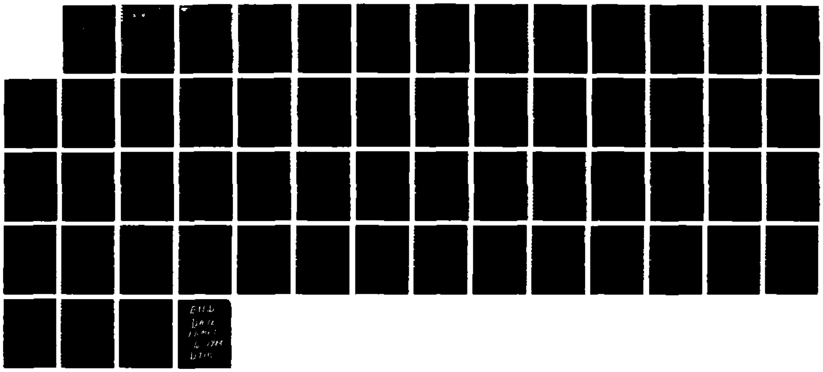
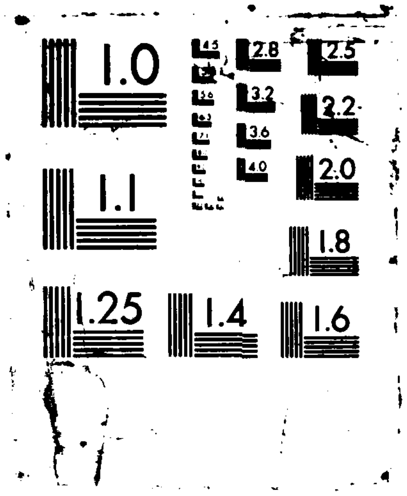
AD-A192 446 COMPUTER-AIDED-CONTROL ENGINEERING (CACE) PRIMITIVES FOR ROBUST AND ADAPT..(U) INTEGRATED SYSTEMS INC SANTA CLARA CA R L KOSUT ET AL. 12 OCT 87 ISI-108 1/1

UNCLASSIFIED AFOSR-TR-87-2024 F49620-86-C-0100 F/G 12/7 NL




1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

(2)

ATION PAGE DTIC FILE COPY

AD-A192 446

JAN 2 1 1988

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	Unclassified
4. PERFORMING ORGANIZATION REPORT NUMBER(S): ISI No. 108	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 87-2024
6a. NAME OF PERFORMING ORGANIZATION: Integrated Systems Inc. / 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: AFOSR/NM
6c. ADDRESS (City, State and ZIP Code): 2500 Mission College Boulevard, Santa Clara, California 95054-1215	7b. ADDRESS (City, State and ZIP Code): AFOSR/NM, Bldg 410, Bolling AFB DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: ~~DCASMA~~ AFOSR / 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: F49620-86-C-0100
8c. ADDRESS (City, State and ZIP Code): 1250 Bayhill Drive, San Bruno, California 94066	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102F	2304	A5	-

11. TITLE (Include Security Classification)
Computer-Aided-Control-Engineering: Primitives for Robust and Adaptive Control

12. PERSONAL AUTHOR(S)
Robert L. Kosut, M. Vidyasagar

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final Technical Report	FROM 8/86 TO 2/87	12 October 1987	54

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The objective of the past research was the development of some mathematical and computational tools that are appropriate to the next generation of CACE (Computer-Aided-Control-Engineering) environments. These CACE packages will be radically different than the present analysis packages in that they will truly be able to perform control system syntesis. In order for this ideal situation to come about, it is necessary first to solve some important problems in the mathematics of control systems as well as in computational techniques. Phase I of the research program addressed the feasibility of solving some of the problems. Phase II will extend the results of Phase I as well as developing primitives for performing robust and adaptive control design. In particular, we will address the following control problems: (i) synthesis of linear control systems, (ii) analysis of transfer function estimation, (iii) analysis of parameter adaptive control, and (iv) language architecture for control design.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED [✓] SAME AS RPT. ☐ DTIC USERS ☐	Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
~~Robert L. Kosut~~ Maj. [illegible]	~~408/980-1500~~ (202) 767-[illegible]	NM

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE. Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

integrated systems inc.

2500 Mission College Boulevard
Santa Clara, California 95054-1215
Tel: (408) 980-1500 • Telex 559631 • FAX: (408) 980-0400

AFOSR-TR- 87-2024

Phase I Final Report
SBIR Program
Under Contract No. F49620-86-C-0100

Computer-Aided-Control-Engineering (CACE): Primitives for Robust and Adaptive Control Systems

Prepared by:

Robert L. Kosut
Integrated Systems, Inc.
2500 Mission College Boulevard
Santa Clara, California 95054

M. Vidyasagar
University of Waterloo
Department of Electrical Engineering
Waterloo, Ontario

Prepared for:

AFOSR
Directorate of Information and Mathematical Sciences
Bolling Air Force Base
Washington, DC 20332

ISI REPORT NO. 108

12 October 1987

88 1 6 028

Table of Contents

List of Figures

Preface

Integrated Systems, Inc. (ISI) is pleased to submit this SBIR Phase I Final Report to the Directorate of Mathematical and Information Sciences of the Air Force Office of Scientific Research. The objectives of this research are to provide some mathematical and computational tools appropriate to the next generation of Computer-Aided-Control-Engineering (CACE) environments for robust and adaptive control. It is anticipated that these results will provide the building blocks for CACE packages which can truly perform control system synthesis.

The principal investigator for this effort is Dr. Robert L. Kosut, who is a Senior Scientist at ISI and a Consulting Professor in the Department of Electrical Engineering at Stanford University. Professor M. Vidyasagar of the Department of Electrical Engineering at the University of Waterloo was a consultant on the Phase I effort.

The report addresses both Phase I research results and follow-on activities for the Phase II.

Section 1
Problem Significance

Phase I research addressed the development of some mathematical and computational tools that are appropriate to the next generation of CACE (Computer Aided Control Engineering) environments. These future CACE packages will be radically different from the present packages in that they will truly be able to perform control systems synthesis, rather than just analysis.

In order for this ideal situation to come about, it is necessary first to solve some important problems in the mathematics of control systems, develop corresponding computational techniques, and then incorporate the results into software subroutines or building blocks, referred to here as *primitives*. These primitives, when linked together by a control system *language architecture* provide the basic CACE synthesis tools which are accessible by the user.

The Phase II objectives involve the development, where appropriate, of mathematical and computational tools for control system synthesis and analysis, the development of the corresponding primitives, and the generation of compatible language architectures. The research and development efforts will concentrate on robust and adaptive control of linear systems. As a by-product of our Phase II research we will also develop some *mock-up software* to test the computational tools and proposed language architecture.

1.1 Summary of Phase I Results: Relation to Phase II

The Phase I effort concentrated on some problems in robust and adaptive control of linear systems. Among the problems investigated during Phase I were the following: (i) new approaches to the simultaneous stabilization problem, (ii) uncertainty modeling, (iii) numerical techniques for constrained $\mathbf{H}_\infty$-optimization problems, and (iv) stability of reduced order adaptive control.

The Phase I objectives are an overture to our Phase II goals which, by virtue of the CACE interface, will serve the traditional application areas such as aircraft stabilization and maneuver guidance, or the more recent application areas of robot manipulator feedback control. For example:

(i) the simultaneous stabilization problem will enable the same set of controller gains to work over a wide range of manipulator geometric configurations or aircraft flight regimes;

(ii) uncertainty modeling will be an important part of robust control strategies for prelinearized models where it is undesirable to include parasitic nonlinearities in the nonlinear dynamic model, other than via a conic sector bound;

(iii) the $\mathbf{H}_\infty$-optimization problem will ensure that linearized controllers will have guaranteed performance robustness to uncertainty, yet will not be overly conservative so as to limit potential performance; and

(iv) adaptive control will be extremely important for coping with unwanted dominant nonlinear effects as well as on-line tuning of controller parameters.

As a result of our efforts during Phase I, we are now in a position to begin the development of some software primitives which correspond to the above control problem solutions. Development of these primitives are a part of the Phase II objectives.

1.2 Status of Computer-Aided-Control-Engineering (CACE)

This subsection summarizes the status of Computer-Aided-Control-Engineering (CACE) technology.

1.2.1 History

The foundations of CACE were laid down in the 60's and 70's with the creation of basic linear algebra subroutine packages such as LINPACK, EISPACK, IMSL and others. These packages were (and in many instances still are) used by engineers from many disciplines to perform the numerical analyses required in their work.

Control engineers and numerical analysts eventually added control specific functionalities to these packages to create control subroutine packages for linear system analysis. These usually included facilities for classical analysis (Bode, Nyquist, and root-locus diagrams) and occasionally some optimal control primitives (such as Riccati or Lyapunov equation solvers).

Control subroutine packages have generally been developed in isolated communities, such as corporations or universities. They tend to handle a fairly limited range of problems and generally embody a particular design philosophy. Since they are not commercially supported, confidence in them is low outside the authoring organization. These factors have kept any one control subroutine package from achieving widespread acceptance.

CACE started out in a new direction with the introduction of MATLAB in the late seventies. Rather than being just a collection of subroutines, MATLAB included an interactive interface which gave the user immediate and easy access to numerous linear algebra routines. Initially intended to be used for basic numerical analysis in mathematics and engineering in general, MATLAB was soon adapted for control engineering by ISI, forming the basis for MATRIX_X, which appeared in 1982. Systems Control Technology later followed about one year later with the introduction of CTRL-C, which also uses MATLAB as the starting point.

There has always been a considerable amount of interest in CACE at universities. Packages such as DELIGHT.MIMO and CLADP all appeared at about the same time as MATRIX_X. DELIGHT.MIMO was interesting because of its specifications-constrained optimization approach in control design and CLADP contained some very useful modern robust multivariable design tools along with classical tools. The history of CACE is shown in Figure 1-1.

1.2.2 CACE at ISI

ISI has taken an integrated approach to CACE by considering the entire control engineering cycle and creating a family of CACE products. We have broken the control cycle into three phases: design, test, and implementation, which are shown in Figure 1-2. Our CACE products, the MATRIX_X design package, the MAX_100 control implementation processor, and the RT_BUILD Ada source code generator respectively address those three phases. Each phase will now be discussed individually.

Design:

The design engineer's ultimate objective is to create a compensator which controls the plant so as to meet all prescribed specifications. The designer usually does not have access to the real plant and would rather not concern himself too much with the details of implementation (although this is now much less of a conern, as we will discuss later). The design cycle is shown in Figure 1-3.

The designer works in the realm of models. He models the plant to be controlled, models the compensator design, and verifies that the compensator model controls the plant model so as to meet the specifications. If he is fortunate, he will have access to test data from the real plant or an experiment which will allow him to refine his mathematical models in a process known as system identification. System identification requires sophisticated numerical algorithms and comprehensive signal processing capabilities.

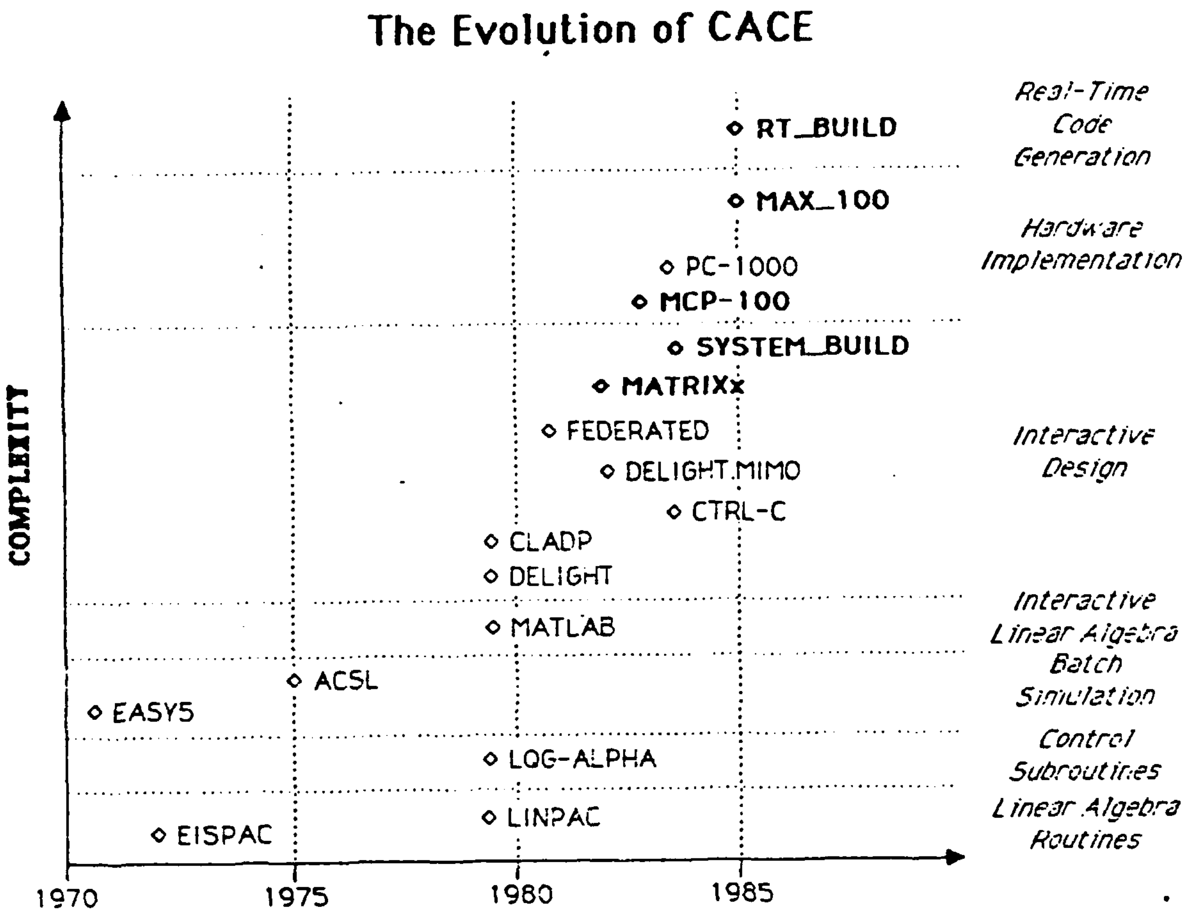


Figure 1-1. History of Computer-Aided Control Engineering

The design of the compensator involves the operation of specific algorithms on the models. Analysis and simulation are required during the design phase to help select the design path and after design to verify performance. Throughout the process, the designer

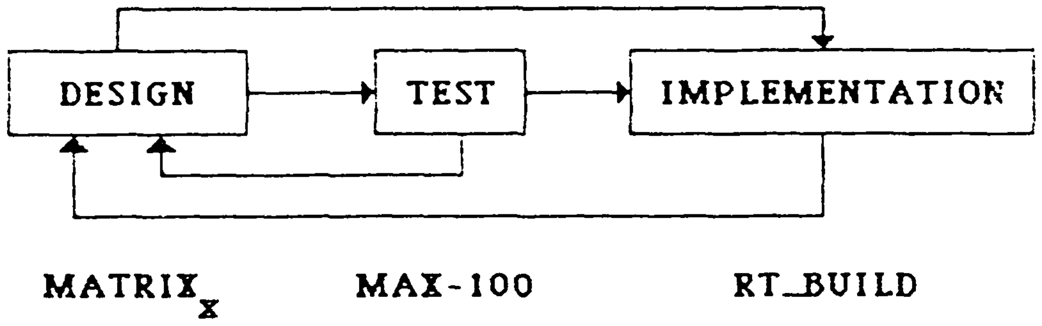


Figure 1-2. The Control Engineering Cycle and ISI Products

requires access to high quality graphics, both for improving his own perceptions of the problem and for communicating with others.

MATRIX$_X$ was designed to enhance the creativity and productivity of the designer by giving him comprehensive and easy to use tools for modeling, system identification, design, and analysis. MATRIX$_X$ consists of an interactive command and menu driven user interface which gives the user access to a large set of linear algebra commands, sophisticated engineering graphics, modern and classical design tools, deterministic and stochastic system performance analysis commands, system identification and signal processing features, and a unique interactive graphic non-linear modeling and simulation facility known as SYSTEM_BUILD.

Test:

A major part of the control engineering effort is dedicated to testing. Control system testing has traditionally been a tedious and expensive process. One of the main

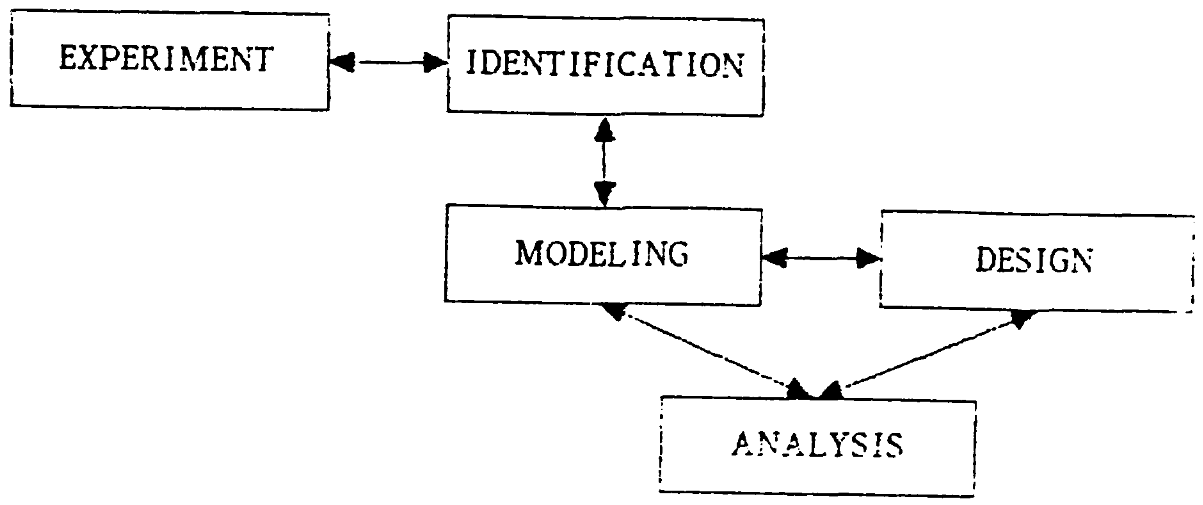


Figure 1-3. Steps in Control Design

purposes of testing is the feedback of information to the designer for use in compensator refinement. Delays and communications problems between design and test have often hindered this process, Shah and Houtchens(1985).

ISI developed the MAX_100 control implementation processor to bring design and test to within a few keystrokes of each other. The MAX_100 directly implements MATRIX$_X$ compensator designs in real-time, interfacing to experiments through analog-to-digital (A/D) and digital-to-analog (D/A) interfaces. The MAX_100 is unique in its ability to implement non-linear multi-rate compensator designs and in its user-transparent connection to the MATRIX$_X$ design package.

Implementation:

The final phase of the control engineering cycle is the implementation phase. In modern digital control *implementations*, the compensator design is transformed into a set of computer instructions. These instructions have traditionally been written in machine language, but high level languages such as FORTRAN, PASCAL, C, JOVIAL, HAL and Ada are now in widespread use. The implementation phase is extremely expensive, time consuming, and error prone. Since the designer does not usually possess the skills required for real-time code generation, the job is usually left to others who are experts in real-time programming.

ISI's RT_BUILD Ada source code generator is intended to bridge the gap between design and implementation by generating Ada source code directly from the MATRIX$_X$ design database, Lehman(1985). The resulting code includes not only the numerical operations prescribed by the design but also the overhead functions required to manage the real-time aspects of implementation, such as interrupt handling, multi-tasking, and fault-detection. RT_BUILD also includes utilities for verifying the real-time code through simulation and analysis.

1.2.3 CACE at University of Waterloo

The University of Waterloo has a long history of folowing up research adavances and carrying them through to the stage of production software. Early examples of this include the WATFOR, WATFIV compilers for FORTRAN, but there are several recent instances as well. More pertinent to the topic of this proposal is the software package SF-PACK, which was developed by M. Vidyasagar and his former doctoral student K.D. Minto. Like many other CACE tools, SFPACK is based on MATLAB, but with some important differences. It is particularly well-suited for carrying out designs based on the stable factorization approach; in addition, since it allows the user to define new functions, virtually all existing methodologies for linear multivariable control can also be programmed in SF-PACK. One of the novel features of SFPACK is that, in addition to working with matrices, it also permits the use of an additional data type known as a *packed matrix*. Specifically, if a linear multivariable system is described by a vector differential equation of the form

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

SFPACK permits one to represent this system as a packed matrix

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

Common physical operations such as series and parallel interconnections, system inversion, etc., can be carried out on the packed matrices, and SFPACK automatically does the necessary bookkeeping on the various submatrices of the packed matrix representation of the resulting system. In this way, SFPACK complements MATRIX_X by making a few operations somewhat easier.

1.2.4 CACE Elsewhere

Although numerous CACE packages exist, few (if any) have achieved the widespread acceptance of MATRIX_X and none can compare with it on the basis of capability, reliability, user friendliness, and availability on a wide range of computers. The basic difference between MATRIX_X and other packages is in the level of integration of its capabilities. Packages such as CTRL-C/ACSL, the Federated System, and Design Master have been assembled piecemeal from various subroutine and design packages. This has resulted in packages with multiple user interfaces, poor reliability, and incomplete capabilities.

Different styles of software user interfaces are required and desireable for distinct disciplines like structures, aerodynamics, and propulsion, since each can be optimized for different users. Although good data communication is desirable between those disciplines, a common syntax to generate structures, aerodynamic, and control data is unreasonable to expect. Control design, modeling, simulation, and identification, however, are tasks which are often performed by a single control engineer or within a small engineering group, and for these tasks only a single interface is necessary.

More recently, PC based CACE packages have begun to proliferate. Unfortunately, these are being developed exclusively for PC's, with no communications capability or growth paths to mainframe resources. The buyer is thus totally constrained to the PC. Althiugh PC's are adequate for many kinds of analyses, they are not sufficient for sophisticated designs or simulations, e.g., aircraft control systems. ISI's $\text{MATRIX}_X/\text{PC}^{\text{TM}}$package does not suffer the same drawbacks, since it is part of a fully integrated product line, with complete data compatibility and communications, as well as full functionality of MATRIX_X ,Pine et al.(1985).

1.3 Next Generation of CACE

At present, despite the recent advances and growing engineering community acceptance just described, the area of computer-aided design for control systems is still in an embryonic stage. All the design packages that are currently marketed, including MATRIX_X, are in reality *analysis* packages, in the sense that the user provides the controller to be used, and the package (whichever one it is) gives enough information in the form of graphs, charts, or numbers to enable the engineer to *evaluate* the design. If the design is not satisfactory, then the onus is on the engineer to provide the next iteration of the controller; the packages themselves provide no assistance in this regard.

Moreover, the bulk of these packages are set up to analyze linear finite dimensional systems. Since almost every dynamical system is nonlinear, if not infinite dimensional as well, it follows that *except for simulation capability, there are no nonlinear analysis packages*. Without such a package it is not possible to conceive of a nonlinear synthesis package. For example, if either a gain scheduled or adaptive control is required because of system performance demands, then the engineer has almost no recourse except time consuming simulations.

We envision that the next generation of CACE packages will be radically different in that they will be *synthesis* rather than analysis packages. Thus, in our view of the future, the engineer will commence the design with uncertain and/or incomplete information about the system to be controlled, as well as a set of design objectives and constraints. Once this information is fed into the CACE program, it will in turn generate a controller that meets the performance requirements while respecting the constraints, or else inform the engineer that the constraints cannot all be satisfied, and suggest some trade-offs. As the engineer thinks of more constraints and/or requirements, these are entered into the software and are accounted for as they are entered. Thus, the CACE process is still interactive, but the level of interaction with the computer is much higher than it is at present.

Such a utopian vision of CACE is possible thanks to dramatic progress in linear and nonlinear control methodologies. As a result, it is indeed possible to develop in a few years, CACE programs that are truly synthesis rather than analysis tools.

1.4 Recent Theoretical Developments

Specific theoretical developments that have advanced our ability to do synthesis have occurred in linear multivariable control, system identification, and adaptive control.

In linear multivariable control, one can mention the *stable factorization* approach [see, e.g., Desoer et al. (1980), Zames (1981), Vidyasagar (1985), Doyle (1985)], which has made it possible to formulate and solve problems such as:

(a) *Stabilization* - Given a plant, final *all* controllers that stabilize it.

(b) *Simultaneous stabilization* - Given several plant models, each representing the plant at different operating points or in different modes, find a *common* controller that stabilizes each plant model.

(c) *Robustness* - Given a nominal plant model, together with an uncertainty profile, find a stabilizing controller that is optimally robust, i.e., it preserves system stability under the widest possible extent of uncertainty.

In system identification a better understanding is now available regarding: (i) the estimation of the plant transfer function from input-output data [Ljung (1985, 1986), Wahlberg and Ljung (1986)], (ii) designing the identification experiment for intended use in control design [Gevers and Ljung (1986), Kosut (1986)], and (iii) the analysis of recursive algorithms for computing parameter estimates [Ljung and Sodestrom (1983)]. For example, it is now possible to solve problems such as:

(a) *Parameter convergence* - Given a recursive algorithm for estimating unknown plant parameters, find the asymptotic trajectories of the parameter estimates.

(b) *Estimation accuracy* - Given a parametric or non-parametric identification algorithm, find the bias and variance of the resulting estimate of the plant transfer function.

(c) *Experiment design* - Given a choice of input spectrum, feedback compensation, and data filtering, optimize the experiment so that performance degradation is minimized when the transfer function estimate is used in a control design.

In adaptive control, it is now possible to assess both stability and robustness properties in certain instances [Kosut and Johnson (1984), Kosut and Bitmead (1986), Anderson et al. (1986), Kosut et al. (1986)]. In particular, one can solve the following problems:

(a) *Slow adaptation* - Given a slowly adjusting parameter adaptive control, find the asymptotic parameter trajectories, assess its stability, and determine its region of attraction and rate of convergence.

(b) *Local stability* - Given a rule for parameter adjustment, one that is not necessarily slowly varying, determine the local stability and convergence properties of the parameter and state trajectories.

Because of these recent adavances in robust and adaptive control, we are now in a position to take the next step in the development of CACE packages for control system synthesis. These are descibed in the next section.

Section 2

Phase II Technical Objectives

The objective of Phase II is to solve some mathematical and computational problems that are appropriate to the development of what we refer to as *primitives* for control design, analysis, and synthesis problems. The feasibility of developing aspects of some of these primitives was investigated in Phase I as described in Section 1. In Phase II, we will expand the scope and develop primitives and mock-up software to perform the following functions:

(1) *Synthesis of linear control systems* - The user inputs to this primitive would be a plant model (or a set of plant models), and uncertainty profile, constraints, and objectives. The primitive would return as output a linear controller, if one exists, as well as system sensitivity to the constraints.

(2) *Analysis of transfer function estimation* - The user inputs would be an algorithm (parametric or non-parametric) for estimating the plant transfer function and a representative *true* plant system. The primitive would return transfer function estimation accuracy as a function of data record length and exogeneous input spectrum.

(3) *Analysis of parameter adaptive control* - The user inputs would be an adaptive algorithm for adjusting controller parameters and a representative *true* plant system. The primitive would return the asymptotic parameter and system state trajectories, measures of stability and convergence, an estimate of the region of attraction and rate of convergence.

(4) *Language architecture for control design* - The user will be able to address the primitives in a consistent formal language which is appropriate to the design of control systems.

The above functions, or primitives, perform tasks at a fairly high level and obviously involve other lower level primitives. For example, the linear control primitive in item (1) requires an optimization primitive once the user inputs have been translated into the appropriate form. The transfer function estimation primitive in item (2) requires model building primitives, perhaps a menu driven algorithm selector, and some means of selecting an exogeneous input spectrum. The adaptive control primitive in item (3) will certainly involve a primitive which characterizes the solution of ordinary differential equations, as well as routines for linearization, generating Lyapunov functions, etc. In addition, because of the several levels of translation involved in passing from one primitive to another, it is obvious that the Phase II efforts will, to some extent, be involved with the development of *language architectures* for control synthesis and analysis, as stated in item (4). In order to test the primitives and proposed language architecture, we will also develop some *mock-up software*. The Phase III effot will involve the further development of these primitives into commercial grade software.

Section 3

Phase II Work Plan

The Phase II Work Plan consists of four major tasks, each corresponding to the objectives listed in Section 2. These are now described in sufficient detail so as to highlight our approach.

3.1 TASK 1: Synthesis of Linear Control Systems

The aim of this task is to address several problems that arise in connection with the synthesis of controllers for linear multivariable systems. To bring some order into the description of this task, we group the various problems into three categories, namely: (i) design for a single operating point, (ii) design for multiple operating points, and (iii) design for time-varying operating points. To some extent, this partitioning is arbitrary, and the various subproblems described below can in fact be regrouped in other ways.

Task 1.1: Design for a Single Operating Point

A logical starting point in any controller synthesis problem is to design a controller for a given single plant, in such a way that several requirements are satisfied. Obviously the most basic requirement is that the controller must stabilize the plant, or somewhat more generally, place all closed-loop poles in some prespecified region within the open left half-plane. But mere stabilization is usually not enough. Typical additional requirements include, bur are not restricted to: robustness against unmodeled sector-bounded perturbations, minimum and maximum bandwidth for the controlled system, specifications of the rise time and settling time for the step response, maximum instantaneous amplitudes for each of the plant inputs, and so on. Problems of thie type are now manageable, in large part due to the so-called Youla parametrization of all controllers that stabilize a given plant. This result is briefly summarized here, both complete details can be found, among other places, in the recent monograph by Vidyasagar (1985). The particular statement below is for discrete-time systems, but there is complete analogy between continuous-time and discrete-time systems. This result can be stated as follows: Let $P(z)$ denote the transfer matrix of the given plant, and factor it in the form

$$P(z) = N(z)[D(z)]^{-1} = [\tilde{D}(z)]^{-1}\tilde{N}(z) , \tag{1.1}$$

where N, D, $\tilde{N}$, $\tilde{D}$ are stable rational matrices; further, find stable rational matrices X, Y, $\tilde{X}$, $\tilde{Y}$ such that the so-called Aryabhatta identity

$$\begin{bmatrix} Y & X \\ -\tilde{N} & \tilde{D} \end{bmatrix} \begin{bmatrix} D & -\tilde{X} \\ N & \tilde{Y} \end{bmatrix} = 1 \tag{1.2}$$

is satisfied. Then the set of all controllers that stabilize the given plant P is given by

$$S(P) = \{(Y - R\tilde{N})^{-1}(X + R\tilde{D})\} - = \{(\tilde{X} + DR)(\tilde{Y} - NR)^{-1}\} , \tag{1.3}$$

as the matrix R varies over the set of all stable rational matrices. The matrix R is known as the Youla parameter, and it encompasses all of the freedom that one has in designing the controller. Thus the problem of designing a suitable controller is equivalent to that of finding a suitable Youla parameter R.

One of the striking advantages of the Youla parametrization is that all transfer matrices in the controller system are in fact affine functions of R. As a consequence, the problem of designing a controller subject to the various requirements above can in fact be reduced to a very general problem formulation, which we refer to as the *canonical problem*. This canonical problem can be stated as follows: Given various stable matrices F_i, G_i, H_i of compatible dimensions, together with stable matrices A, B, C and constants α_i, find a stable matrix R that minimizes the cost functional

$$J = \|A - BRC\| \tag{1.4}$$

subject to the constraints

$$\|F_i - G_i R H_i\|_i \leq \alpha_i \ , \ i = 1, \ldots, k \ . \tag{1.5}$$

In the above, the norms $\|\cdot\|$ and $\|\cdot\|_i$ can be the $\mathbf{H}_2$ norm, the $\mathbf{H}_\infty$ norm, of the ℓ_1 norm, depending on the nature of the performance requirement. Once an optimal R is found that solves the above canonical problem, it can be substituted into the Youla parametrization to determine the corresponding controller.

It is at once noticed that the above canonical problem has a very nice structure, namely: it is a convex programming problem. In other words, both the objective function and all constraints are convex functions of R, which is the variable of optimization. This has several advantages. For instance, every local minimum of the problem is also a global minimum. Moreover, convex programming is a well-studied subject, and several powerful methodologies are available for solving such problems. Thus, once the various performance requirements are translated into corresponding constraints on the Youla parameter R, finding a suitable controller is, in principle at least, straight-forward. Once the canonical problem is solved and a suitable controller is found, typically the control engineer then changes those of the performance requirements that are soft constraints, and solves the new problem. Thus the underlying philosophy is to make the individual problem formulation as complete as possible and to let the computer do all the hard work; in this way the engineer enters the design loop at a very high level. Now, if such design is to be done on-line, it is clear that the time required for the solution of the canonical problem is of paramount importance, since it has to be solved afresh each time the performance requirements are modified. In Phase I of the project, we have established the feasibility of posing the controller design problem as an optimization problem, and have suggested some methods for solving the resulting optimization problem. Following up this work, in Phase II of the project we would like to study *efficient* ways of solving the canonical problem.

The major issue that needs to be confronted in solving the canonical optimization problem is that the parameter space, i.e., the set of stable rational matrices, is infinite-dimensional. While there are methods for solving infinite-dimensional programming problems, in practice it is highly desirable to approximate the problem by a finite-dimensional one. In Phase I we suggested the following procedure for doing this. We began by observing that the set of polynomials $\{z^i\}$, $i \geq 0$ is a basis for the set of stable rational functions. Thus every stable rational matrix can be expressed as an infinite sum

$$R(z) = \sum_{i=0}^{\infty} R_i z^i \ , \tag{1.6}$$

for an appropriate sequence of constant matrices $\{R_i\}$. Now, in order to approximate the canonical problem by a finite-dimensional one, we merely truncate the above sum, as

$$R(z) = \sum_{i=0}^{N} R_i z^i \ , \tag{1.7}$$

where N is a pre-selected large number. Now the matrices $R_0, \ldots, R_N$ are the variables of optimization, and the objective function and constraints are still convex functions of these variables. Thus we can continue to use convex programming to find an optimal set of matrices $R_0, \ldots, R_N$. Of course, the resulting polynomial matrix

$$R_{opt}^{(N)}(z) = \sum_{i=0}^{N} R_i z^i \tag{1.8}$$

is only suboptimal for the original canonical problem, but by choosing the integer N to be sufficiently large, we can get as close as we wish to the true optimal solution to the original problem. Finally, by substituting the Youla parameter $R_{opt}^{(N)}$ into the expression for the controller, we can obtain a controller which is suboptimal for the problem at hand; moreover, as $N \to \infty$, this controller approaches an optimal one.

The above procedure is quite appealing, but has two drawbacks. First, the size of the convex programming problem resulting from the finite-dimensional approximation can become quite large very quickly. To illustrate this point, note that if the plant has dimensions $n \times m$ (so that the controller C and the Youla parameter R have dimensions $m \times n$), then the number of variables of optimization is nmN. Thus, if the plant has dimension 5×5 and one uses a 20-th order polynomial approximation to R, then the number of variables is 500! The second drawback is that the order of the controllers that result form such an approach is generically equal to $N + k$, where k is the McMillan degree (i.e., the state-space order) of the plant. Thus, if one uses a reasonably large value for N (which is necessary in order to achieve an accurate polynomial approximation to a rational function), then the consequence is that even quite low-order plants will require high-order controllers. The principal aim of this subtask is to address these two drawbacks.

In order to counter high dimensionality in the convex programming problem, we propose to try out alternative means of converting the infinite-dimensional canonical problem into a finite-dimensional one. A logical starting point is to replace the Taylor series expansion

$$R(z) = \sum_{i=0}^{\infty} R_i z^i \tag{1.9}$$

by a series of the form

$$R(z) = \sum_{i=0}^{\infty} R_i (z + \lambda)^{-1} \ , \tag{1.10}$$

where λ is a real number with $|\lambda| > 1$. In effect, the Taylor series has the property that all partial sums of the series have all of their poles at infinity, whereas the alternate form suggested above has the property that all partial sums have all of the poles at $-\lambda$. By appropriate choice of λ, it should be possible to achieve more rapid convergence of the series than would be the case with a Taylor series. This alternative formulation has the advantage that the finite-dimensional approximation to the original canonical problem would still be a convex programming problem, and thus easy to solve.

Another alternative is to expand $R(z)$ in a continued fraction expansion of the form

$$R(z) = R_p(z) + \cfrac{R_1}{z + \cfrac{R_2}{z + \cfrac{R_3}{z + \cdots}}} \ . \tag{1.11}$$

It is a fold theorem that continued fraction expansions tend to converge faster than Taylor series. There are two difficulties with using a continued fraction expansion, however.

First, it is not clear what restrictions on the coefficients R_i ensure that $R(z)$ is a stable rational function. Second, if the expansion is truncated after a finite number of terms and the resulting coefficients are used as the variables of optimization, then the resulting optimization would no longer be a convex programming problem, but rather a nonlinear programming problem which is more difficult to solve.

To obtain controllers of reasonably small McMillan degree, we propose to proceed as follows. Once a suboptimal choice is made for the Youla parameter R, it is possible to form the composite matrix

$$A_c = [\tilde{D}_c \;\; NTC] = [Y - R\tilde{N} \;\; x + R\tilde{D}] \;. \tag{1.12}$$

The matrix A_c is stable, even if the controller itself is not. Thus it is possible to approximate the matrix A_c by another stable matrix M of lower McMillan degree, using for example the order reduction due to (Glover 1984). Suppose the resulting matrix M, which is stable and has the same dimensions as A_c, is partitioned commensurately with (1.12), as

$$M = [F \;\; G] \;. \tag{1.13}$$

Then the system $\bar{C} = F^{-1}G$ is a approximation to the original controller $C = \tilde{D}_c^{-1}\tilde{N}_c$, and moreover it has the same McMillan degree as the matrix M. Finally, it is possible to obtain an estimate of the graph metric distance between the systems C and $\bar{C}$. Complete details may be found in (Vidyasagar et al. 1987).

Task 1.2: Design for Multiple Operating Points

Next we examine the problem of designing a single controller for several operating points. Such a problem can arise in several ways. Two common examples are (i) controller design for a nonlinear system at distinct operating points or distinct modes of operation, and (ii) controller design for a linear system subject to failures in sensors and/or actuators. A typical application is the design of a controller for an aircraft which is effective over a range of flight conditions such as altitude and Mach number. The difference between the problem studied here and that studied in the first subtask is that, in the latter, one is given a single plant together with associated performance requirements, whereas in the former, one is given a collection of plants and a set of performance requirements for each plant. In contrast with the problem of designing for a single operating point, controller design for multiple operating points is relatively undeveloped, and much needs to be done.

Let $P_1, \ldots, P_r$ denote the Models of the plant at the various operating points. Note that the various plants need not have the same McMillan degree, i.e., the dimension of the state space model can change as the operating point changes. However, it is assumed that the dimensions of the matrix $P_i(z)$ is the same for all i; the loss of sensors (actuators) can be accommodated by setting corresponding rows (columns) equal to zero. The problem now is to choose a common controller C such that, for each i, the controller C stabilized the plant P_i, and in addition, satisfies the usual sorts of performance requirements such as robustness, etc.

In the case where only a single plant is to be controller, merely stabilizing the plant is not difficult problem. The problem of choosing a common controller C to stabilize each of a given set of plants $P_1, \ldots, P_r$ is known as the simultaneous stabilization problem. Depending on the nature of the plants, such a controller may or may not exist. The following results (Vidyasagar and Viswanadham 1982) gives a necessary and sufficient condition of the existence of a common stabilizing controller.

Theorem 3.1 Given the plants $P_1, \ldots, P_r$, factorize each plant P_i in the form

$$P_i = N_i D_i^{-1} = \tilde{D}_i^{-1} \tilde{N}_i \ , \tag{1.14}$$

and choose matrices X_i, Y_i, $\tilde{X}_i$, $\tilde{Y}_i$ such that

$$\begin{bmatrix} Y_i & X_i \\ = \tilde{N}_i & \tilde{D}_i \end{bmatrix} \begin{bmatrix} D_i & -\tilde{X}_i \\ N_i & \tilde{Y}_i \end{bmatrix} = I \ . \tag{1.15}$$

Now define

$$\begin{bmatrix} A_i \\ B_i \end{bmatrix} = \begin{bmatrix} Y_1 & X_1 \\ -\tilde{N}_1 & \tilde{D}_1 \end{bmatrix} \begin{bmatrix} D_i \\ N_i \end{bmatrix} \ , \ i = 2, \ldots, r \ , \tag{1.16}$$

and define the auxiliary systems

$$Q_i = B_i A_i^{-1} \ , \ i = 2, \ldots, r \ . \tag{1.17}$$

Then the plants $P_1, \ldots, P_r$ can be simultaneously stabilized if and only if there exists a stable matrix M such that M stabilizes each auxiliary plant Q_i, $i = 2, \ldots, r$.

In Phase I of this project, we examined a generalization of the simultaneous stabilization problem whereby one was given not only a set of plants $P_1, \ldots, P_r$, but also associated domains of stability $\mathbf{D}_1, \ldots, \mathbf{D}_r$; the objective is to find, if possible, a common stabilizing controller C which placed the closed-loop poles of the i-th system in the region $\mathbf{D}_i$. We derived the following results for this problem, which generalized Theorem 3.1.

Theorem 3.2 Suppose $\mathbf{D}_1$ is a subset of $\mathbf{D}_i$ for $i = 2, \ldots, r$. Let all symbols be as in Theorem 3.1, except that all poles of the eight matrices in (1.15) are in the region $\mathbf{D}_i$. Then there exists a controller that places all closed-loop poles of the i-th system in the region $\mathbf{D}_i$ if and only if there exists a matrix M whose poles are all in the region $\mathbf{D}_1$ such that all zeros of $A_i = MB_i$ are in the region $\mathbf{D}_i$.

In addition to the above, there is another set of results that is worthy of mention in this context.

Theorem 3.3 Suppose the plants $P_1, \ldots, P_r$ all have dimensions $n \times m$. Then simultaneous pole assignment is generic if $r \leq \max\{n, m\}$.

Comparing the known results for simultaneous design with those for design at a single operating point, we can notice an important difference at once. In the case of a single plant, we have available the Youla parametrization, which gives an expression for all controller that stabilize the given plant, and which serves as the starting point for the development of a more elaborate design procedure, culminating in the canonical problem. In contrast, there is no result available to date which gives an expression for all simultaneously stabilizing controllers. The only exception to this statement is in Section 3.3 of (Vidyasagar 1985), which gives a highly indirect expression for all controllers that simultaneously stabilize a pair of scalar plants. We believe that the problem of deriving an expression for all simultaneously stabilizing controller is hopelessly intractable in the general case. On the other hand, in the case where the plants all have either a single input or a single output, it is indeed possible to find such an expression, though it is a bit unwieldy. In the general case, it is possible to derive expressions that generate large families of simultaneously stabilizing controllers (though by no means all of them). We now describe the procedure in the former case, where the number of outputs equals one; the case where the number of inputs equals one is entirely similar. Note that the procedure is based on the contents of (Vidyasagar 1985, Section 7.6). The results in the general case are not described here in the interests of brevity.

Given the plants $P_1, \ldots, P_r$, find l.c.f.'s $(d_i, \tilde{N}_i)$ for each plant P_i, and form the matrix

$$M = \begin{bmatrix} d_1 & \tilde{N}_1 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ d_r & \tilde{N}_r \end{bmatrix} . \tag{1.18}$$

Now a controller $C = N_c d_c^{-1}$ simultaneously stabilized each plant P_i if, and only if each of the return differences

$$d_i d_c + \tilde{N}_i N_c = u_i \tag{1.19}$$

is a unit. Now define

$$U = \begin{bmatrix} u_1 \\ \cdot \\ \cdot \\ \cdot \\ u_r \end{bmatrix} \tag{1.20}$$

and note that

$$M \begin{bmatrix} d_c \\ N_c \end{bmatrix} = U \ . \tag{1.21}$$

Then a necessary and sufficient condition for C to simultaneously stabilize all plants is that each element of U is a unit. Thus the set of all controllers that achieve simultaneous stabilization is precisely the set of all solutions to (1.21). These solutions can be parametrized as follows. Find a matrix L such that L complements M, i.e., such that

$$A = \begin{bmatrix} L \\ M \end{bmatrix} \tag{1.22}$$

is a unit matrix, and define $S = L^{-1}$. Then the set of all solutions to (1.21) is

$$\begin{bmatrix} d_c \\ N_c \end{bmatrix} = S \begin{bmatrix} U \\ q \end{bmatrix} \ , \tag{1.23}$$

where q is an arbitrary vector of stable functions of the appropriate dimension. Now it is shown in (Vidyasagar 1985, Section 2.4) that every unit is of the form exp(m) for some analytic function m. Thus the set of all solutions to (1.21) is given by

$$\begin{bmatrix} d_c \\ N_c \end{bmatrix} = S \begin{bmatrix} \exp(m_1) \\ \cdot \\ \cdot \\ \cdot \\ \frac{\exp(m_r)}{q} \end{bmatrix} \ . \tag{1.24}$$

The conclusion is that it is quite feasible to formulate something like the canonical problem for the case of simultaneous design, and this is one of the objectives of the Phase II research effort.

Task 1.3: Design for a Time-Varying Operating Point

Thus far, we have discussed the design of controllers over a range of operating conditions as a simultaneous design problem. This presupposes that the purpose of the controller is to achieve good performance at any of a given set of static operating conditions. In reality, however, this is not what controllers are called upon to do. Going back to the earlier example of a flight control system, the controller is expected to maintain satisfactory performance as the flight conditions vary with time over a range of values. Thus, in order to be useful in practice, any synthesis methodology must be capable of tackling the case where the system is linear and time-varying. The objective of this subtask is to study this problem and propose some likely solutions. The next several paragraphs present some technical background on the graph metric which will play an important role in this subtask. Then the actual problem to be studied and the approach to be adopted are stated.

Suppose the system which we want to control is described by a set of linear differential equations with time-varying coefficients. These coefficients reflect the time variation of physical parameters. To be specific, suppose the system equations are of the form

$$\sum_{i=0}^{n} A_i(t)y^{(i)}(t) = \sum_{i=0}^{m} B_i(t)u^{(i)}(t) \ . \tag{1.25}$$

With this linear time-varying system, we can associate a family P_λ, $\lambda \geq 0$ of linear time-invariant systems, known as the frozen systems, which are defined as follows. As the name implies, the frozen systems are merely the time-varying system frozen at a particular time. Specifically, the frozen system $P_\lambda(s)$ is the system whose transfer matrix is

$$P_\lambda(s) = [A_\lambda(s)]^{-1}[B_\lambda(s)] \ , \tag{1.26}$$

where

$$A_\lambda(s) = \sum_{i=0}^{n} A_i(\lambda)s^i \ , \ B_\lambda(s) = \sum_{i=0}^{m} B_i(\lambda)s^i \ . \tag{1.27}$$

Now, if the time-varying system is slowly varying, then the problem of synthesizing a controller is quite a bit simpler than otherwise. But what does slowly varying mean? Intuitively, the time-varying system can be considered to be slowly varying if the rate of change of the systems P_λ, in the space of transfer matrices, is substantially smaller than the time constants of the individual (time invariant) transfer matrices P_λ. However, in order to make this intuition notion more precise, it is necessary to be able to define the rate of change of a system in some mathematically exact fashion. If we had a notion of distance, i.e., a metric, on the set of all transfer functions, then it is easy to do this.

Suppose such a metric is available, and call it d; thus $d(P_1, P_2)$ is the distance between two (not necessarily stable) transfer functions P_1 and P_2. Now we can examine the rate of change of the quantity $d(P_t, P_\tau)$ with respect to either argument to ascertain how rapidly the system is changing. For instance, for a fixed time t_0, the rate of change of the quantity $d(P_{t+t_0}, P_{t_0})$ gives an indication of how rapidly the system is changing at time t_0. If this rate of change is quite small compared with the dominant time constants of the plant P_{t_0}, then we can say that the system is slowly varying at time t_0. We believe that it is possible to combine this notion of a slowly varying system with known techniques for the synthesis of linear time-invariant systems to derive effective methods for the design of linear time-varying systems. The purpose of this task is to make the above ideas concrete.

The preceding paragraph shows that it is quite useful to have a notion of a metric on the set of transfer functions. Moreover, this metric should be reasonable on physical grounds from the standpoint of feedback stabilization. That is, if the distance between two plants $d(P, Q)$ is small, then a controller that stabilizes the plant P should also stabilize the plant Q, and moreover, the resulting stable feedback systems should have nearly identical responses. The graph metric, introduced in (Vidyasagar 1984), (Vidyasagar 1985, Ch. 7) has these properties. This metric and its principal properties are briefly recalled here in the interests of convenience.

Given a transfer matrix P, a right-coprime factorization (N, D) of P *is said to be normalized* if the matrix

$$A_p = \begin{bmatrix} D \\ N \end{bmatrix} \tag{1.28}$$

is inner. It can be shown that every rational matrix has a normalized r.c.f., and that it is unique within right multiplication by a real orthogonal matrix. Now suppose P and Q are two transfer matrices having the same dimensions, and let A_p, A_Q be normalized r.c.f.'s of P and Q respectively. Define

$$\delta(P,Q) = \underset{\|R\|_\infty \le 1}{\infty} \|A_p - A_Q R\|_\infty \,, \tag{1.29}$$

$$d(P,Q) = \max\{\delta(P,Q)\,, \delta(Q,P)\} \,. \tag{1.30}$$

Then $d(P, Q)$ is the graph metric distance between P and Q. Note that it is possible to define the graph metric between any pair of plants of the same dimensions, irrespective of their McMillan degrees, for example.

Next, suppose a plant P is stabilized by a controller C, and define

$$T(P,C) = [P \quad I](I + CP)^{-1} \begin{bmatrix} C \\ I \end{bmatrix} \,. \tag{1.31}$$

Then it is shown in (Vidyasagar and Kimura 1986) that feedback stability is preserved if P and C are changed to P_1 and C_1 respectively, provided

$$d(P,P_1)\|T(P,C)\|_\infty + d(C,C_1)\|T(C,P)\|_\infty < 1 . \tag{1.32}$$

The preceding paragraphs illustrate the efficacy of the graph metric in studying the robustness of feedback stability. Unfortunately, at present there is no precise way of computing the graph metric distance between two plants. However, in view of the results obtained in the Phase I effort, it is possible now to compute the distance to an arbitrary degree of accuracy. Observe that the problem of computing the distance $d(P,Q)$ is a particular instance of the canonical problem discussed in subtask 1, since the quantity $\delta(P,Q)$ is the value of the constrained optimization problem

$$\min \|A_p - A_Q R\|_\infty \text{ subject to } \|R\|_\infty \le 1 . \tag{1.33}$$

Having now demonstrated the feasibility of computing the graph metric distance between two plants, we now state the problem to be studied in this subtask. Suppose we are given a family of plants P_λ, where λ is a vector valued argument varying over a set Λ; λ could simply be time, or it could be a vector of physical parameters (e.g., altitude and Mach number in the case of aircraft). We wish to find a common controller C such that the feedback system shown in Figure 3-1 is stable under two situations:

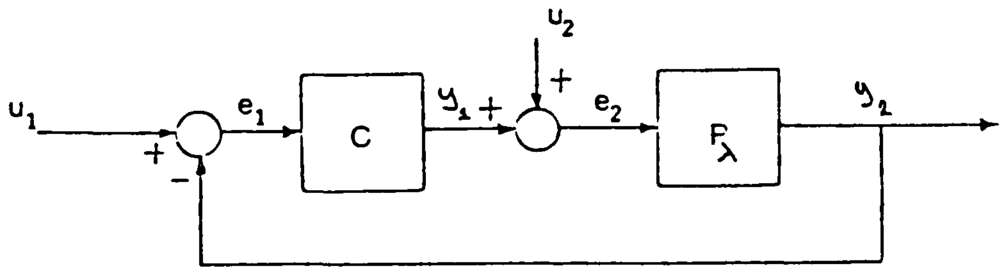


Figure 3-1. Feedback System with Varying Parameters

(i) when λ equals any constant vector in Λ, and (ii) when λ varies sufficiently slowly over the set Λ. The first problem, namely that of finding conditions under which an infinite family of plants is simultaneously stablizable, is generally studied under the heading of robust stabilization; see (Vidyasagar and Kimura 1986) for example. However, the second problem is quite new. The available theory of slowly varying systems is almost *entirely* stated in a state space setting; see (Vidyasagar 1978, Section 5.7) for a good summary of available results. It is thus a challenge to work an *input-output* version of the stability theory of slowly varying systems. Intuitively, one should be able to prove a theorem which says something like: If a controller C stabilizes each plant P_λ, $\lambda \in \Lambda$, and if each closed-loop transfer matrix has degree of stability μ, then the system of Figure 3-1 is stable even when λ is a function of time, provided $|\lambda(t)| \leq 1/\mu$. In other words, in order to assure the stability of a slowly varying system, the common controller C should not only stabilize the system, but also achieve a minimum degree of stability. Part of this subtask is to make the above intuitive notions precise, by finding a suitable mathematical definition for the concept of degree of stability.

There is another problem to be studied as a part of this subtask. Consider again the problem of designing a flight control system as a motivating example. In this case one is given a flight envelope, and it is usually quite unrealistic to expect that a single controller will prove to be satisfactory over the entire flight envelope. Under these conditions, the logical thing to do is to partition the flight envelope into several regions, such that within each region a single controller can be used. As the aircraft makes a transition from one region to another, the controller is switched to the appropriate one. What is being described above is a more general version of gain scheduling, which might be called controller and merely adjusts the parameters of the controller (the gains) as the plant moves from one regime into another, whereas we are proposing that the entire controller should be switched. This suggests several issues, such as: (1) How does one go about partitioning the flight envelope into the various regions? Is there a systematic way of choosing the regions so that they are as large as possible? (2) As the plant moves from one region into another and the controller is switched, how can one be certain that transient stability is assured? The method of designing controllers for each static operating condition does not say anything about the transient stability. We believe that, at the end of the Phase II effort, we will be able to address such issues. For the moment, we will simply point out how the two problems mentioned above are natural outgrowths of the various problems mentioned earlier as a part of Task 1.

First, consider the problem of partitioning a given flight envelope, or more generally a given set Λ is parameter space, into a union of subsets such that, within each subset Λ_i it is possible to design a common satisfactory controller. As a first attempt, this can be done by trial and error. In other words, once the engineer has in mind a particular partitioning, the problem of deciding whether or not there exists a common satisfactory controller within each subset of the partitioning is precisely the problem studied in subtask 1.1, and can be posed as a canonical optimization problem.

Next, consider the problem of assuring not only steady state but transient stability for the controlled system as the plant moves from one regime to another and the controller is switched appropriately. This is a natural extension of the problem stated earlier in this subtask, namely that of studying the stability of slowly varying systems from an input-output viewpoint. In the present instance, the plant is (presumably) slowly varying, but the controller is changing abruptly, i.e., switching from one configuration to another.

Summary

For the convenience of the reader the various research issues raised as a part of this task are briefly summarized below. These are not intended as complete problem statements, but merely as synopses of the problem statements.

1. Improve the efficiency of the solution procedures for the canonical optimization problem by seeking alternate representations of stable rational functions.
2. Find an analog to the canonical problem in the case of simultaneous design.
3. Develop a stability theory for slowly varying systems which is stated in an input-output framework, rather than a state space framework.
4. Study the transient stability of slowly varying systems when the controller is switched.

3.2 TASK 2: Transfer Function Estimation

The main objective of this task is to address some problems that arise in connection with the development of CACE primitives for the analysis of algorithms which estimate a plant transfer function from input-output data. This task will involve the development, where necessary, of the appropriate mathematical and computational tools, as well as the appropriate language architecture. As a by-product of our research we will also develop some mock-up software to test the numerical methods associated with the primitive.

There are a great many practical situations where a control system must be calibrated or adjusted on-line so as to account for changes in the plant or poor initial modeling. Of course a robust control will account for some changes, but the possible plant variations may be so large as to preclude a satisfactory performance. One is then led to a control *re*-design based on current records of plant input-output data, i.e., *adaptive control.* In this task and the next (Task 3, Section 3.3), we discuss two related aspects of adaptive control: (1) transfer function estimation, and (2) parameter adaptive algorithms.

This task will be organized into three major categories or sub-tasks, namely: (i)computing the mean-square-error, (ii) experiment design, and (iii) estimating model accuracy.

Task 2.1: Computing the Mean-Square-Error (MSE)

The estimation of a system's transfer function from input-output data has, of course, a long history, and we will not attempt to document that here. There are many excellent survey articles and textbooks that can be referenced, e.g., Jenkins and Watts (1968), Astrom and Eykhoff (1970), *Automatica: Special Issue on Identification*(Jan. 1981), Ljung and Soderstrom (1983), to name a few. These references clearly explain the theory and practice of both parametric and non-parametric methods of transfer function estimation. Parametric methods usually involve the minimization of some time-domain function of the parameters using an iterative or recursive algorithm. Non-parametric methods involve the computation of correlation functions and/or their respective spectral densities. In either case, it is possible to obtain theoretical results which provide asymptotic expressions, as the data record length increases, for the mean-square-error (MSE) of the transfer function estimate. In the context of our previous discussion of robust control (Section 3.1), it is precisely the MSE which reflects model accuracy.

The Problem Set-up

Suppose that the true plant system to be estimated can be described by the discrete-time relation

$$y(t) = P_o(q)u(t) + d(t) \tag{2.1}$$

(We use t to denote sample times, i.e., $t = 0, 1, 2, ...,$ etc; q is the shift operator where $qx(t) = x(t+1)$ and $q^{-1}x(t) = x(t-1)$. Also, we use the term transfer function to denote an operator which depends on q.) In (2.1), $y(t)$ and $u(t)$ are the measured input and output , $P_o(q)$ is the transfer function, and $d(t)$ is the disturbance. The vector $[u(t)\ y(t)]^T$ is assumed to have the spectral density matrix

$$S(\omega) = \begin{bmatrix} S_{uu}(\omega) & S_{ud}(\omega) \\ S_{du}(\omega) & S_{dd}(\omega) \end{bmatrix} = \int_{-\infty}^{\infty} R(\tau)e^{-j\omega\tau}\, d\tau \ , -\pi \leq \omega \leq \pi \tag{2.2}$$

and correlation matrix

$$R(\tau) = \mathcal{E}\{\begin{bmatrix} u(t)u(t+\tau) & u(t)d(t+\tau) \\ d(t)u(t+\tau) & d(t)d(t+\tau) \end{bmatrix}\} \tag{2.3}$$

where $\mathcal{E}(\cdot)$ is the expectation operator. It is further assumed that d(t) is the output of a linear system with transfer function $W_o(q)$ which is driven by white noise $v(t)$ of intensity σ_o. Thus,

$$d(t) = W_o(q)v(t) \tag{2.4}$$

and

$$\begin{aligned} S_{dd}(\omega) &= |W_o(e^{j\omega})|^2\sigma_o^2 \\ S_{ud}(\omega) &= W_o(e^{j\omega})S_{uv}(\omega) \end{aligned} \tag{2.5}$$

The estimation problem is to estimate $P_o(q)$ and $W_o(q)$ from the observed *finite* data record

$$z^N = \{y(t), u(t) : t = 1, ..., N\} \tag{2.6}$$

Parametric Methods

Parametric methods of identification proceed by first selecting a set of candidate models:

$$y(t) = P(q,\theta)u(t) + W(q,\theta)v(t), \quad \theta \in \mathcal{F} \subset \mathbf{R}\checkmark \tag{2.7}$$

The set $\mathcal{F}$ will be defined below, but any one choice of $\theta \in \mathcal{F}$ yields a fixed model in the set. Assuming that $v(t)$ is white noise, then the best prediction of $y(t)$ given data up to $t-1$ is

$$\hat{y}(t,\theta) = W^{-1}(q,\theta)P(q,\theta)u(t) + [1 - P^{-1}(q,\theta)]y(t) \tag{2.8}$$

The map $(y,u) \mapsto \hat{y}$ defined by (2.8) is referred to as the ***one-step ahead predictor***. Hence, the ***prediction error*** is

$$\begin{aligned} \varepsilon(t,\theta) &= y(t) - \hat{y}(t,\theta) & (2.9) \\ &= W^{-1}(q,\theta)[y(t) - P(q,\theta)u(t)] & (2.10) \end{aligned}$$

The set $\mathcal{F}$ is now defined as those $\theta \in \mathbf{R}^p$ such that the predictor is stable, thus $\mathcal{F} = \{\theta \in \mathbf{R}\checkmark = \mathcal{W}^{-\infty}$ and $W^{-1}P$ are stable.

A proceedure for selecting $\theta \in \mathcal{F}$ is to minimize some function of the prediction error. For example, let the estimate be

$$\theta_N = \arg\min_{\theta \in \mathcal{F}} V_N(\theta) \tag{2.11}$$

where

$$V_N(\theta) = \frac{1}{N} \sum_{t=1}^{N} \varepsilon^2(t,\theta) \tag{2.12}$$

Techniques for computing θ_N involve either iterative or recursive algorithms. These will be discussed in a later section.

Suppose we have computed θ_N. The next step is to form the transfer function estimates $P(q,\theta_N)$ and $W(q,\theta_N)$. It is important when using these estimates in control design to be confident of their accuracy. For example, we would like to know the convergence rate as well as the properties of the limiting estimates $P(q,\theta_*), W(q,\theta_*)$ with respect to the true system $P_o(q), W_o(q)$. A knowlege of the mean-square-error would determine the requisite robustness of control that is required to use the estimates for design. Expressions for the MSE which are asymptotic in the data length can be obtained. Usefull numerical methods do not as yet exist, although our Phase I feasibility study indicates that such methods can be developed in Phase II.

Asymptotic Expressions for the MSE

Expressions for the MSE are in general quite complicated, even the asymptotic results for large values of N. It is shown in Ljung(1985), under fairly weak conditions on the model set, that with probability one,

$$\lim_{N\to\infty} \theta_N = \theta_* \tag{2.13}$$

where

$$\begin{aligned} \theta_* &= \arg\min_{\theta \in \mathcal{F}} \bar{V}(\theta) \\ \bar{V}(\theta) &= \mathcal{E}[\varepsilon^2(t,\theta)] \end{aligned} \tag{2.14}$$

Moreover, the quantity

$$\sqrt{N}(\theta_N - \theta_*) \tag{2.15}$$

is asymptotically normal with zero mean and covariance matrix

$$R_* = [\bar{V}''(\theta_*)]^{-1} \lim_{N\to\infty} \mathcal{E}\{N[V_N'(\theta_*)]^T V_N'(\theta_*)\}[\bar{V}''(\theta_*)]^{-1} \tag{2.16}$$

where $'$ and $''$ denote differentiation with respect to θ, once and twice, respectively. We remark that these results hold for more general norms of the prediction error, i.e., not necessarily quadratic. The asymptotic expressions for the transfer function estimates can be obtained from the above results. It is convenient to introduce the operators

$$\begin{aligned} T(q,\theta) &= [P(q,\theta) \;\; W(q,\theta)] \\ T_o(q) &= [P_o(q) \;\; W_o(q)] \end{aligned} \tag{2.17}$$

These transfer operators represent the map $(v,u) \mapsto y$ for the plant model and true plant. Then, with probability one,

$$\lim_{N\to\infty} T(q,\theta_N) = T(q,\theta_*) \tag{2.18}$$

Moreover, the quantity

$$\sqrt{N}[T(e^{j\omega},\theta_N) - T(e^{j\omega},\theta_*)]^T \tag{2.19}$$

is asymptotically normal with zero mean and covariance matrix

$$Q_*(\omega) = [T'(e^{j\omega},\theta_*)]^T R_* T'(e^{j\omega},\theta_*) \tag{2.20}$$

Finally, the MSE of the transfer function estimate is

$$M_N(\omega) = \mathcal{E}\{B^T(e^{j\omega},\theta_N)B(e^{-j\omega},\theta_N)\} \tag{2.21}$$

where

$$B(q,\theta) = T(q,\theta) - T_o(q) \tag{2.22}$$

For large N the MSE is approximately

$$M_N(\omega) \approx B^T(e^{j\omega},\theta_*)B(e^{-j\omega},\theta_*) + \frac{1}{N}Q_*(\omega) \tag{2.23}$$

The quantity $B(e^{j\omega},\theta_*)$ is referred to as the *bias* and $(1/N)Q_*(\omega)$ is the *variance*.

On calculating the MSE

It is obvious that the expressions above for the MSE are quite complicated in the sense that they are beyond hand calculations. The expressions can be significantly simplified when the model *order* is large. Simulated data shows that the asymptotic result for large data length and model order is often valid for low orders, but this is not satisfactory for a quantitative analysis. Our interest is to develop numerical methods to handle the calculation for *any* model order, and this is a goal of Phase II.

At this point we can make two additional observations about the above expression for the MSE. In the first place,the result is valid for some large data length N, but there is no means to determine the error induced for any particular N. Secondly, since the asymptotic resullt depends on θ_*, it is necessary to calculate this limiting parameter estimate. Of course θ_* is never known to the user beforehand, and hence, is only a quantity whose properties are useful for ***analysis*** of the estimator. During Phase I we studied the feasibility of computing the asymptotic MSE above, as well as calculating θ_*. During Phase II we will develop the computational techniques and appropriate primitives. We also plan to study the problem of computing *non*-asymptotic expressions for the MSE, i.e., for any data length and model order.

Non-Parametric Methods

Parametric methods as we have discussed them, involve transfer function models which can depend on the parameters in a specified way. Such models can arise from physics and then the parameters have a physical meaning. Often canonical models are used where now the parameters can be taken as coefficients in a transfer function. In the former case there are fewer parameters but these enter into the transfer function coefficients in a complicated manner. In the latter case there are more parameters to learn , but they enter the model in a simple manner, thereby reducing computation. Methods which do not require such structural knowledge or assumptions are referred to as non-parametric; the only assumption is that the system has a transfer function, i.e., it can be described as in (2.1)-(2.5).

Available methods for non-parametric transfer function estimation rely on estimating spectral densities. The methods are based on time-series and Fourier analysis of finite data sequences. There are many excellent textbooks on the subject, e.g., Jenkins and Watts(1968), Brillinger(1975), and Priestly (1981). The reason that we consider non-parametric methods is that they can provide estimates of model accuracy as a function of frequency, particularly over those frequency ranges where either the model structure is poorly known, or else the model accuracy in that range is unimportant for control design.

For purposes of illustration, consider system (2.1)-(2.5) with the additional assumption that $u(t)$ and $d(t)$ are uncorrelated, i.e., $S_{ud}(\omega) = 0$. We then have the relations:

$$\begin{aligned} S_{yu}(\omega) &= P_o(e^{j\omega})S_{uu}(\omega) \\ S_{yy}(\omega) &= |P_o(e^{j\omega})|^2 S_{uu}(\omega) + S_{dd}(\omega) \end{aligned} \tag{2.24}$$

Suppose we have obtained estimates of $S_{yy}(\omega), S_{yu}(\omega)$ and $S_{uu}(\omega)$ from the finite data record z^N (2.6). Denote these estimates by $S_{yy}^N(\omega), S_{yu}^N(\omega)$, and $S_{uu}^N(\omega)$. Using (2.24) we can take as estimates of $P_o(e^{j\omega})$ and $S_{dd}(\omega)$ the following:

$$\begin{aligned} \hat{P}(\omega) &= S_{yu}^N(\omega)/S_{uu}^N(\omega) \\ \hat{S}_{dd}(\omega) &= S_{yy}^N(\omega) - |\hat{P}(\omega)|^2 S_{uu}^N(\omega) \end{aligned} \tag{2.25}$$

The properties of these estimates clearly depend on the ability to estimate spectral densities from a finite data record. The standard techniques involve data windowing in time and frequency, aligning, anti-aliasing filters, and many other proceedures which it is not possible to desribe here. The details can be found in the previously mentioned references. However, just as in the parametric case, it is also possible here to obtain expressions for the MSE. A typical expression approximate expression for large N is

$$\mathcal{E}\{|\hat{P}(\omega) - P_o(e^{j\omega})|^2\} \approx M^2(\gamma)|R(\omega)|^2 + \frac{1}{N}L(\gamma)S_{dd}(\omega)/S_{uu}(\omega) \tag{2.26}$$

where

$$R(\omega) = P_o''(e^{j\omega}) + P_o'(e^{j\omega})S_{uu}'(\omega)/S_{uu}(\omega) \tag{2.27}$$

with $'$ and $''$ denoting differentiation with respect to ω, once and twice, respectively. Also

$$\begin{aligned} M(\gamma) &= \int_{-\pi}^{\pi} \omega^2 W_\gamma(\omega)\, d\omega \\ L(\gamma) &= \int_{-\pi}^{\pi} W_\gamma^2(\omega)\, d\omega \end{aligned} \tag{2.28}$$

where $W_\gamma(\omega)$ is the *lag window* of width $1/\gamma$. The window is used to generate "smooth" spectral estimates and as seen in the above expressions can be used to adjust the MSE. Typically, as γ increases, the window becomes more narrow, $M(\gamma)$ decreases, and $L(\gamma)$ increases. Thus, as γ increases, the first term (the bias) decreases, but the second term (the variance) increases. Clearly for large N there is an optimal choice of lag window width to minimize the MSE for fixed N, and this can be calculated. Obviously hand calculations of the MSE are not possible for any problem of reasonable complexity. It is an objective of Phase II to develop techniques to efficiently perform the calculation. We also point out that (2.26) is an asymptotic expression, and as stated before , we are more concerned with computing the MSE for any finite N, one that is not necessarily large. Moreover, a quantitiative assessment of how large is large is not known at this time. We plan toexplore these computational issues in Phase II.

Summary

In this section we have indicated that the MSE is a function of the true plant, the estimation proceedure, and the length of the data record. Hence, computation of the MSE provides an *analysis* tool for determining the effectiveness of an on-line control adjusted scheme which is based on transfer function estimation. The feasibility of developing a primitive for MSE computation for both parametric and non-parametric methods of transfer function estimation was investigated during the Phase I effort. During Phase II we plan to address the computational and mathematical problems described above and the develop a primitive to perform the MSE computation. Appropriate language architectures will be investigated along with the development of some mock-up software.

Task 2.2: Experiment design

The MSE can also be used to design the estimation experiment, so that during the *actual* experiment the maximum amount of data is extracted in computing the transfer function estimate. This particular issue, referred to as *experiment design*, has been extensively investigated, e.g., see the above references. Recently ,there has been a specific interest in using these results in order to design experiments which optimize the use of the transfer function estimate in control design, e.g., Ljung(1985), Wahlberg and Ljung(1986), Gevers and Ljung(1986). These results minimize a weighted norm of the MSE with respect to free parameters or choices in the experiment, e.g., input spectrum, model order, etc. Since the MSE depends on the true plant, the methods offer usefull guidelines for designing the experiment. In Phase II we will develop computational proceedures for determining the various design variables.

At the present time the use of the MSE as a measure of experiment design for the case when the intended use of the transfer function estimate is control synthesis is still in the beginning stages. One of the major problems is that the MSE is not a *direct* measure of closed loop performance. During Phase II we plan to extend the known results to direct measures of control performance.

Task 2.3: Estimating model accuracy

As espoused in Task 1 (Section 3.1), the requisite information for robust stabilization is a nominal model of the plant together with an uncertainty profile. From the above discussion, computation of the MSE allows one to *evaluate* a control design which is based on a nominal (estimated) model together with an *a priori* estimate of model accuracy. During Phase I, we studied the feasibility of developing a proceedure by which model accuracy could be obtained, or estimated, from on-line data. The basic idea is to use both parametric and non-parametric methods of transfer function estimation, where the non-parametric methods yield direct estimates of the accuracy. Essentially, this approach provides a direct estimate of the MSE. The accuracy to which the estimate is set depends on the criteria for control design. Some preliminary results will be reported in Kosut(1987). The basic idea is to postulate a parametric model with two types of parameters, say $\alpha \in \mathbf{R}^k$ and $\beta \in \mathbf{R}^\ell$. The α parameters correspond to, say, physical parameters in the usual sense of model building, but the β parameters are auxiliary and used to account for poorly known aspects of the system dynamics. For example, in the parametric model (2.7) the θ parameters consist of both α and β type parameters. To illustrate the point further, let $P(q,\theta)$ in (2.7) have the decomposition

$$P(q,\theta) = \bar{P}(q,\alpha)\tilde{P}(q,\beta) \tag{2.29}$$

where $\bar{P}(q,\alpha)$ is the usual parametric model and where $\tilde{P}(q,\beta)$ is a simple structure which accounts for unmodeled dynamics. For example, high frequency unmodeled dynamics could be represented in a crude way by

$$\tilde{P}(q,\beta) = \frac{\beta_1 q}{q-\beta_2} \tag{2.30}$$

Obviously more elaborate forms can be established depending on the kind of a priori knowledge available regarding the location and type of unmodeled or poorly known dynamics. A proposed generic form is shown in Figure 3-2, where $G(q)$ is a known interconnection transfer matrix, $K(q,\alpha)$ contains all the known parametric stucture, and $L(q,\beta)$ represents the poorly known dyamics. The true plant would be represented by the corresponding triple $G(q), K_o(q), L_o(q)$.

This model isuseful for control design because the parameters α and β can be calculated from (2.11) [Kosut,1987)], and then a non-parametric method can be used to estimate the model error between the estimated system $G(q), K(q,\hat{\alpha}), L(q,\hat{\beta})$ and the true system $G(q), K_o(q), L_o(q)$. When this proceedure is coupled together with the sysnthesis methods discussed in Task 1(Section 3.1), we have an on-line robust control design scheme,

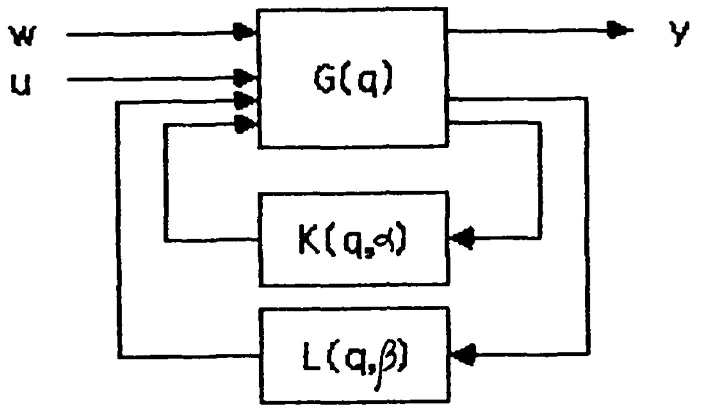


Figure 3-2. A generic parametric model structure

what is referred to as ***adaptive calibration.*** During Phase II, we plan to develop the appropriate primitives and mock-up software associated with the generic model structure in Figure 3-2, as well as developing the mathematical and computational tools for on-line model accuracy estimation.

3.3 TASK 3: Analysis of Parameter Adaptive Control

The main objective of this task is the development of a CACE primitive and mock-up software for the analysis of parameter adaptive control of a linear system. This task will involve the development, where necessary, of the appropriate mathematical and computational tools, as well as the requisite language architecture for user interface with the primitive. We have distributed the work in this task into three major areas: (i)adaptive control of linear systems, (ii) time-scale decomposition and averaging, ans (iii)fixed point analysis. Before describing these subtasks we will provide some background information.

Background and Current Status

Uncertainty underlies the reason for adaptive control. At one extreme there are physical processes which in many instances defy practical mathematical modeling, e.g., engine dynamics, structural joints, and welding, to name a few. At the other end of the spectrum are systems whose model structure is sufficiently well-known, but whose parameters are uncertain, e.g., large space structures, disc drives. Such uncertainty can significantly limit the performance attainable from a fixed model based control architecture. Therefore, the use of an adaptive control system gives more options for minimizing the risk in achieving performance objectives.

The aim of adaptive control is to implement in real-time and on-line as many as possible of the design functions now performed off-line. To realize this aim requires a theory of stability and robustness of such inherently nonlinear systems, the availability of software tools for analysis and design, and the hardware for implementation.

At the present time we stand a the beginning stages in the development of theory applicable to adaptive control systems. In the past few years these has been vigorous activity and debate on the question of the robustness of the adaptive control, i.e., what happens when ideal conditions are violated, as would be expected in practice. Although simulations of simple systems under apparently minor non-ideal conditions have shown degraded performance and even instabilities, e.g., Rohrs and co-workers (1981, 1982), there is equally convincing evidence of practical successes of complex industrial systems, e.g., Astrom (1981).

Motivated by these facts, many researchers pursued the robustness issue. The result of some of these efforts are contained in the recently published textbook: *Stability of Adaptive Systems: Passivity and Averaging Analysis*, MIT Press, 1986, Cambridge, Mass. The authors are B.D.O. Anderson, R.R. Bitmead, C.R. Johnson, Jr., P.V. Kokotovic, R.L. Kosut, I.M.Y. Mareels, L. Praly, and B.D. Riedle. This publication is obviously an outgrowth of considerable international collaboration among several researchers. * The material in the text represents refinements of earlier work: Reidle and Kokotovic (1985, 1986) on slow adaptation and the integral manifold; Astrom (1983, 1984) showing how the method of averaging explains instabilities and drift; Kosut, Anderson, and Mareels(1987) on the relation between averaging and persistent excitation; and Kosut and Anderson (1986), Kosut and Johnson(1985) on linearization and local stability.

Averaging: Uses and Limitations

Although the above mentioned text documents a well-defined mathematical research problem, the picture is not at all complete. In the first place, the method of averaging requires slow adaptation which can be counter-productive because performance can be below par for the long period of time it takes for the parameters to adjust. Secondly, the results are valid only for a portion of the state-space, in particular, the parameters are restricted to more within a subset of the constant parameter stability set. Another area of concern when using the method of averaging as an analysis tool, is that the speed of adaptation required to satisfy the theoretical conditions is in many instances far below that as determined from simulations. Thus, although slow adaptation allows for an analysis which provides quantitative robustness measures, there are inherent restrictions in the analysis.

Transient Analysis

To remove these restrictions requires understanding the *transient behavior* of adaptive systems. Methods for analyzing the transient were examined during Phase I. Some of the results are reported in Kosut et al. (1986) and Kosut and Bitmead (1986). This investigation of the transient properties shows that some of the interesting phenomena can be analyzed. The tools for analysis involve a combination of *small gain theory, passivity*, and the *method of averaging* with these all linked together by the *Contraction Mapping Principal*.

Beyond Hand Calculations

Although each of these tools, in principal, involves straightforward calculations,

* Two workshops (Aug. 1984, 1985) organized by Prof. P.V. Kokotovic (Univ. of Illinois) and separate visits by Dr. Kosut, Prof. Kokotovic, and Prof. C.R. Johnson (Cornell Univ.) to Prof. B.D. Anderson's group at the Australian National University played a significant part in these research efforts. Dr. Kosut's visits were supported by AFOSR Contract F49620-85-C-0094, NSF Grant INT-85-13400, and a Visiting Fellow Award from the Australian National University.

it is clear that the level of complexity of a realistic adaptive system is well beyond hand calculation. Hence, an area for further work is in the development of software tools which can eliminate some of the tedious parts of the analysis.

The same issue can in fact be raised regarding the slow adaptation analysis discussed earlier. Even simple examples can just barely be worked out by hand. At the present time there are no available "user-friendly" software tools for dealing principally with nonlinear systems. This is a research issue in both mathematics and computation, and it is one we feel is essential to a study of adaptive systems and the development of primitives for the Phase II effort.

Task 3.1: Adaptive Linear Control

In this section we will descibe in some detail our approach to developing the mathematical and computational tools for analyzing the parameter adaptive system shown below in Figure 3-3.

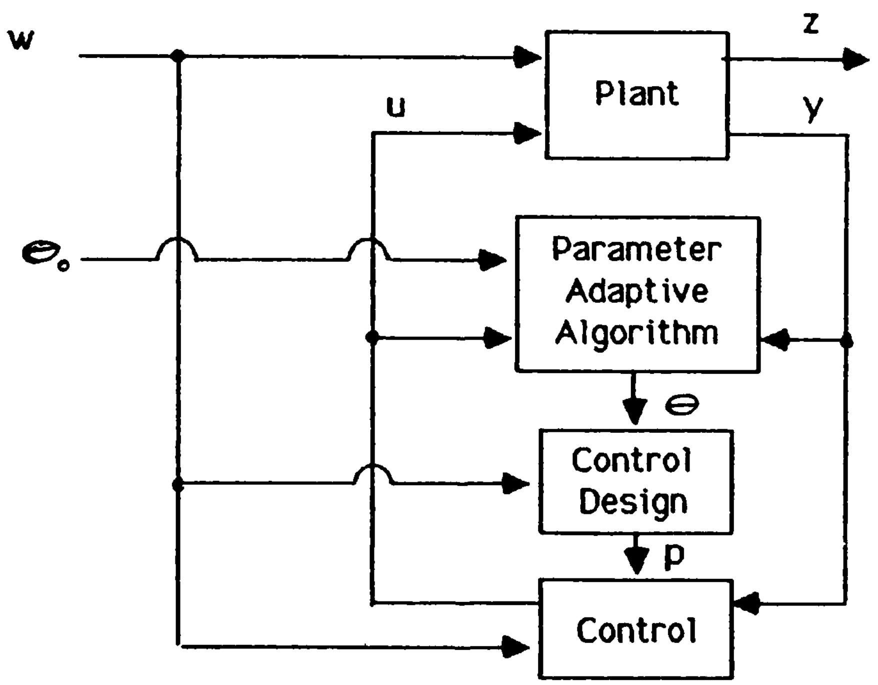


Figure 3-3. Parameter adaptive control

In this system $u(t)$ and $y(t)$ are the vectors of measured inputs and outputs, $w(t)$ is a vector of exogenous inputs,i.e., references, disturbances, and noise sources. The vector of adaptive parameters is $\theta(t)$, which is mapped into a control parameter vector $\rho(t)$ by some design rule, typically a memoryless nonlinear function implicitly defined by the design rule, e.g., LQG or optimal $\mathbf{H}_\infty$ methods, as discussed in Section 3.1(Task 1). The vector θ_o denotes the initial parameter value, say at $t = 0$. This system structure is valid for most parameter adaptive systems, be they continuous, discrete, or hybrid [see, e.g., Kosut and Bitmead(1986), Ljung and Soderstrom(1983)].

In order to illustrate the basic ideas, we will assume here that the parameter adaptive algorithm is continuously adjusted. In practice, parameter adjustments would be at discrete times or the control signal is the output of a digital computer. Suppose that the plant is linear-time-invariant, and for fixed values of the control parameter vector ρ the control is also linear-time-invariant. Then, the parameter adaptive system of Figure 3-3 may be described by the differential equations

$$\begin{aligned} \dot{x} &= A(\theta)x + B(\theta)w(t) \\ \dot{\theta} &= \gamma(t)q(x,\theta) \end{aligned} \tag{3.1}$$

where $\theta(t) \in \mathbf{R}^p$ is the adaptive parameter, and $x(t) \in \mathbf{R}^n$ is the system state, consisting of plant, controller, and filter states. The matrix functions $A(\theta), B(\theta)$ are determined by the design rule $\theta \mapsto \rho$ and the parametric controller structure. The nonlinear function $q(x,\theta)$ is determined by the choice of algorithm, and $\gamma(t)$ is the speed of adjustment, often referred to as the adaptation gain. For example, when using a recursive-least-squares (RLS) algorithm we have $\gamma(t) = 1/t$, whereas with a gradient based algorithm, such as least-mean-squares (LMS), we have $\gamma(t) = \gamma$, a positive constant.

Consider the simple gradient algorithm

$$\dot{\theta} = \gamma\phi\varepsilon \tag{3.2}$$

where $\phi(t) \in \mathbf{R}^p$ is referred to as the *regessor*, $\varepsilon(t)$ as the *prediction error*, and γ is the positive constant *adaptation gain*. In most cases we also have

$$\begin{aligned} \phi &= Dx \\ \varepsilon &= c^T x \end{aligned} \tag{3.3}$$

where $x(t) \in \mathbf{R}^n$ is the system state governed by (3.1) with $D \in \mathbf{R}^{p\times n}$ and $c \in \mathbf{R}^n$. Thus, the adaptive algorithm has the simpler form

$$\dot{\theta} = \gamma q(x) \tag{3.4}$$

where $q(x)$ is the polynomial in x given by

$$q(x) = c^T x D x \tag{3.5}$$

This algorithm has been used as a prototype for many studies, e.g., Anderson et al.(1986), Kosut et al.(1987). The genisis of the algorithm is to adjust $\theta(t)$ so that it asymptotically approaches a member of the set

$$\mathcal{F}_{opt} = \{\theta \in \mathbf{R}^p : \text{avg}[\varepsilon^2] \text{ is a minimum}\} \tag{3.6}$$

In the ideal case, the parametrization can be selected so as to acheive what is called *perfect matching*, i.e., for some restricted class of exogenous inputs (usually no disturbaces), the set $\mathcal{F}_{opt}$ has a single member such that $\varepsilon(t) = 0$. Under more realistic conditions, the best that can happen is that $\theta(t)$ asymptotically approaches a small neighborhood of

$$\mathcal{F}_* = \{\theta \in \mathbf{R}^p : \text{avg}[\phi\varepsilon] = 0\} \tag{3.7}$$

If for fixed θ, the regressor $\phi(t)$ is constructed to be identical with $-\partial\varepsilon(t)/\partial\theta$, then $\mathcal{F}_*$ is the set of all *local minima* of avg$[\varepsilon^2]$. Since $\partial\varepsilon(t)/\partial\theta$ is a function of the true, but unknown plant, the regessor can only realistically be constructed as an approximation. From a practical point of view it is acceptable that the parameters approach and remain in a small neighborhood of $\mathcal{F}_*$, provided that members of this set also produce acceptable closed loop performance.

Assuming this is so, let $\theta_* \in \mathbf{R}^p$ denote such a setting, of which there could be many. We refer to each θ_* as a *tuned parameter* and to the corresponding system

$$\begin{aligned} \dot{x}_* &= A(\theta_*)x + B(\theta_*)w(t) \\ \varepsilon_* &= c^T x_* \\ \phi_* &= D x_* \end{aligned} \tag{3.8}$$

as the *tuned system* [see, e.g., Kosut and Friedlander(1985)]. Clearly (3.8) is the same as system (3.1) but with $\theta(t)$ fixed at θ_*. We can now pose the following questions regarding the adaptive system (3.1):

(1) How do the tuned parameters depend on the exogenous inputs?

(2) Is the adaptive system stable in a neighborhood of the tuned system, i.e., are solutions $\{x(t), \theta(t)\}$ stable near $\{x_*(t), \theta_*\}$?

(3) What is the region of attraction in (x, θ) to a small neighborhood of $\{x_*(t), \theta_*\}$?

The primitives we envision for answering these questions would accept as an input the adaptive system described by (3.1). Our mock-up software would, for example, translate the block diagram of Figure 3-3 into (3.1) through a formal language architecture.

Task 3.2: Time-Scale Decomposition and Averaging

Time-Scale Decomposition

To see more explicitly what is involved, particularly where some mathematical and computational research is required, we will proceed with our example sysytem (3.1),(3.2). Following the proceedures given in Anderson et al.(1986), we study the behavior of (3.1) in the neighborhood of *all* constant parameter solutions. For this purpose, let $\nu(t,\theta)$ denote the state $x(t)$ when $\gamma = 0$. We refer to $\nu(t,\theta)$ as the *frozen parameter system state*, or *frozen state* for short. Hence, for each $\theta \in \mathbf{R}^p$, $\nu(t,\theta)$ satisfies the *partial* differential equation

$$\partial\nu/\partial t = A(\theta)\nu + B(\theta)w(t) \tag{3.9}$$

By introducing the error state

$$\eta(t) = x(t) - \nu(t,\theta(t)) \tag{3.10}$$

the (x,θ)-system of (3.1) can be transformed into the equivalent (η,θ)-system:

$$\begin{aligned} \dot{\theta} &= \gamma f(t,\theta,\eta) \\ \dot{\eta} &= A(\theta)\eta - \gamma g(t,\theta,\eta) \end{aligned} \tag{3.11}$$

where the functions f and g are given by

$$\begin{aligned} f(t,\theta,\eta) &= q(\nu(t,\theta)+\eta) \\ g(t,\theta,\eta) &= [\partial\nu(t,\theta)/\partial\theta]f(t,\theta,\eta) \end{aligned} \tag{3.12}$$

The transformation of (3.1) into (3.11) is referred to as a *time-scale decomposition*, because in general, $\theta(t)$ changes much more slowly than $\eta(t)$. This is certainly the case whenever γ, the adaptation gain, is small. But it is also true whenever $\theta(t)$ is near convergence. Since the (η,θ)-system of (3.11) is equivalent to the original (x,θ)-system of (3.1), the answers to the questions posed before will involve the analysis of (3.11). For example, determining the tuned parameter set defined by (3.7) means solving for those $\theta_* \in \mathbf{R}^p$ which satisfy

$$\text{avg}[f(\cdot,\theta_*,0)] = 0 \tag{3.13}$$

Out of all possible solutions, select only those which also satisfy performance criteria for the tuned system (3.8), e.g., $\text{Re}\,\lambda[A(\theta_*)] \le -\beta_*$, where β_* is a positive constant which specifies the damping requirement.

Averaging

A further restriction of those $\theta_* \in \mathbf{R}^p$ satisfying (3.13) is that solutions (x, θ) of (3.1) which originate in a small neighborhood of the tuned solutions (x_*, θ_*) should remain there. Under fairly mild "smoothness" conditions on the functions f and g in (3.12), it is shown in Anderson et al(1986), that for a sufficiently small adaptation gain γ, and a sufficiently small peak value of the tuned error signal $\varepsilon_*(t)$, solutions of (3.1) originating in a small neighborhood of the tuned system (x_*, θ_*) will remain there if

$$\max_i \operatorname{Re} \lambda_i[B(\theta_*)] < 0 \tag{3.14}$$

and, moreover, will leave there if

$$\max_i \operatorname{Re} \lambda_i[B(\theta_*)] > 0 \tag{3.15}$$

where the matrix function $\theta \mapsto B(\theta)$ is given by

$$B(\theta) = \frac{\partial}{\partial \theta} \operatorname{avg}[f(\cdot, \theta, 0)] \tag{3.16}$$

The sharp stability-instability boundary expressed by (3.14) and (3.15) allows not only for an assessment of a particular design, but also indicates how to modify and improve the algorithm. In particular we can also show that under the conditions stated above, if (3.14) holds then

$$\begin{aligned} \limsup_{t\to\infty} \|\theta(t) - \theta_*\| &= O(\limsup_{t\to\infty} |\varepsilon_*(t)|) + O(\gamma) \\ \limsup_{t\to\infty} \|\eta(t)\| &= O(\gamma) \end{aligned} \tag{3.17}$$

Specific expressions can be obtained for the right hand sides above, but these are usually complicated, and more importantly, are conservative. Methods to reduce the conservatism was explored in Phase I. These methods involve the use of fixed point analysis which we will now briefly descibe. It is worth mentioning that fixed point theory is also the fundamental tool behind the derivation of (3.14)-(3.17), together with the method of averaging. As we will see from the discussion to follow, and as was discovered during Phase I, averaging is not required.

Task 3.3: Fixed Point Analysis

Here we will descibe a very powerfull nonlinear analysis tool, namely the Banach contraction mapping principal,which among other possibilities, enables one to compute the rate of convergence and region of attraction for the adaptive system. Actually, the adaptive system as represented by (3.11) can be analyzed by calling upon one of several fixed point theorems, as explained in Kosut and Bitmead(1986), and Kosut(1987). For example, the *Banach Fixed Point Theorem*, sometimes referred to as the *Contraction Mapping Theorem*(CMT), states that an operator Γ has a unique fixed point in a closed subset $\mathcal{M}$ of a Banach space if Γ is *contractive* on $\mathcal{M}$, e.g., Hale(1969). Referring to (3.11) we take Γ as the mapping of functions $\bar{\theta}(t)$ into functions $\theta(t)$ defined implicitly as follows:

$$\begin{aligned} \dot{\theta} &= \gamma f(t, \theta, \bar{\eta}) \\ \dot{\bar{\eta}} &= A(\bar{\theta})\bar{\eta} - \gamma g(t, \bar{\theta}, \bar{\eta}) \end{aligned} \tag{3.18}$$

Observe that fixed points of Γ in $\mathcal{M}$, i.e., those functions $\theta \in \mathcal{M}$ which satisfy the operator equation

$$\theta = \Gamma\theta \tag{3.19}$$

are solutions in $\mathcal{M}$ of the parameter trajectories of the adaptive system (3.11), or equivalently (3.1). For example, the results (3.14)-(3.17) are arrived at by choosing

$$\mathcal{M} = \{\theta \in \mathcal{C}[0, \infty) : \|\theta(t) - \theta_*\| \le r_o + r_1 \exp(-\sigma t)\} \tag{3.20}$$

where $\mathcal{C}[0, \infty)$ is the Banach space of continuous bounded functions, and r_o, r_1, and σ are positive constants.

In the process of establishing that Γ is contractive on $\mathcal{M}$, we utilize the method of averaging to establish the stability, near θ_*, of

$$\dot{\theta} = \gamma f(t, \theta, 0) \tag{3.21}$$

This is the origin of condition (3.14). It is important to point out that neither averaging nor small γ is required to establish the stability of (3.21) near θ_*. For example, as we discovered in Phase I, if the function $f(t, \theta, 0)$ is periodic in t uniformly for θ in a compact set, then stability of (3.21) near θ_* can be established by linearization and Floquet Theory [Kosut(1987)]. Condition (3.14) is replaced by a weaker condition and the limitation on the allowable size of the adaptation gain is considerably reduced over that imposed by averaging theory. During Phase II, we plan to use these ideas to develop a primitive which numerically establishes the conditions for Γ to be contractive on $\mathcal{M}$. Establishing such conditions can be viewed as a *canonical problem* in analyzing the stability properties of adaptive systems. Observe that because the contraction analysis considers operators in Banach spaces, the same results apply to discrete-time or hybrid adaptive systems, i.e., any linear adaptive control of a linear plant.

Summary

In this section we have briefly indicated that the analysis of linear adaptive systems is mathematically equivalent to the study of system(3.11), i.e.,

$$\begin{aligned}\dot{\theta} &= \gamma f(t,\theta,\eta) \\ \dot{\eta} &= A(\theta)\eta - \gamma g(t,\theta,\eta)\end{aligned} \tag{3.22}$$

Under ideal conditions, the adaptive parameter $\theta(t)$ asymptotically approaches a constant $\theta_* \in \mathbf{R}^p$ which satisfies

$$f(t,\theta_*,0) = 0 \tag{3.23}$$

and the error state asymptotically approaches zero. When the ideal conditions no longer hold, as is normally the case, the best to hope for is that $\theta(t)$ will now approach a small neighborhood of θ_*, which is now a solution of

$$\text{avg}[f(\cdot,\theta_*,0)] = 0 \tag{3.24}$$

Moreover, the error $\eta(t)$ will not asymptotically vanish, but will become small with a zero average.

During Phase II we propose to develop an analysis primitive which will quantify the behavior of (3.22). In particular, the primitive will answer questions about its asymptotic and transient characteristics, such as:

(1) *Asymptotic analysis*: What are the stability properties of (3.22) in the neighborhood of $(\theta,\eta) = (\theta_*,0)$?

(2) *Transient analysis*: What is the rate of convergence and region of attraction of (3.22) to a small neighborhood of $(\theta,\eta) = (\theta_*,0)$?

We have shown that the above characteristics can be extracted from (3.22) by performing a fixed point analysis of the operator $\Gamma : \bar{\theta} \to \theta$ defined implicitly by (3.18). Since fixed points of Γ in a Banach subspace $\mathcal{M}$ are parameter trajectories $\theta(t)$ which solve (3.22), it follows that developing a primitive for determining conditions for Γ to be contractive on $\mathcal{M}$ provides a reasonable analysis approach. During Phase II we will develop and refine the fixed point analysis tool so that the primitives would optimize the contraction conditions, thereby resulting in a non-conservative analysis.

3.4 TASK 4: Language Architecture for Control Design

The objective of this task is to make the results of the research carried out in the first three tasks widely accessible by incorporating them in a language specially tailored for control design. The task will consist of identifying suitable primitives for the language, coming up with a suitable syntax for the language, and finally developing some mock-up software to demonstrate the feasibility of the language architecture. Clearly the specific details of the language architecture can be decided upon only after the completion of the first three tasks. Thus in this section we describe the general philosophy that will guide our language development efforts. To make the ideas clearer, we will make analogies with existing software such as MATLAB, MATRIX$_X$ and SFPACK where appropriate.

By now it is universally accepted that numerical computations are best done in FORTRAN. The existing software subroutines such as LINPACK, EISPACK, IMSL and son on have been validated through usage by innumerable users, and it would be foolhardy to attempt a translation into another language. Software developers have thus concentrated on retaining these FORTRAN subroutines as the basic building blocks around which a superstructure can be built up in another language. For example, the original MATLAB not only used FORTRAN subroutines, but also a command parser written in FORTRAN, whereas the recent PC-MATLAB has a compiler written in C. Another trend has been the introduction of higher level commands at an ever-increasing level. Thus MATLAB permitted the user to define executive routines which sere essentially macros but did not permit the substitution of variable names; in contrast, recent software such as MATRIX$_X$ and SFPACK permit user-defined functions. Whereas MATLAB functions for order reduction, $\mathbf{H}_\infty$ norm minimization, etc. By the word *primitives*, we mean these basic building blocks. In MATLAB, operations such as taking the exponential of a matrix, or carrying out its singular value decomposition, etc. are the primitive operations, and the user must express whatever he wishes to do in terms of these primitive operations. In SFPACK, the primitives include the original MATLAB primitives, but in addition contain much higher level commands such as finding an $\mathbf{H}_\infty$-norm minimizing controller, given the plant and the desired McMillan degree of the reduced order approximation. Obviously, the availability of these higher level primitives makes the designer's task much easier. In the new proposed software, an obvious primitive would be the solution of the canonical problem, given the various parameters of the optimization problem. But other likely candidates will emerge during the course of the research.

The language architecture consists of the manner in which problems are stated and solutions are presented back to the user. Consider for instance the problem of designing a controller at a single operating point, as described in subtask 1.1. As mentioned earlier, this problem can be solved by posing it as a canonical optimization problem. However, from the standpoint of the user, it is much more natural to specify the problem in terms of the design constraints rather than in terms of the various F_i, G_i matrices arising in Equation (1.5). In developing a suitable language architecture, we will take such issues into consideration. An illustration of a user-interface which is very natural from a control designer's standpoint is given in (Boyd et al. 1986).

The final objective of this task is to develop some mock-up software which illustrates the feasibility and applicability of the research results developed in Tasks 1 to 3. By mock-up software we mean software which is essentially complete from the standpoint of the numerics and the algorithmic procedures, but which requires further work on the user interface, input-output formatting and the like. The mock-up software can then be turned over to commercial software developers who can then proceed further and come up with a commercially viable product similar to MATRIX$_X$.

References

B.D.O. Anderson, R.R. Bitmead, C.R. Johnson, Jr., P.V. Kokotovic, R.L. Kosut, I.M.Y. Mareels, L. Praly, and B.D. Riedle, *Stability of Adaptive Systems: Passivity and Averaging Analysis*, MIT Press, 1986.

K.J. Astrom and P. Eykhoff (1971), "System Identification—A survey", *Automatica*, 7:123–167.

M. Bodson, S. Sastry, B.D.O. Anderson, I. Mareels, and R.R. Bitmead, (1986), "Nonlinear Averaging Theorems, and the Determination of Parameter Convergence Rates in Adaptive Control", *Systems and Contr. Letters*, to appear.

S. Boyd et al. (1986), "A New CAD Method and Associated Architectures for Linear Controllers", ISL Tech. Report L-104-81,1, Stanford University, Dec. 1986.

D.R. Brillinger (1975), *Time Series: Data Analysis and Theory*, Holt, Rinehart, and Wintron, New York.

R.R. Bitmead, A.C. Tsoi, and P.J. Parker (1984), "A Kalman Filtering Approach to Short-Time Fourier Analysis", ANU Report, Canberra, Australia, Feb. 1984.

C.A. Desoer, R.W. Liu, J. Murray and R. Sacks (1980), "Feedback System Design: The Fractional Representation Approach to Analysis and Synthesis", *IEEE Trans. Aut. Control*, **Vol. AC-25**, pp. 399–412, June 1980.

C.A. Desoer and M. Vidyasagar, *Feedback Systems: Input-Output Properties*, Academic Press, 1975.

J.C. Doyle and C.C. Chu (1985), "Robust Control of Multivariable and Large Scale Systems", *Honeywell SRC: Final Technical Report* on AFOSR Contract F49620-84-C-0088, March 1986.

M. Gevers and L. Ljung (1986), "Optimal Experiment Designs with Respect to the Intended Model Application",*Autmatica*, **22**(5):543–554..

B.K. Ghosh and C.I. Byrnes, "Simultaneous Stabilization and Simultaneous Pole-Placement by Non-Switching Dynamic Compensation", *IEEE Trans. Auto. Contr.*, **AC-28**:735–741, June 1983.

K. Glover, "All Optimal Hankel-Norm Approximations of Linear Multivariable Systems and Their L_∞-error Bounds", *Int. J. Control*, **Vol. 39**; pp. 1115–1193, June 1984.

J.K. Hale (1980), *Ordinary Differential Equations*, Kreiger, Molaban, FL, originally published (1969), Wiley (Interscience), New York.

G.M. Jenkins and D.G. Watts (1968), *Spectral Analysis and Its Applications*, Holden-Day, San Francisco.

R.L. Kosut,† B.D.O. Anderson, and I.M.Y. Mareels, "Stability Theory for Adaptive Systems: Methods of Averaging and Persistency of Excitation", *IEEE Trans. on Aut. Contr.*, **Vol. AC-32**, No. 1, pp. 20–34, Jan. 1987.

R.L. Kosut (1987) "Conditions for Convergance and Divergence of Parameter Adaptive Systems", to be presented at *ISCAG*, Philadelphia, PA, May 1987.

R.L. Kosut, "Floquet Anaysis of Adaptive Systems", *Fifth Yale Workshop on Applications of Adaptive Systems Theory*, New Haven, CT, May 1987.

R.L. Kosut (1987), "Adaptive Uncertainty Estimation", *1987 ACC* , to appear, Minneapolis, MN, June 1987.

R.L. Kosut,†I.M.Y. Mareels, B.D.O. Anderson, R.R. Bitmead, and C.R. Johnson, Jr., "Transient Analysis of Adaptive Control", to be presented at the *IFAC, 10th World Congress*, Munich, Germany, July 1987.

R.L. Kosut and R.R. Bitmead (1986), "Fixed Point Theorems for the Stability Analysis of Adaptive Systems", *IFAC Workshop on Adaptive Control*, **AC-30**:834.

R.L. Kosut,"Adaptive Calibration: An Approach to Uncertainty Modeling and On-Line Robust Control Design", *Proc. 25th IEEE CDC*, Ahtens, Greece, Dec. 1986.

R.L. Kosut, "Towards and On-Line Procedure for Automated Robust Control Design: The Adaptive Calibration Problem", presented at *1986 ACC*, June 1986.

R.L. Kosut, B.D.O. Anderson, and I. Mareels (1985), (1987), "Stability Theory for Adaptive Systems: Methods of Averaging and Persistency of Excitation", *Dec. and Contr.*, Ft. Lauderdale, FL.; *IEEE Trans. Aut. Contr.*, **Vol. AC-32**, No. 1, pp. 26–34, Jan. 1987.

R.L. Kosut and B.D.O. Anderson, "Local Stability Analysis for a Class of Adaptive Systems", *IEEE Trans. Auto. Contr.*, Jan. 1986.

R.L. Kosut and B. Friedlander (1985), "Robust Adaptive Control: Conditions for Global Stability", *IEEE Trans. on Aut. Contr.*, **AC-30**(7):610–624.

R.L. Kosut and C.R. Johnson, Jr. (1984), "An Input-Output View of Robustness in Adaptive Control", *Automatica*, **20**(5)569–581.

L. Lehman, "RT_BUILD: A Graphical Implementation Tool for Ada Real-Time Control Software", *Proc. 2nd IEEE Symp. on CACSD*, Santa Barbara, CA, 1985.

† Research performed while R.L. Kosut was a Visiting Fellow at the Australian National University.

L. Ljung (1985), "Asymptotic Variance Expressions for Identified Black-Box Transfer Function Models", *IEEE Trans. on Aut. Contr.*, **AC-30**:834.

L. Ljung (1985), "On the Estimation of Transfer Functions", *Autmatica*, **21**(6), Nov. 1985.

L. Ljung and T. Soderstrom (1983), *Theory and Practice of Recursive Identification*, MIT Press.

K.D. Minto and M. Vidyasagar, "A State-Space Approach to Simultaneous Stabilization", *Control: Theory and Advanced Tech.*, (to appear).

K.D. Minto (1985), *Design of Reliable Control Systems; Theory and Computation*, Ph.D. Thesis, University of Waterloo, Sept. 1985.

R.A. Pine, R.A. Walker, and N.K. Gupta, "MATRIX$_X$/PC Modeling, Simulation, and Optimization on Personal Computers", *IEEE Proceedings*, **Vol. 73**, No. 12, Dec. 1985.

M.B. Priestley (1981), *Spectral Analysis and Time Series*, Academic Press, New York.

B.D. Riedle and P.V. Kototovic (1986), "Integral Manifolds of Slow Adaptation", *IEEE Trans. Aut. Control*, **Vol. 31**, No. 4, pp. 316–324, April 1986.

B.D. Riedle and P.V. Kokotovic (1985), "A Stability-Instability Boundary for Disturbance-Free Slow Adaptation and Unmodeled Dynamics", *IEEE Trans. on Aut. Contr.*, **AC-30**:1027–1030.

R. Saeks and J. Murray, "Fractional Representation, Algebraic Geometry and the Simulationeous Stabilization Problem", *IEEE Trans. Auto. Contr.*, **AC-237**:895–903, Aug. 1982.

S. Shah and S. Houtchens, "A Workbench for Design and Mechanization of Multivariable Nonlinear and Adaptive Controller, *Proceedings 1985 ACC* , Boston, MA, June 1985.

M. Vidyasagar, B.C. Levy and N. Viswanadham, "A Note on the Genericity of Simultaneous Stabilizability and Pole Assignability", *Circuits, Systems and Signal Processing*, (to appear).

M. Vidyasagar (1985), *Control system Synthesis: A Factorization Approach*, MIT Press, Cambridge, MA, 1985.

M. Vidyasagar, "The Graph Metric for Unstable Plants and Robustness Esstimates for Feedback Stability", *IEEE Trans. Auto. Contr.*, **AC-29**:403–418, May 1984.

M. Vidyasagar and H. Kimura (1986), "Robust Controllers for Uncertain Linear Multivariable Systems", *Autmatica*, 1986.

M. Vidyasagar and N. Viswanadham, "Algebraic Design Techniques for Reliable Stabilization", *IEEE Trans. Auto. Contr.*, **AC-27**:1085–1095, Oct. 1982.

M. Vidyasagar, *Nonlinear systems Analysis*, Prentice Hall, 1978.

G. Zames and B.A. Francis, "A New Approach to Classical Frequency Methods: Feedback and Minimax Sensitivity", *IEEE Trans. Auto. Contr.*, **AC-28**:585–601, May 1983.

G. Zames, "Feedback and Optimal Sensitivity: Model Reference Transformations, Multiplicative Seminorms and Approximate Inverses", *IEEE Trans. Auto. Contr.*, **AC-26**:301–320, April 1981.