indices must appear in a single block. The scalar argument $b$ is optional, and specifies the size of the block that the indices must appear in. The function returns those partitions in $P$ that pass the conditions set by the arguments.

Again, while $\text{pfilterb}$ and $\text{pfilterv}$ were designed specifically for the partitioning problem, they are not the only functions available that can assist with filtering. A host of built-in MATLAB functions can be quite useful for this purpose. If you find that neither $\text{pfilterb}$ nor $\text{pfilterv}$ addresses a particular issue, you are encouraged to delve deeper into MATLAB.

\(^3\text{See [1] for an explanation of this quantity.}\)
1.7 A Typical Session

Control configuration design will typically require a number of CCD TOOLBOX sessions with a great deal of analysis between sessions. Nevertheless, this section will attempt to illustrate how the many available functions may be integrated for a successful approach to control configuration design.

Suppose you have entered matrices $A$, $B$, $C$, and $D$ that contain the state-space representation of your system. This system has 20 states, with 12 measurements and 10 manipulations available for closed-loop control. Since you have checked to see that the $A$ matrix is nonsingular, you can obtain the unscaled steady-state system matrix as follows:

$$ Gu = tfw(A,B,C,D,0); $$

Now this matrix should be scaled. You know the physical system fairly well, so you can specify the SSDOI for each of the first 6 measurements and 7 of the manipulations. You therefore build your deviation vectors:

$$ yd = [1.2 \ 500.0 \ 0.43 \ 2.7 \ 9.4 \ 0.01 \ Inf \ Inf \ Inf \ Inf \ Inf \ Inf]; $$
$$ ud = [Inf \ . \ 684.0 \ 56.0 \ 1.48 \ 9.75 \ 0.03 \ Inf \ 2.24 \ 1.04 \ Inf]; $$

and rely upon the function ussdoi to obtain estimates of the remaining 3 manipulation SSDOI's:

$$ us = ussdoi(Gu,yd,ud); $$

You then use the product of $Gu$ and $us$ to gauge the proper range for the remaining 6 measurement SSDOI's. After the full set of SSDOI's has been finalized and placed in the vectors $ys$ and $us$, you scale the system:

$$ G = scale(Gu,ys,us); $$

Since the matrix $G$ is $12 \times 10$, the system size should be reduced prior to an exhaustive search of the subsystems for selection. You are rather confident of your system scaling and have a nonsquare system, so you opt for Heuristic 1:

$$ [yr,ur] = rdcon(G,2,8,8); $$
$$ Gr = G(yr,ur); $$
You will now work with the reduced system $G_r$, always recalling that the indices of the reduced system must be interpreted through $y_r$ and $u_r$ if they are to be related back to the indices of the original system $G$. Now you wish to identify all possible subsystems for an $8 \times 8$ system:

$$S = \text{select}(8);$$

To evaluate these subsystems, apply the robust stability criteria:

$$S_1 = \text{rstab}(G_r,1,10,S);$$
$$S_2 = \text{rstab}(G_r,2,10,S1);$$

Remember that these criteria are applicable over a frequency range. Therefore you may choose to obtain $G$ at a frequency other than $w=0$ and apply the criteria again:

$$G_u = \text{tfw}(A,B,C,D,2);$$
$$G = \text{scale}(G_u,ys,us);$$
$$G_r = G(y_r,u_r);$$
$$S_3 = \text{rstab}(G_r,1,\text{db3*10},S2);$$
$$S_4 = \text{rstab}(G_r,2,\text{db3*10},S3);$$

Note that $\text{db3}$ is the $3\text{dB}$ parameter required by the viability criteria. It is a special value in the CCD TOOLBOX.

You must now devote significant effort to analyzing the results of the above steps. Some of this includes filtering the subsystems. For instance, you may be curious to know which of the subsystems that passed both criteria at both frequencies contain the fifth measurement and at least one of the second and fourth manipulations of the reduced system:

$$SF_1 = \text{sfilter}(S4,1,5,1,0);$$
$$SF_2 = \text{sfilter}(SF1,2,[2 4],1,1);$$

After further analysis, you identify several subsets of the measurements and manipulations that should be subjected to partitioning. Recall that the partitioning criteria apply at an upper frequency range. You will therefore need to compute the system matrices for the selected subsets at the appropriate frequencies. Suppose you choose to look at $w=7$ initially. One of the selected subsets is described by the first row of $SF_2$ and has 5 selected measurements and 5 selected manipulations:
\begin{verbatim}
Gu7 = tfw(A,B,C,D,7);
G7 = scale(Gu7,ys,us);
Gr7 = G7(yr,ur);
Gp = Gr7(SF2(1,1:5),SF2(1,9:13));
\end{verbatim}

Now obtain the set of all distinct partitions for a 5 \times 5 system:

\[ P = \text{partit}(5); \]

This set can be evaluated for \( Gp \) using the cross-feed performance degradation criteria:

\[ P1 = \text{cfpd}(Gp,1,0.15,P); \]
\[ P2 = \text{cfpd}(Gp,2,0.15,P1); \]

Filter the partitions that passed both criteria to see which ones have the \{2,2,1\} block structure:

\[ PF1 = \text{pfilterb}(P2,[2 2 1]); \]

At this point you are free to proceed, either by obtaining similar data at other frequencies or by continuing to analyze the partitions that result from the above. Your final conclusions will of course depend on the physics of the particular problem.
References


2 Reference

bdiag ................................................. 23
block ................................................. 24
blocksub ............................................. 25
cfpd ................................................... 26
combin ............................................... 27
concol ............................................... 28
conrow ............................................... 29
cprs .................................................. 30
db3 ................................................... 31
normv ............................................... 32
partit ............................................... 33
partsub ............................................. 34
pfilterb ............................................. 35
pfilterv ............................................. 36
rdcon ............................................... 37
rdrga ............................................... 38
rga ................................................. 39
rgelem .............................................. 40
rstab ............................................... 41
scale ............................................... 42
select .............................................. 43
selsub ............................................. 44
sfilter ............................................. 45
tfw .................................................. 45
ussdoi .............................................. 47
vrcol ............................................... 48
vrrow ............................................... 49
### System Representation

| tfw          | transfer function matrix |

### Scaling

| scale       | scale system matrix     |
| ussdoi      | manipulation SSDOI heuristic |

### System Size Reduction

| concol      | column reduction via heuristic 1 |
| conrow      | row reduction via heuristic 1 |
| rdcon       | heuristic 1 |
| rdrga       | heuristic 2 |
| rgelem      | heuristic 3 |

### Selection

| selsub      | subset of all distinct square subsystems |
| select      | all distinct square subsystems |
| rstab       | necessary conditions for robust stability |
| sfilter     | filter selected subsystems |
### Partitioning

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocksub</td>
<td>subset of all block structures</td>
</tr>
<tr>
<td>block</td>
<td>all block structures</td>
</tr>
<tr>
<td>vrcol</td>
<td>column rearrangements</td>
</tr>
<tr>
<td>vrrow</td>
<td>row rearrangements</td>
</tr>
<tr>
<td>partsub</td>
<td>subset of all distinct partitions</td>
</tr>
<tr>
<td>partit</td>
<td>all distinct partitions</td>
</tr>
<tr>
<td>cprs</td>
<td>complementary partial row sums</td>
</tr>
<tr>
<td>normv</td>
<td>norm of V</td>
</tr>
<tr>
<td>cfpd</td>
<td>necessary conditions for low cross-feed performance degradation</td>
</tr>
<tr>
<td>pfilterb</td>
<td>filter partitions by block structure</td>
</tr>
<tr>
<td>pfilterv</td>
<td>filter partitions by variable rearrangement</td>
</tr>
</tbody>
</table>

### Utilities

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bdiag</td>
<td>block diagonal of a matrix</td>
</tr>
<tr>
<td>combin</td>
<td>combinations</td>
</tr>
<tr>
<td>rga</td>
<td>relative gain array</td>
</tr>
</tbody>
</table>

### Special Values

<table>
<thead>
<tr>
<th>Term</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>db3</td>
<td>3.414</td>
</tr>
</tbody>
</table>
Purpose:
  Block-diagonal of a matrix.

Synopsis:
  bdiag(b, A)

Description:
  bdiag(b, A) is a square matrix with the dimension of A, containing
  the diagonal blocks of A and zeros everywhere else. The argument
  b is a vector that describes the block diagonal structure of A. The
  ith element of b is an integer that corresponds to the size of the ith
  diagonal block of A.

See Also:
  block, partition, cfpd
Purpose: All block structures.

Synopsis: block(n)

Description: block(n) is a matrix that specifies all distinct block structures that can be imposed on a size n system. Each row specifies a different block structure and has n elements. Each element is an integer corresponding to a diagonal block size. The block sizes will be consistently arranged in a nonincreasing order. Since the number of blocks may be less than n, the last few elements of a row may be zeros.

Algorithm: This function calls blocksub for each possible leading block size.

See Also: blocksub, partit
blocksub

Purpose:
Subset of all block structures.

Synopsis:
blocksub(n,b)

Description:
blocksub(n,b) is a matrix that specifies all distinct block structures that can be imposed on a size n system, where the leading block sizes correspond to the elements of b.

Note that b must conform to the ccd convention that block sizes are listed in a nonincreasing order. This means that b not only specifies the leading block sizes, but the largest block size. That is, the first element of b will be the largest block size for all of the block structures in blocksub(n,b).

Each row of blocksub(n,b) specifies a different block structure and has n elements. Each element is an integer corresponding to a diagonal block size. The block sizes will be consistently arranged in a nonincreasing order, and in each case the leading block sizes will correspond to the elements of b. Since the number of blocks may be less than n, the last few elements of a row may be zeros.

Algorithm:
This function calls itself recursively in order to specify only distinct block structures.

See Also:
block, partsub, partit
Purpose:
Necessary conditions for low cross-feed performance degradation.

Synopsis:
\[ \text{cfpd}(G, \text{crit}, dh) \]
\[ \text{cfpd}(G, \text{crit}, dh, P) \]

Description:
\text{cfpd}(G, \text{crit}, dh) \text{ applies necessary conditions for low cross-feed performance degradation to all possible partitions of the square system } G. \\
\text{cfpd}(G, \text{crit}, dh, P) \text{ applies necessary conditions for low cross-feed performance degradation to a set of partitions } P \text{ of the square system } G. \\
G \text{ is the transfer function matrix of a system, evaluated at a particular frequency. It is assumed to have dimension } n. \\
\text{crit} \text{ specifies which criterion should be applied. It can only be either of the following values:}

\begin{align*}
    1 & \quad \text{RGA criterion (scale-independent)} \\
    2 & \quad \text{V criterion (scale-dependent)}
\end{align*}

dh \text{ is the upper bound required by the criteria. It corresponds to the cross-feed performance degradation margin.}

P \text{ must conform to ccd convention in its description of the partitions. The first } 3n \text{ elements of each row must be of the form } [b \ yi \ ui], \text{ where } b, \ yi, \ ui \text{ are length } n \text{ row vectors describing the block structure, row variable rearrangements, and column variable rearrangements, respectively. Additional columns may be present, but will be ignored by the function.}

\text{cfpd} \text{ produces a matrix where each row contains a partition of } G \text{ which passes the specified criterion. Partitions are described according to ccd convention. The last element of each row is the quantity evaluated by the criterion for that partition.}

See Also:
\text{partit, cprs, normv}

26
Purpose:
Combinations.

Synopsis:
combin(x,n)
combin(x,n,xb)

Description:
combin(x,n) produces all combinations of n items chosen from the set x.
combin(x,n,xb) prepends to each of these combinations a fixed set of items, xb, which are common to all combinations.
combin returns a matrix with row dimension specified by the standard combinatorial formula, and column dimension equal to n plus length(xb). Each row contains a unique combination.

Algorithm:
This function calls itself recursively.
Purpose:
Column reduction via Heuristic 1.

Synopsis:
concol(G, yb, ub, yi, ui)

Description:
This function is called by rdcon to eliminate one column from a system matrix, based on the condition number of the resulting reduced system matrix.

G is the system matrix, evaluated at a particular frequency.
yb and ub are the indices of measurements and manipulations that must be included in the reduced system matrix and may not be eliminated by Heuristic 1. These vectors may be defined to be [].
yi and ui are the indices of measurements and manipulations that may be eliminated via application of Heuristic 1. Note that some of the measurement/manipulation indices may not be in either yi/ ui or yb/ ub if they have been already been eliminated through a previous application of concol or conrow.

The function produces a vector in the form of ui which has one less element than ui. The missing element corresponds to the column index of the eliminated manipulation.

See Also:
rdcon, conrow
conrow

Purpose:
Row reduction via Heuristic 1.

Synopsis:
conrow(G,yb,ub,yi,ui)

Description:
This function is called by rdcon to eliminate one row from a system matrix, based on the condition number of the resulting reduced system matrix.

G is the system matrix, evaluated at a particular frequency.

yb and ub are the indices of measurements and manipulations that must be included in the reduced system matrix and may not be eliminated by Heuristic 1. These vectors may be defined to be [ ].

yi and ui are the indices of measurements and manipulations that may be eliminated via application of Heuristic 1. Note that some of the measurement/manipulation indices may not be in either yi/ui or yb/ub if they have been already been eliminated through a previous application of concol or conrow.

The function produces a vector in the form of yi which has one less element than yi. The missing element corresponds to the row index of the eliminated measurement.

See Also:
rdcon, concol
Purpose:
Complementary partial row sums.

Synopsis:
cprs(b,yi,ui,L)

description:
cprs(b,yi,ui,L) returns a vector containing complementary partial row sums for a partitioned matrix L. The partition is specified by b, yi, and ui in accordance with ccd convention. The ordering of the row sums is consistent with the ordering in yi.

A complementary partial row sum is the sum of the elements on a row of L that lie outside the diagonal block that contains that row.

See Also:
cfPD
Purpose:
3dB parameter.

Synopsis:
\texttt{db3}

Description:
\[
db3 = \frac{2}{(2 - \sqrt{2})} \approx 3.414
\]
This value is important for various ccd criteria.

See Also:
\texttt{rstab, cfpd}
Purpose:
Norm of V.

Synopsis:
\texttt{normv(b,yi,ui,G)}

Description:
\texttt{normv(b,yi,ui,G)} computes the norm of V for a square partitioned matrix G. The partition is specified by \( b, yi, \) and \( ui \) in accordance with ccd convention.

\( V \) is the matrix quantity required by one of the necessary conditions for low cross-feed performance degradation.

See Also:
\texttt{cfpd}
partit

Purpose:
All distinct partitions.

Synopsis:
partit(n)

Description:
partit(n) returns all distinct partitions of a square system with dimension n.
partit conforms to ccd convention in its description of the partitions. The $3n$ elements of each row are of the form $[b \ y_i \ u_i]$, where $b$, $y_i$, $u_i$ are length $n$ row vectors describing the block structure, row variable rearrangements, and column variable rearrangements, respectively.

See Also:
partsub
partsub

Purpose:
Subset of all distinct partitions.

Synopsis:
partsub(b)

Description:
partsub(b) returns all distinct partitions of a square system that have the block structure b. partsub assumes that the dimension of the system is n, where n equals the sum of the elements of b.

partsub conforms to ccd convention in its description of the partitions. The 3n elements of each row are of the form [b y i u i], where b, y i, u i are length n row vectors describing the block structure, row variable rearrangements, and column variable rearrangements, respectively.

See Also:
vrrow, vrcol
**pfilterb**

**Purpose:**
Filter partitions by block structure.

**Synopsis:**

\[ pfilterb(P, b) \]

**Description:**

\[ pfilterb(P, b) \] filters a set of partitions with respect to the block structure.

*P* is a matrix that describes a set of partitions according to ccd convention. Specifically, each row must begin with the row vector \([b \ y_1 \ u_1]\). At most one additional column may be present in *P*, but will be ignored by the function.

*b* specifies the desired block structure. It must conform to ccd convention.

*pfilterb* returns those rows of *P* that contain partitions with block structure *b*. Each row describes a partition using ccd convention.

**See Also:**

* pfilterv, partit
pfilterv

Purpose:
Filter partitions by variable rearrangement.

Synopsis:
\texttt{pfilterv(P,rc,ind)}
\texttt{pfilterv(P,rc,ind,bl)}

Description:
pfilterv filters a set of partitions with respect to the variable rearrangement.

\( P \) is a matrix that describes a set of partitions according to ccd convention. Specifically, each row must begin with the row vector \([b\ yi\ ui]\). At most one additional column may be present in \( P \), but will be ignored by the function.

\( rc \) is a scalar that specifies whether the filtering will be done on row \((rc=1)\) or column \((rc=2)\) indices.

\( ind \) is a vector containing the filter indices. These are indices that are specified to appear together in a single block.

\( bl \) is an optional argument. When present, it specifies the size block that the filter indices must appear in.

pfilterv returns a matrix containing rows of \( P \) that pass the filter criteria. Each row represents a partition, described according to ccd convention.

See Also:
pfilterb, partit
Purpose:
System size reduction via Heuristic 1.

Synopsis:
\[ [yr,ur] = rdcon(G,m,n) \]
\[ [yr,ur] = rdcon(G,m,n,yb,ub) \]

Description:
\[ rdcon(G,m,n) \] uses Heuristic 1 to reduce the size of a system matrix \( G \). \( rdcon \) should only be employed if the system is scaled properly, since Heuristic 1 is quite dependent on the system scaling.

\( m \) and \( n \) specify the desired reduced row and column dimensions, respectively.

The function returns the indices of the measurements and manipulations retained in the reduced system. The length \( m \) row vector \( yr \) contains the retained measurement indices. The length \( n \) row vector \( ur \) contains the retained manipulation indices.

If certain of the measurements and manipulations must be retained in the reduced system, the user can specify the indices of these variables in \( yb \) and \( ub \). Then the vectors \( yr \) and \( ur \) returned by \( rdcon(G,m,n,yb,ub) \) will always include these indices.

See Also:
concol, conrow, rdrga, rgelem
Purpose:
System size reduction via Heuristic 2.

Synopsis:
\[ [yr,ur] = rdrga(G,n) \]
\[ [yr,ur] = rdrga(G,n,yb,ub) \]

Description:
\( rdrga(G,n) \) uses Heuristic 2 to reduce the size of a system matrix \( G \). \( rdrga \) can only be applied to square systems, and yields a square reduced system. However, Heuristic 2 employs the Relative Gain Array, so that the results of \( rdrga \) will be independent of scaling.

\( n \) specifies the desired reduced dimension.

The function returns the indices of the measurements and manipulations retained in the reduced system. The length \( n \) row vector \( yr \) contains the retained measurement indices. The length \( n \) row vector \( ur \) contains the retained manipulation indices.

If certain of the measurements and manipulations must be retained in the reduced system, the user can specify the indices of these variables in \( yb \) and \( ub \). Then the vectors \( yr \) and \( ur \) returned by \( rdrga(G,n,yb,ub) \) will always include these indices.

See Also:
rdcon, rgelem

rdrga

rdrga
rga

Purpose:
Relative gain array.

Synopsis:
rga(G)

Description:
rga(G) returns the relative gain array of the square matrix (real or complex) G.
Purpose:
System size reduction via Heuristic 3.

Synopsis:
\[ [yr,ur] = \text{rgelem}(G,n) \]
\[ [yr,ur] = \text{rgelem}(G,n,yb,ub) \]

Description:
\text{rgelem}(G,n) uses Heuristic 3 to reduce the size of a system matrix \( G \). \text{rgelem} can only be applied to square systems, and yields a square reduced system. However, Heuristic 3 employs the Relative Gain Array, so that the results of \text{rgelem} will be independent of scaling.

\( n \) specifies the desired reduced dimension.

The function returns the indices of the measurements and manipulations retained in the reduced system. The length \( n \) row vector \( yr \) contains the retained measurement indices. The length \( n \) row vector \( ur \) contains the retained manipulation indices.

If certain of the measurements and manipulations must be retained in the reduced system, the user can specify the indices of these variables in \( yb \) and \( ub \). Then the vectors \( yr \) and \( ur \) returned by \text{rgelem}(G,n,yb,ub) will always include these indices.

See Also:
\text{rdcon}, \text{rdrga}
Purpose:
Necessary conditions for robust stability.

Synopsis:
rstab(G,crit,bd)
rstab(G,crit,bd,S)

Description:
rstab(G,crit,bd) applies necessary conditions for robust stability to all square selection subsystems of the system G.
rstab(G,crit,dh,S) applies necessary conditions for robust stability to a set S of square selection subsystems of the system G.

G is the transfer function matrix of a system, evaluated at a particular frequency. It may be nonsquare, but the smallest dimension is assumed to be m.

crit specifies which criterion should be applied. It can only be either of the following values:

1 — RGA criterion (scale-independent)
2 — Condition number criterion (scale-dependent)

bd is the upper bound required by the criteria. It is a function of the specified relative-additive uncertainty margin.

S must conform to ccd convention in its description of the selection subsystems. Specifically, each row must begin with the row vector [yi ui]. The m elements of yi must contain the row indices of the selected measurements and additional zeros as necessary. The m elements of ui must contain the column indices of the selected manipulations and additional zeros as necessary. Additional columns may be present, but will be ignored by the function.

rstab produces a matrix where each row contains a square selected subsystem of G which passes the specified criterion. Selected subsystems are described according to ccd convention. The last element of each row is the quantity evaluated by the criterion for that selected subsystem.

See Also:
select, selsub
Purpose:
Scale system matrix.

Synopsis:
\texttt{scale(G,ys,us)}

Description:
\texttt{scale(G,ys,us)} scales a system matrix \texttt{G} using the vectors \texttt{ys} and \texttt{us}. \texttt{G} may be nonsquare, and is assumed to have been computed at a particular frequency.

\texttt{ys} and \texttt{us} specify the scale factors. They should contain the smallest significant deviations of interest (SSDOI's) for the measurements and manipulations, respectively. The SSDOI's should have the same units and be listed in the same order as the measurements and manipulations of \texttt{G}.

The function returns a scaled system matrix. The measurements and manipulations for this matrix will have units of SSDOI.

See Also:
\texttt{ussdoi}
select

Purpose:
All distinct square subsystems.

Synopsis:
  select(sz)
  select(sz,yb,ub)

Description:
  select(sz) selects all distinct square subsystems from a size sz system.
  select(sz,yb,ub) selects from a size sz system only those distinct square subsystems that include the set of row indices yb and the set of column indices ub.

sz specifies the dimensions of the system. If the system is nonsquare, then sz is a 2-element row vector [m,n], where m is assumed to be the row dimension and n is assumed to be the column dimension. If the system is square, then sz may be simply a scalar.

yb and ub are optional vectors which, if non-empty, contain row and column indices, respectively, that must appear in every selected subsystem. One may not be present without the other, so that occasionally one must be defined to be the empty vector, [ ].

select conforms to ccd convention in its description of the selected subsystems. Each row has the form [yi ui], where the min(sz) elements of yi specify selected row indices while the min(sz) elements of ui specify selected column indices. The selected square subsystems range in size from 2 to min(sz).

See Also:
  selsub
Purpose:
Subset of all distinct square subsystems.

Synopsis:
\texttt{selsub}(k,sz)
\texttt{selsub}(k,sz,yb,ub)

Description:
\texttt{selsub}(k,sz) selects all distinct square subsystems with dimension $k$ from a size $sz$ system.

\texttt{selsub}(k,sz,yb,ub) selects only distinct $k$-dimensional square subsystems from a size $sz$ system that include the set of row indices $yb$ and the set of column indices $ub$.

$sz$ specifies the dimensions of the system. If the system is nonsquare, then $sz$ is a 2-element row vector $[m,n]$, where $m$ is assumed to be the row dimension and $n$ is assumed to be the column dimension. If the system is square, then $sz$ may be simply a scalar.

$k$ can range in size from 2 to $\text{min}(sz)$.

$yb$ and $ub$ are optional vectors which, if non-empty, contain row and column indices, respectively, that must appear in every selected subsystem. One may not be present without the other, so that occasionally one must be defined to be the empty vector, $[]$.

\texttt{selsub} conforms to ccd convention in its description of the selected subsystems. Each row will have $2\text{min}(sz)$ elements in the form $[yi\ u_i]$. The first $k$ nonzero elements of $yi$ specify selected row indices while the first $k$ nonzero elements of $u_i$ specify selected column indices.

See Also:
\texttt{select}
Purpose:
Filter selected subsystems.

Synopsis:
$sfilter(S,rc,ind,num,fl)$

Description:
$sfilter(S,rc,ind,num,fl)$ filters a set of selected subsystems with respect to specified row or column indices.

$S$ is a matrix that describes a set of selected subsystems according to ccd convention. Specifically, each row must begin with the row vector $[y_i u_i]$. Assume that the minimum dimension of the original system is $m$. Then the $m$ elements of $y_i$ must contain the row indices of the selected measurements and additional zeros as necessary. The $m$ elements of $u_i$ must contain the column indices of the selected manipulations and additional zeros as necessary. At most one additional column may be present in $S$, but will be ignored by the function.

$rc$ is a scalar that specifies whether the filtering will be done on row ($rc=1$) or column ($rc=2$) indices.

$ind$ is a vector containing the filter indices. These are indices that the filter will look for (either to include or exclude, as specified by $num$ and $fl$) in each selected subsystem.

$num$ is a scalar that specifies how many of the filter indices should appear together within a selected subsystem. Note that specifying a value of zero can cause the filter to exclude subsystems that contain the specified filter indices (depending on the value of $fl$).

$fl$ is a scalar that specifies the filter logic. That is, whether subsystems should include at least $num$ of the filter indices ($fl=1$), exactly $num$ of the filter indices ($fl=0$), or at most $num$ of the filter indices ($fl=-1$).

$sfilter$ returns a matrix containing rows of $S$ that pass the filter criteria. Each row represents a selected subsystem, described according to ccd convention.

See Also:
select

45
tfw

Purpose:
Transfer function matrix.

Synopsis:
G = tfw(A,B,C,D,w)

Description:
This function evaluates the transfer function matrix of a system at a given frequency, w, from the state-space matrices of the system. It uses the commands:

s = j*w;
G = C/(s*eye(A)-A)*B+D;

To obtain the steady-state transfer function matrix, a value of w=0 may be entered. Note, however, that the matrix A must be nonsingular for this to be possible.
ussdoi

**Purpose:**
Manipulation SSDOI heuristic.

**Synopsis:**

\[
\text{us} = \text{ussdoi}(G,yd,ud)
\]

**Description:**
This function obtains estimates for unknown manipulation SSDOI's.

- \( G \) is a transfer function matrix of a system, evaluated at steady-state. It may be nonsquare.
- \( yd \) and \( ud \) are vectors containing the smallest significant deviations of interest (SSDOI's) for the measurements and manipulations. The deviations should have the same units and be listed in the same order as the measurements and manipulations of the system. Unknown SSDOI's should be assigned a value of \( \text{Inf} \). The user should take care to insure that the SSDOI's supplied by \( yd \) are meaningful, since they will be the basis for obtaining estimates for the unknown manipulation SSDOI's.
- \( \text{us} \) contains estimates of the manipulation SSDOI's for the system. It will only differ from the deviation vector \( ud \) in the elements where \( \text{Inf} \) was located. That is, the function will not alter any SSDOI's supplied in \( ud \). This function may not always be able to provide an estimate for every manipulation SSDOI. In such a case, the vector \( \text{us} \) will still include some elements with a value of \( \text{Inf} \).

**See Also:**

scale
vrcol

Purpose:
Column rearrangements.

Synopsis:
\texttt{vrcol(b)}

Description:
\texttt{vrcol(b)} performs all column rearrangements within a given block structure \texttt{b} that contribute to distinct partitions of a system.
\texttt{vrcol} assumes that the system is square with dimension \(n\), where \(n\) equals the sum of the elements of \texttt{b}.
\texttt{b} should conform to ccd convention for specifying the block structure of a partitioned system.
\texttt{vrcol} returns a matrix where each row describes a distinct ordering of the columns of a partitioned system within the block structure \texttt{b}. Each row will have \(n\) integer elements, corresponding to column indices.

Algorithm:
In order to identify only distinct partitions, the presence of repeated diagonal blocks in a block structure must be accounted for by either the column variable rearrangement function or the row variable rearrangement function. Note that this function accounts for the presence of repeated blocks. Therefore, the companion function \texttt{vrrow} does not account for repeated blocks.

See Also:
\texttt{combin, vrrow, partsub}
vrrow

Purpose:
Row rearrangements.

Synopsis:
vrrow(b)

Description:
vrrow(b) performs all distinct row rearrangements within a given block structure b.

vrrow assumes that the system is square with dimension n, where n equals the sum of the elements of b.

b should conform to ccd convention for specifying the block structure of a partitioned system.

vrrow returns a matrix where each row describes a distinct ordering of the rows of a partitioned system within the block structure b. Each row will have n integer elements, corresponding to row indices.

Algorithm:
In order to identify only distinct partitions, the presence of repeated diagonal blocks in a block structure must be accounted for by either the column variable rearrangement function or the row variable rearrangement function. Note that the companion function vrcol accounts for the presence of repeated blocks. Therefore, vrrow does not account for repeated blocks.

See Also:
combin, vrcol, partsu