(2)

AGARD-AR-206

AGARD-AR-206

AD-A166 617

AGARD

ADVISORY GROUP FOR AEROSPACE RESEARCH & DEVELOPMENT

7 RUE ANCELLE 92200 NEUILLY SUR SEINE FRANCE

AGARD ADVISORY REPORT No.206

Final Report on FMP Working Group WG-12 on Validation of Missile System Simulation

NORTH ATLANTIC TREATY ORGANIZATION

DISTRIBUTION AND AVAILABILITY ON BACK COVER

AGARD-AR-206

NORTH ATLANTIC TREATY ORGANIZATION

ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT

(ORGANISATION DU TRAITE DE L'ATLANTIQUE NORD)

AGARD ADVISORY REPORT No.206

FINAL REPORT of the FLIGHT MECHANICS PANEL WORKING GROUP WG-12 on VALIDATION OF MISSILE SYSTEM SIMULATION

This Advisory Report was prepared at the request of the AGARD Flight Mechanics Panel.

THE MISSION OF AGARD

Published November 1985

Copyright © AGARD 1985
All Rights Reserved

ISBN 92-835-1514-5

Printed by Specialised Printing Services Limited
40 Chigwell Lane, Loughton, Essex IG10 3TZ

PREFACE

One of the major findings in the AGARD Flight Mechanics Panel (FMP) AGARDograph No.279, "Survey of Missile Simulation and Flight Mechanics Facilities in NATO", by Willard M.Holmes, was the fact that, "Notably missing from a vast majority of facilities were any established or formal procedures for accomplishing any level of simulation model validation." The AGARD FMP felt this omission warranted further action and subsequently recommended the formation of Working Group 12, "Validation of Missile System Simulation".

Once approved by the AGARD National Delegates, Working Group 12 held four meetings over the 1982—1984 time period as follows: (a) 21 October 1982, London, England; (b) 5—6 May 1983, München, West Germany; (c) 24—26 October 1983, Eglin AFB FL, USA, and finally (d) 27—29 March 1984, Paris, France.

This report documents the working group findings. The working group consisting of Mr M.J.Douat, France; Mr Werner Bub, Germany; Mr Klaus Hausel, Germany; Mr Karl Ernst Platt, Germany; Mr Amilcare Gazzina, Italy; Mr B.J.Wanstall, United Kingdom; Mr D.Hyde, United Kingdom; Dr Willard Holmes, United States; Mr Lawrence Byrd, United States (representing the AGARD Guidance and Control Panel), and Mr Ronald Anderson, United States.

RONALD O.ANDERSON
FMP Member
Chairman, WG-12

CONTENTS

1. INTRODUCTION

The AGARD Flight Mechanics Panel (FMP) has been concerned with the problem of missile simulation in general, and cooperative NATO developments of missile system simulations, test, and evaluation as related to missile system flight mechanics in particular. As a result, the FMP sponsored a survey of missile simulation facilities reported in AGARDograph No. 279, "Survey of Missile Simulation and Flight Mechanics in NATO," by Willard M. Holmes, April 1983. One of the key findings in this survey was that very little effort is being expended in missile simulation validation. Furthermore, validation techniques themselves are generally not standard; often being ill-defined or undocumented.

This fact motivated the formation of FMP Working Group 12, "Validation of Missile System Simulation." This report documents the group findings.

Although there is no shortage of reference material on missile simulations per se, there seems to be no "history" on the subject, like the manned simulation community, (e.g., The Pilot Maker, by Floyd L. Kelly as told to Robert B. Parks, Grosset and Dunlap, 1970). The manned simulation world, founded first in training simulators and later in research and development simulators, is also just now realizing the need for simulation validation.

Perhaps, in both cases, the evolution of simulation equipment from extremely crude mechanical devices to modern electronic "arcade game simulators" happened so fast that it is only recently that people began to ask, "Do they really work"? Or, perhaps the early equipment reliability problems were such that everyone was quite happy if the simulator appeared to be "working", let alone providing "useful" results.

In any event, times have changed, and in both manned aircraft simulations and missile simulations the question of validation is becoming of more and more serious concern.

But where does one start in terms of defining how to validate missile simulations when few organizations do any formal validation now and there are no agreed upon standards whatsoever? This was the extremely broad and difficult question the working group had to repeatedly ask itself.

After a number of false starts, the working group concluded that the most important objective was to find a "method" of organizing simulation validation techniques rather than a collection of actual methods. In addition, the concept of "confidence" in missile simulations began to play a dominant role in the group thinking from the very start of the activity. That is, the purpose of the validation was as important as, or perhaps more important than, the methods themselves. In addition, many of the thoughts on validation methods seemed much easier to organize if one thought in terms of developing confidence in the behavior of a simulated model. Therefore, a perhaps larger portion of the group effort was expended on a hierarchical model representation of Confidence Level in Model Behavior (CLIMB) that included documentation that is integral to the total process to insure "confidence" in the final result. In fact, this documentation may instill more confidence than originally envisioned. This "organization of thought" provides one of the major portions of the group final results.

The remainder of this report is organized as follows. A Terminology Section is provided to include some fundamental definitions. While not intended to be "final", this section presents a good start on a standard set of terms. The next section on the CLIMB process forms the heart of this report and is by far the most important contribution the group could provide within the time span available. Hopefully, this section will form a basis for further work in the area plus a stand-alone start toward a unified approach to missile system simulation validation.

The next section on Computer Languages could also be a separate working group topic, being only touched upon here.

Next, a brief section on Software Validation/Verification/ Assessment Methods follows. This section could easily be the foundation of a working group report by itself, but this task was clearly beyond the two-year effort of the present activity.

Finally, overall recommendations are presented. As might be expected, the extremely difficult subject has led to a number of suggested follow-on activities for FMP consideration.

Throughout the working group activity, the members were extremely conscientious and devoted to their task. Each meeting generated a mound of documentation to be shared by the other members prior to the next meeting. These notes, or "homework", are too voluminous to reproduce here, however, the information was invaluable in the development of this final report and in forming the thoughts of each group member. The members themselves represented one of the most professional groups the Chairman has ever been associated with, and hopefully this final report reflects this fact. If not, the problem was the sheer complexity and difficulty associated with the working group task and not the quality and enthusiasm of the members themselves.

II. TERMINOLOGY

2.1 INTRODUCTION

The following general terminology was adopted by the working group. This terminology is a somewhat modified version of the list that appear in the March 1979 issue of SIMULATION:

To provide a proper framework to review the credibility of a simulation, it is convenient to divide the simulation environment into three basic elements as depicted in Figure 2-1. The inner arrows describe the processes which relate the elements to each other and the outer arrows refer to the procedures which evaluate the credibility of these processes. This basic pictorial concept can be further refined as shown in Figure 2-2:

Each of the basic elements and their interrelationships are dealt with in the following set of definitions:

2.2 DESCRIPTION OF TERMINOLOGY

MISSILE SIMULATION	All analytical, digital computer, hybrid computer, or hardware-in-the-loop dynamic evaluations of tactical missiles to gain insight about reality
MODEL DOCUMENTATION	Systematic and coherent written and graphical representation of the associated data base according to a specified format and with a specific purpose
REALITY	An entity, situation, or system which has been selected for analysis
CONCEPTUAL MODEL	Model builder's perception and understanding of the system to be simulated. It consists of a hypothetical complete explanation of how the system functions. It often takes the form of verbal description, complemented by block diagrams, flow charts and systems specifications. In most cases, the large complexity of the conceptual model precludes its consideration as a possible simulation model. In view of the requirements of the intended simulation studies, the modeler establishes the complexity of the simulation model and degree of detail necessary. This information, generally in descriptive (verbal) form, complemented by block diagrams

	and flow charts constitutes the Conceptual Simulation Model or abbreviated to Conceptual Model. At the same time, this represents the requirements on the formal simulation model The formal model provides the technical description of the simulation model. It takes the form of mathematical equations, adequate description of logic flow and model data, complemented by the necessary detailed text. Both the conceptual and the formal model together form the theoretical model
DOMAIN OF INTENDED APPLICATION (OF CONCEPTUAL MODEL)	Prescribed conditions for which the CONCEPTUAL MODEL is intended to match REALITY
LEVEL OF AGREEMENT (OF CONCEPTUAL MODEL)	Expected agreement between the CONCEPTUAL MODEL and REALITY, consistent with the DOMAIN OF INTENDED APPLICATION and the purpose for which the model was built
MODEL QUALIFICATION	Determination of adequacy of the CONCEPTUAL MODEL to provide an acceptable LEVEL OF AGREEMENT for the DOMAIN OF INTENDED APPLICATION This may involve a comparison of alternate methods of missile simulation to establish credibility with two or more independent data sets or results
COMPUTERIZED MODEL	An operational computer program which implements a CONCEPTUAL MODEL
MODEL VERIFICATION	The process of showing that the proposed conceptual and the associated formal model are an adequate and consistent representation of the system to be simulated, all in view of the intended application. The method used is basically expert critique, which makes use of expertise and past experience in order to assess the adequacy of the conceptual model and the derivation of the formal model. Suitable documentation should allow

	following and understanding the ideas of the model builder in deriving the theoretical model
DOMAIN OF APPLICABILITY (OF COMPUTERIZED MODEL)	Prescribed conditions for which the COMPUTERIZED MODEL has been tested, compared against REALITY to the extent possible, and judged suitable for use (by MODEL VALIDATION, as described below)
RANGE OF ACCURACY (OF COMPUTERIZED MODEL)	Demonstrated agreement between the COMPUTERIZED MODEL and REALITY within a stipulated DOMAIN OF APPLICABILITY
IMPLEMENTATION	The process of programming the formal model on an adequate computer. It is recommended to apply software engineering methods such as top-down design, structured programming, top-down implementation and testing, etc.
PROGRAM VERIFICATION	The process of demonstrating that the formal model has been correctly implemented on the computer. This includes source code inspections, code walk throughs and tests of the model behavior predicted on the basis of the theoretical model (analytical solutions, behavior for small signals, etc.)
MODEL VALIDATION	The process of demonstrating through objective testing that the theoretical model and its implementation form an adequate representation of the system to be simulated, judged in view of the intended application. Model generated output data are being compared against actual data obtained by experiments performed on the real system
ASSESSMENT	The process of applying subjective judgment (i.e., expert opinion) to arrive at conclusions concerning the adequacy of the missile design, hardware, simulations (hardware-in-the-loop and mathematical), and testing

CERTIFICATION DOCUMENTATION	Documentation to communicate information concerning a model's credibility and applicability, containing, as a minimum, the following basic elements: (1) Statement of purpose for which the model has been built (2) Verbal and/or analytical description of the CONCEPTUAL MODEL and COMPUTERIZED MODEL (3) Specification of the DOMAIN OF APPLICABILITY and RANGE OF ACCURACY related to the purpose for which the model is intended (4) Description of tests used for MODEL VERIFICATION AND MODEL VALIDATION and a discussion of their adequacy
MODEL CERTIFICATION	Acceptance by the model user of the CERTIFICATION DOCUMENTATION as adequate evidence that the COMPUTERIZED MODEL can be effectively utilized for a specific application

III. CONFIDENCE LEVEL IN MODEL BEHAVIOR (CLIMB)

3.1 INTRODUCTION

CLIMB (Confidence Levels In Model Behavior) is a five level hierarchical process for representing information about a model such that a third party user can readily develop confidence in the model's behavior. The casual observer of the format for CLIMB process may conclude that this is a special but comprehensive documentation procedure. However, a closer look by the more than casual observer interested in simulation model development will recognize a hierarchical data representation structure for achieving a specific purpose. A closer examination of CLIMB reveals that a type of knowledge base is specified. This knowledge base serves as a guide for producing information and simulation model generated data required to develop confidence in the model's ability to achieve the intended purpose.

The term "knowledge base" as used here includes two components: facts about the domain of intended application and heuristics or rules of thumb for solving problems in the domain of interest. An example of the facts associated with a particular model might include the differential equations describing the dynamics of the modeled process. Also, textbooks and journals provide widely shared facts generally accepted by simulation modelers. Heuristics, on the other hand, are "educated guesses" or rules of good practice acquired by the experienced modeler over years of experience in developing and implementing simulation models. These educated guesses are what the experienced model developer uses as a guide in making many decisions during the model development process. Due to oversight and many other reasons, seldom if ever are the heuristics of the modeler reported for a particular model development. This results in an inadequate knowledge base for establishing model credibility. All experienced or expert model developers have their own "rules of thumb" for solving particular simulation design and modeling problems. Given several equal viable choices that may be available to the modeler, his rules of thumb will influence a particular choice in establishing the basic structure of the model. Assumptions made in developing and implementing the model reflect the heuristics of the modeler, as well as assumptions made about the domain of intended application. Establishing credibility with second and third party users of a model requires that more than just the facts about a model be reported. In addition, relevant heuristics used by the modeler must also be captured, i.e., a knowledge base must be available and in a usable form.

A major element in developing model credibility is the assumptions made by expert modelers related to model verification and validation. A missile model that is validated by going from an analytical model to flight test data does not have the same level of credibility as a model that was validated using intermediate stages of sub-system testing and model validation combined with hardware-in-the-loop operation. This assumption appears to be made about the hardware-in-the-loop option, even if the number of real world flight tests are less than the flights with the analytical model option. If this is a rule of thumb of a modeler, this will more than likely influence the structure of the system and subsystem models and the data recorded from the validation tests.

The CLIMB process, as described here, can be viewed as a process for capturing knowledge about a model sufficient to develop levels of confidence in the behavior of the model in question for a particular domain of application. In addition, CLIMB functions as a practical hierarchical knowledge structure for identifying model operation using at least three components: (1) documentation format, (2) knowledge base requirements and (3) a guide for simulation model development for generating results conducive to increasing confidence in model performance. When good model development practices are used, the CLIMB process captures that portion of the knowledge base available during model development and validation efforts. Using this approach, CLIMB does not restrict the choices of the model developer, but specifies what knowledge should be recorded once choices are made. This includes all stages of model development and validation typically associated with missile system development. These stages include: (1) the analytical model, (2) subsystem testing and model validation, (3) hardware-in-the-loop operation and (4) testing of the real world system with model updates and validation.

3.2 BACKGROUND ON THE DEVELOPMENT OF CLIMB

Results from a study of 24 major simulation facilities in five member nations in the NATO community (Ref 1) established the need for a procedure that would serve as a guide for developing confidence in simulation models. Interviews with facility managers, model developers, and simulationist during on-site visits identified at least three elements that must be included in any effective procedure for developing confidence in simulation model behavior. The elements are identified as: (1) documentation guide and format, (2) a structure for representing domain specific knowledges and, (3) a guide for generating simulation model data bases appropriate for building confidence in the model's behavior.

Documentation determines the life, death, duration, and quality of a model's existence. Results from the referenced study revealed that the largest and most active simulation facilities had little or no guidelines for documenting the simulation model development and validation efforts. In some instances, military specifications were used to meet documentation requirements. For the most part, model developers were left to choose what was to be documented and how it was to be recorded.

Reviewing model documentation reports generated from different departments within the same organization did not instill confidence in completeness or usability of the contents. The general view of documentation as derived from this study can be depicted as shown in Figure 3.1. Bits and pieces of data are acquired during and after the model development effort and put together as a documented data base for the model. The data, frequently with missing critical information, is assembled through a process that results in a multi-volume set of documents. All too frequently, this dazzling array of paper, charts, and graphs lend little or no motivation to potential users to take a serious look at the model. A conclusion from this observation is that documentation should be an integral part of the model development process.

Model validation techniques were not used in any systematic fashion in a vast majority of the facilities surveyed. In a few facilities, the approach was to attack the model at all levels using numerous methods and techniques to correlate the simulation results. Other validation efforts attempted to apply specific techniques to particular parts of the model. The spectrum of validation efforts can be viewed as using the random tool box approach in accomplishing model validation as depicted in Figure 3.2. Here, validation is mostly considered an operation that takes place after the model is designed and developed using whatever tools that are readily available.

Results from interviews conducted during this study indicate that in practice, a hierarchy of validation methods are used for confidence building in simulation models. The methods most frequently used are (1) expert opinions, (2) data plots with overlays, and (3) charts and graphs with mathematical analysis techniques. The vast majority of responses to the question "What procedure do you use to validate your simulation models"? was to execute and change the model until there was a "Good Feeling" about the model. How is this "Good Feeling" communicated to a second or third party? The typical response was that the interested party must exercise and change the model and compare data with various sources, just as the developer did to understand and gain insight into the model behavior.

A second conclusion from this study is that expert opinions, data plots, and mathematical analyses are important methodologies, but are not the main issues to be addressed in developing confidence in simulation models. These results indicate that the basic issues are the nature of the model and real world data bases. The model either represents a theoretical system or a real world system. The real world system is associated with laboratory test data, hardware-in-the-loop operation and real world data. If a theoretical model produces data for which no real world system exists, then only a limited confidence level could be established in the model's behavior. However, if the model produced data that could be compared with real world systems or other validated models, an increased level of confidence in model behavior is established. Additionally, if model generated data compares with results produced from a hardware-in-the-loop simulation, then another increase in level of confidence can be established. This process continues until multiple sets and sources of real world system data are compared with model generated data, producing an ever higher level of confidence in model behavior. The process of relating confidence levels and data bases is graphically depicted in Figure 3.3.

All the validation methods and mathematical tools can be used to perform data analysis at each CLIMB level: data plots, overlays, Monte Carlo analysis, chi-squared and Kolmogorov-Smirnov test, hypothesis testing, comparison of means and variances, etc. However, the confidence level in model behavior will not increase beyond a level determined by the nature of the data available to compare with the model generated data. In order to increase credibility, there must be a source of data at a progressively closer and closer level to the real world system. The elaborate use of expert opinions, data plots, and mathematical analyses only enhance the "Good Feeling" about the model performance at a particular confidence level, but does not provide a basis for increasing the confidence level in a model's ability to reflect the real world. This observation is graphically represented by the staircase levels of confidence and associated data base shown in Figure 3.3.

3.3 APPLICATION OF THE CLIMB PROCESS

The conceptual structure of the CLIMB process was developed during the first meeting of the working group. Details of the five level structure evolved through repeated review and application of the concept to existing simulation models. Results from individual modeling efforts were used to develop the details at each level of CLIMB (Appendix A). This provided both a broader view of the application and greater depth in defining the knowledge base for the hierarchical levels. An existing documented simulation model was recast in the CLIMB format through level three (Appendix B). This example provided the opportunity to identify significant operational consideration in the application and use of the CLIMB process with existing models.

The application of CLIMB to these existing models demonstrated vividly to the members of this working group the effectiveness of the CLIMB process, revealing in several instances insufficient cata in the model generated data base and the lack of information regarding model operation. Conclusions can be drawn about the use of CLIMB in two areas. One, the most efficient and effective uses of CLIMB are in the areas of establishing a basic framework for new model development and validation efforts. The knowledge required for desired confidence levels will be available during development with straight forward documentation resulting. Two, investing the manpower and computer resources for the application of CLIMB to developed or existing simulation models will be most effective in areas where: (a) the model will be used and may be updated by third parties not involved in the development, e.g., international transfer of models and (b) the model will be used over extended periods where the developer would not be available to establish confidence in the model behavior, i.e., models of operational weapons systems that require modeling and analysis support from different groups over the life of the system.

2.4 SUMMARY OF THE CLIMB PROCESS

The CLIMB (Confidence Levels in Model Behavior) process is a hierarchical process for developing confidence in simulation models through the integrated use of documentation, identification of appropriate knowledge bases for a given level of confidence, and a progressive use of analysis tools to broaden the confidence in model performance. A summary of the essential information for each identified CLIMB level is as follows:

CLIMB LEVEL 1: Model Summary, Results and Conclusion. This level includes information on the objective of the simulation, model developer, function of the model, domain of application, major assumption made in model development, criteria for model validation and the results of model application. At this level, only functional diagrams with major subsystems and critical variables are identified. The overview nature of the information here is intended to give the potential model user sufficient information to take the first step in reviewing the model capability without getting lost in details. Expert opinion is typically the major tool for confidence building at this level with descriptive rather than technical documentation.

CLIMB LEVEL 2: System Models and Submodels Theoretical and Indirect Data Base. The data base source is from theoretical models or existing validated models. Method of data comparison is included along with technical and descriptive documentation of submodels. A benchmark scenario with model results are given along with verification procedures of any computer implementation.

CLIMB LEVEL 3: Subsystem Real World Data Base. This level includes real world data for at least one major subsystem to be compared with the simulation model generated data. Documentation is provided for the total complex model including benchmark results. Data collection and validation methods are described.

CLIMB LEVEL 4: Hardware-In-The-Loop Operation Data. This data base includes results from a Hardware-In-The-Loop (HWIL) simulation with major subsystem models being replaced with actual hardware operation. Methods of data base comparisons are identified along with criteria for subsystem model validation. Specifics on the computer configuration for HWIL operation is also given.

CLIMB LEVEL 5: Total Real World Systems Operations. A data base is available from the real world system test and operation. As a minimum, the critical variables are compared with corresponding system data. Results of validation of system variables are given along with methods of validation according to established validation criteria.

A detailed outline of the elements in the total CLIMB process is reported in Appendix A.

IV. COMPUTER LANGUAGES

4.1 ADA-BASED SIMULATION LANGUAGES

The fact that ADA has been declared to be the standard for embedded software in the future has some impact on dynamic system simulation languages. The simulator, besides being used as a tool for system development, is also used as a system integration facility that includes the integration of embedded software. This second domain of application has a growing importance. Since the operational software to be embedded has to be written in ADA, there is a strong need for the integration facility software also to be written in ADA because there will be no way to interface ADA to any other language. Such interfacing would be in direct contradiction to ADA's design target of software reliability. This means that today's simulation languages will no longer be applicable for the purpose of embedded software integration. Thus, a new generation of simulation languages will have to be introduced even though it may not really be possible to do this in the given time span. One possibility to overcome the time problem is to stay with the existing simulation languages and to modify the compiler to transform from the native simulation language code into an ADA source code program that will operate in a preprogrammed ADA simulation system. This would be equivalent to the existing languages that use FORTRAN as an intermediate language.

In order to start as soon as possible with new software, PASCAL could be used until ADA becomes available. Besides the aspect of embedded software integration, ADA can be helpful in identifying modern software techniques that should be included in future simulation languages. Examples of such broadly accepted common features would be:

- GOTO--free structured programming techniques

- modularization of programs into complete, self-contained sub-modules

- mandatory declaration of variables, etc.

4.2 SIMULATION SYSTEM ENVIRONMENT

There is a great number of problems related to simulation and projects making use of simulation tools. Some of these problems that are normally not mentioned when dealing with simulation languages include:

(1) Life cycle maintenance costs for weapon performance simulation programs

(2) Ease of software maintenance during development

Item (2) is very important and needs to be discussed further.

During a project involving a great number of engineers, it must never happen that a copy of a "reference standard program" is made which is not made from a certified source program and not clearly identified and registered.

Any change to the reference standard program has to be immediately duplicated on all existing copies of that reference program so that, during development, this requirement cannot be fulfilled without activating a considerable amount of manpower. Therefore, one design objective for future simulation languages should be to offer tools for facilitating such "software consistency management" work.

The software management problem automatically leads to the problem of describing the environment of the program. It is not sufficient to define only the program itself and the language used for the compilation, but all environmental conditions have to be described such as computer type and model, operating systems, libraries, etc. It would be most desirable if future simulation languages be included as an integral part of a bigger simulation system environment description with system features like those mentioned software consistency management features being included. There are a great number of problems that are most common in today's simulation facilities and that could be avoided by a well designed environment and a language supporting that environment. Therefore, it seems to be reasonable to suggest further investigations in that area.

4.3 LANGUAGE FEATURES

A wish list of desirable language features has been compiled. This list is presented here without comments just for information:

- portability
- improved error debugging, error avoidance, automatic detection of errors (computer/language/compiler problem)
- output/plot capability, stripplot, plot overlay, specific plot for pin-pointing e.g., showing jumps of the discrete controllers.
- non-ideal behavior of digital controllers
  . synchronization, time delay, time matching ("discrete time event")
  . fixed point arithmetic, scaling, overflow, ADC, DAC
- simulation of noise
- statistical runs, Monte Carlo runs, statistical evaluation, connection to statistical programs
- connection to control analysis plus synthesis programs for verification, analysis, synthesis, optimization "ANALYZ"
- integration algorithms plus step sizes
  multiple integration algorithms plus step
  sizes interpolation, extrapolation

- configuration control
- simulation of specific points such as friction ("discrete state event")
- real time capability plus hardware-in-the-loop capability
- modularity (i.e., easy decomposition of models into sub-units)
- structured programming techniques
- automatized "software consistency control" administering the different releases and versions of library programs
- decomposition of a simulation "program" into "model" and driving environment "experiment"
- data base concept for result data and post-processor programs

V. SOFTWARE/VERIFICATION/VALIDATION/ASSESSMENT METHODS

5.1 INTRODUCTION

Following much discussion/debate by members of this working group, a general consensus was reached that a detailed treatment of specific verification and validation techniques, their merits, drawbacks, etc., as applied to guided missile simulation development would far exceed the scope and time limitations imposed on WG-12 and, therefore, not be attempted. It was decided instead to concentrate the group's primary efforts on outlining the CLIMB process previously discussed. Although the primary efforts of this working group were not directed at exploring the merits of verification and validation techniques, for completeness an attempt is made in this section to identify many of the popular approaches with a specific effort made to identify references so the reader can explore in more detail particular approaches of interest.

Consistent with the terminology established in Section II, Figure 5.1 illustrates graphically the interrelationships between verification, validation, and assessment. As can be seen in the figure, all key elements of verification stem from some form of mathematical or computer code checks performed on the simulation to verify that the answers produced by the simulation are consistent with theory. One key answer is that to the question, "Is the simulation code error free and does it produce answers consistent with the math model?" It can also be seen from the figure that a rather large menu of verification options and techniques are commonly used. With the increasing use of highly structured and sometimes automated simulation languages, techniques as these are gradually being incorporated directly into the languages, thus making the verification process easier and more straightforward to accomplish. Each of the verification techniques illustrated in the figure are discussed in Paragraph 5.2 below.

Validation, on the other hand, requires real world data to compare against simulation results; not theoretical predictions as is the case for verification. As shown in Figure 5.1, some form of hardware testing must be conducted to obtain the necessary real world data. These tests generally consist of bench test, hardware-in-the-loop (HIL) simulation, and free-flight trials. Unlike verification, which as discussed above is beginning to take on a rather structured straightforward approach and can very closely approach an absolute, validation is still widely varying and subject to individual preferences and technique. To accomplish model and simulation validation (usally accomplished simultaneously), some form of objective mathematical method must be applied to compare available real world data to simulation predictions. Additionally, the real world data base used for validation must be independent from that used in the original model development. A good example of this is the validation of a missile aerodynamic model using data obtained from actual vehicle flight testing when the original model had been developed from wind tunnel data or aerodynamic prediction computer programs. Several mathematical methods being used by members of this working group for validation and others reported in open literature are discussed in Paragraph 5.3 below.

The last topic to be discussed in this section of the report is assessment. Assessment, as shown in Figure 5.1, plays a significant role in the missile overall design, development, and testing process. Two types of assessment are identified in the figure; one, in-house and two, third party. Although each type involves subjective judgment, consistent with the terminology in Section II, one feeds information back into the missile design process while the other feeds information to the customer's management decision process. The general assessment problem along with details of both assessment types are discussed further in Paragraph 5.4 below.

5.2 VERIFICATION/VALIDATION METHODS

5.2.1 Systematic Program Testing: (A difficult task.)

One of the largest illusions in software development is the belief that it is possible to get by with little program testing and verification (Ref 2). According to studies in the USA (Ref 3), the verification requires 30% - 50% of the total project costs (Fig 5.2). The higher the portion, the more complex the system. No methods, techniques and tools are available to generate programs without errors.

It is said that the originators of programming, among others, John von Neumann, have been totally surprised by the error-rate of their programs (Ref 4). Methods for a mathematical proof that the program is faultless have not been fully developed up to now. So the testing is the only means for minimizing the error-rate (Ref 5). The testing begins with the functional specification, continues with the verification of the different modules and ends with the integration test (Figs 5.3 and 5.4).

There exist some general rules, which can help to minimize the errors:

- Well defined specification and program design, a farsighted program structure and terminology

- Careful coding

- Design of the program with regard to testing; Implementation of testing aids already in coding (Ref 6)

- Advanced computer tools for formal testing computers, languages and compilers with improved error avoidance

- Selection of adequate design and test strategies (Fig 5.5):

  . "Top-down". The modules are substituted by program "stubs." If the overhead program has been tested, the stubs are replaced by detailed modules (Ref 7)

  . "Bottom-up". The design and testing begins with the lower modules. The upper programs or calling programs, are substituted by test drivers that are later replaced by real programs (Ref 8)

"Hardest-First". The programming and testing begins in the middle with difficult modules. The program grows both up and down (Refs 9 and 10).

Each of these strategies has its advantages and disadvantages and there is no clear recommendation in the software engineering literature as to which of these methods is the best.

There are two approaches for the determination of test conditions: the "black box" and the "white box" approaches. The "black box" approach considers the module to be tested as black box between the input and output without being interested in the inner details of this box or module (Ref 11). The "white box" approach, on the other hand, studies the details of the module or test object in order to derive the test conditions. Thus, the amount of test cases can be minimized. However, the test object has to be understood very well by the tester. A tester, who is not familiar with the details of the program, should begin with the "black box" approach (Ref 12).

It is recommended that a test laboratory book be kept in which all test conditions and results are documented. Fig 5.6 and 5.7 show typical test systems. Fig 5.8 shows the "dual program analysis" is that used at NASA.

Management and costs of testing and quality assurance (QA). The test management has to take into account: (a) costs of testing (Table 5.1), (b) planning of testing, (c) organization of testing, (d) specification of testing, and (e) revision of testing. The testing of a program consists of the following test phases:

- Static formal tests - The program is compiled but not yet run.

- Dynamic formal tests - First runs with debugging aids.

- Test of the modules - Module verification.

- Integration test - Verification of the overall system.

- Final acceptance test (Ref 13).

The first two formal tests rely heavily on the available computer and its language. Typical methods available in CYBER-Fortran V and ACSL are shown in Tables 5.2 and 5.3 and Appendix C.

- The earlier the quality assurance starts, the cheaper it is in proportion to development costs of the program; i.e.,
  - 0.093 if started at program specification
  - 0.125 if started at debugging
- The more favorable the test conditions, the cheaper the quality assurance becomes:
  - 0.125 with favorable conditions
  - 0.420 with unfavorable conditions
- The larger the system (> 64 k), the better the relation of quality assurance costs to development costs become:

  Favorable conditions:

32% for small systems	(< 32 K)
12% for large systems	(> 64 K)

  Unfavorable conditions

69% for small systems	(< 32 K)
42% for large systems	(> 64 K)

Table 5.1: Costs of Testing and Quality Assurance Derived from IEEE Studies.

Static formal tests

- source code inspection
- inspection of the Fortran-list precompiled by the simulation language
- inspection of the cross reference listing
- inspection of the load list
- inspection of the program portability (Compilation of the program in the "ANSI-mode (Fortran) to ascertain the statements flagged as "non-ANSI").

Dynamic formal tests

- Runs with "no preset = zero". (Some computers do not have this option by which the error rate is increased)

Runs with "DEBUGGING AIDS"

- ACSL DEBUG
- Cyber Trace back (Post mortem dump, which prints a readable summary of the error condition and the state of the program at the time of failure in terms of the names used in the original program.)
- Cyber Interactive Debug facility (CID).

Table 5.2: Computer Tools for Formal Testing

TABLE 5.3: Recommendations of MBB/UA for CRL-Inspection of Fortran-Programs

Inspection of Cross Reference List (CRL)

Requirements	Verification
- Variables have to be explicitly declared by type declaration statements	Line numbers of first reference (left column in CRL of references) have to be less than line number of first statement after type declarations
- Variables have to be defined before first usage	Line number of first definition has to be less than first (nondeclaration) reference
- No unused variables in the program	0 There is no variable flagged "UNUSED" 0 There is no variable which is defined but referred to only once (type declaration)
- Input, transfer and control variables (formal parameters) must not be redefined in the program	Variable marked F.P., which intentionally are input variables, are defined only in line No 1
- Output and error indication variables have to be defined in the program	Variables marked F.P., which intentionally are output variables, have to be defined in line No 1 plus at least one more line
- Variables contained in DATA-statements must not be redefined in the program	Variables defined in DATA-statements are defined only once (i.e., in the DATA-statement)

The further tests are called verification tests. A typical test consists of the following phases:

- Recognize an error
- Make a hypothesis about the location where the error will occur and the source of error
- check the hypothesis by the available computer tools and detailed "test-print-outs"
- correct the error.

The first phase, recognizing an error, is the essential and most difficult point of the verification.

We define an error as the difference between exact solutions and simulated solutions. The main difficulty with simulation programs is that the exact solution is only known for simple modules under certain restrictions. With complicated models, the exact solution may only be approximated by numerical simulation techniques. In the next section, methods are described for verifying and checking these approximations.

5.2.2 General Verification Methods for Missile Simulations

There exists an extensive number of publications on the theme "verification of software", but nothing specific has been published with respect to "verification of missile simulations". The only exception is the paper of S. Schlesinger from the Aerospace Corporation, El Segundo, California (Ref 14). With the example of the simulation of an analog autopilot, typical verification tests are described.

In the following, we apply the verification methods to the simulation of a more modern missile, which uses microprocessors instead of the former analog controllers. It turns out that simulating digital control systems, which are also called "sampled-data control systems", is a great challenge from the point of simulation and theory. The following is intended to remain not only general, but to illustrate some problems with examples resulting from experiences with a large 6 - DOF simulation written in ACSL and Fortran V. The verification methods that have been applied include:

- Verification of the equations of the model
- Hand check
- Verification against existing and relevant theory
- Verification against other simulation programs
- Degeneracy tests
  When parameters are selected to eliminate the effect of a particular feature in a model, then the resulting output from

the computer simulation shall act as if the characteristic modelled by the eliminated feature is, in fact, totally absent

- Consistency test
  Similar simulation cases yield essentially similar results even though stated to the computer model with differing combinations of descriptive parameters.

- Verification of integration algorithms, stepsizes, sorting and timing. Sorting and timing errors may introduce additional time delays to the overall system

- Logic tests - branch/path tests

- Integration tests
  The purpose of integration tests is to verify the interface between (verified) modules from a static and dynamic point of view.

- Stochastic test

5.2.2.1 Implementation of testing aids already in coding

These testing aids are recommended:

- Switches for degeneracy tests. For example a rate gyro may be simulated

```
GYSW = 0.  no errors, no time lag
     = 1.  errors, no time lag
     = 2.  errors, time lag
```

- Switches for consistency tests. A typical application may be switches for verifying the subsystems "Autopilot + Airframe" in pitch against the yaw channel. The test conditions must be such, that according to the theory, the pitch and yaw channel must yield identical results.

- Switches and test driving signals for opening control loops. A typical example is that of the most inner loop; it is first verified by means of a deterministic signal (step, ramp, sinusodial signal). After this test, the next upper control loop is closed.

- Specific "test-print-outs" for logic-, branch-, path- and timing- testing.

We distinguish:

- Deterministic verification: The data of the plant are fixed and the test driving signals are deterministic.

- Stochastic verification: A stochastic signal is used as test driver. Also, the plant data may statistically vary according to the assumed tolerances (3 σ values).

A general rule says, that a simulation must first be verified by means of deterministic methods.

5.2.3 Specific Verification Methods for Missile Simulations

5.2.3.1 Verification of the equations of the model

Several equations are needed to represent the image of a spatial target as seen by the missile seeker head. For verifying these equations, one must know the terminology and derivation of the equations, which should be documented and so detailed that they can be verified by another person.

5.2.3.2 Hand checks

Examples are:

- Verifications of the correct implementation of the aerodynamics.
- Verifications of the integration algorithms by a sinusodial input signal which is integrated twice. The result should again be a sinusodial signal.
- Graphical verification of the image processing of the seeker seeing a spatial target.

5.2.3.3 Verification against existing and relevant theory

Guidance and control theory has become a well established topic with a history of roughly 50 years and an extensive number of publications. However, there exists a large gap between the fairly sophisticated theory and the simple PID-controllers and PN-navigation, which are predominantly used in practice. One of the reasons for this gap is the fact that modern control theory is quite difficult to apply. The theory of sampled data control systems is especially much more difficult to apply than the conventional theory for continuous systems. Fig 5.9 illustrates this. Calculating the exact time response, x, to the deterministic inputs, x_c, which may be a step, a ramp or a sinusodial signal, requires considerable skill in z-transform analysis (Ref 15) and a substantial number of sophisticated numerical calculations even though only a simple digital PD-controller has to be analyzed. Also, random inputs, such as the measurement noise in Fig 5.9 may be taken into account. For example, there exists a theory for calculating the variance of the output, x, due to the noise output, n(t).

Special computer programs for control analysis and design exist. An example is the Computer Aided Control System Design (CADSD) from ETH Zentrum, Institute for Automatic Control, Zurich, Switzerland (Ref 16) which consists of:

- SIMNON - A simulation program for a continuous system with discrete-time regulators (sampled data systems)

- IDPACK - A program system for data analysis and identification of linear deterministic stochastic multi-input, single-output systems

- SYNPAC - A state-space oriented control systems design program

- POPAC - A frequency-domain oriented design program, and

- MODPAC - A program for transformations between different control system representations

Another method exists for analying the digital control by means of approximations. This method has the benefit that a special control analysis program is not necessary and hand checks may be used, thus providing more insight into potential problems. For example, the digital controller (Fig 5.9) can be converted into an analog controller by use of a bilinear transformation and the sampler + zero order hold may be approximated by a first order Pade time lag ($\tau = T/2$) or a time delay(e^{-s} T/2). Thus the conventional methods in the s-plane may be used. Blakelock (Ref 17) calls the first approximation "digitization" method and compares it with the exact z-transform method.

There are many other theories available, which may be used for verification of simulations:

- Nonliear Control Theory

- Kalman Filter Theory

- Guidance Theory
  Conventional and modern guidance theory with deterministic and stochastic inputs. A typical example is to verify the miss distance due to noise of the seeker.

- Strapdown Algorithms Theory

5.2.3.4 Verification against other simulation programs

Examples of this method of verification include:

- 6 DOF against 3 DOF
- 3 DOF against 3D (3-dimensional with trimmed aerodynamics)
- 6 DOF (CSMP) against 6 DOF (ACSL) (Verification of the conversion from one simulation language/computer to another simulation language/computer.)

5.2.3.5 Degeneracy test

Degeneracy implies simplifying the phenomena being modeled by selecting certain special parameter values. An example follows:

For the stabilization of the seeker platform, the following degeneracy test has been conducted:

- Simulate the gyros and look-angle pick offs without errors and time lag

- Let the two-axis stabilization degenerate to a pitch-only-stabilization.

Under these simplified conditions, the stabilization should be perfect. In this manner, errors in the program can be detected much easier.

5.2.3.6 Consistency tests

See Paragraph 5.2.2.1 under the testing aid titled "Switches for consistency tests."

5.2.3.7 Verification of integration algorithms and stepsizes, sorting, timing and repeatability

Similar to guidance and control theory, the theory about integration algorithms has become a well established topic with a long history. However, there exists a large gap of knowledge between the numerical mathematician and the practical simulationist. Furthermore, it turns out that algorithms that had been extremely powerful for the simulation of continuous systems, such as the Adams multistep methods, are no longer optimal for problems having frequent discontinuities (Ref 18). Modern algorithms, which are commonly called Adams PECE (Predict, Evaluate, Correct, Evaluate) automatically vary the stepsize and order for solving a problem to a given requested accuracy. W. Bub shows in Ref 19 that it is possible to evaluate the numerical stability and other features of integration algorithms by means of the z-transform * and recommends Adams multistep methods for the solution of linear transfer functions. The other class of integration algorithms are called Runge-Kutta methods. They are most widely used for the numerical solution of first-order, first-degree equations. In contrast to the Adams multistep algorithms, they are self-starting.

"Runge-Kutta" methods are easy to implement on a digital computer and are probably preferable to predictor-corrector techniques for most purposes. Their main disadvantage is that it is difficult to keep a check on the truncation error. The simplest way to check a solution is to repeat it with a halved step length - though more efficient means are suggested in specialist texts. A further point is that the widely used fourth-order Runge-Kutta method requires four evaluations per step of the function "f", compared with at most two per step of a predictor-corrector solution of comparable accuracy. The latter method may, therefore, be more suitable in cases where it is very complicated, so that its evaluation is expensive in computing time. (Ref 20)

---

* Numerical mathematicians use other methods.

After this general literature overview, we shall deal with the problem of simulating a fast digitally controlled missile, whose smallest sampling time, T, must be very small due to the required high bandwidth of seeker head stabilization and autopilot. The main issues are:

(a) Optimal integration algorithm and stepsize (assumption: the program uses a unique integration algorithm and stepsize)

(b) Verification of the sorting and timing

(c) Simulation of the non-ideal behavior of the digital controllers

(d) Multiple integration algorithms and stepsizes to save execution time.

(a) Choice and verification of the integration algorithms and stepsizes

A general rule of thumb is that the stepsize, h, should be at least $h = 1/2\ T_{min}$, with T_{min} = minimum of the smallest sampling interval T, smallest time constant τ, smallest $\tau_n = 1/\omega_n$, smallest $\tau_{\sim} = 1/\omega_{\sim}$. The last 2 expressions refer to the bandwidth of the noise, respectively the frequency of the sinusodial signal (see Fig 5.9). In this case, the sampling time was the main driver for the stepsize. The cheapest algorithm is the Adams-Bashforth 1 (AB 1), which requires only one evaluation per step of the derivative function "f" (see below). It has been compared with the Runge-Kutta 2, which requires two evaluations per step. The starting algorithm of the AB 1 has been executed only at the beginning of simulation and not as theoretically required at each discontinuity (i.e., sampling interval). Experience with a large program (30 integrators) was that h = 0.5 T was sufficient for RK 2, but not for AB 1, which requires in any case h = 0.25 T. With this stepsize, the AB 1 is still slightly inferior to the RK 2 with h = 0.5 T.

<u>Adams-Bashforth (AB 1)</u>

t = 0 $\quad x_o \rightarrow x_o$

t = h $\quad x_1 = x_o + h\ x_o \rightarrow x_1$

$\quad x_1 = x_o + h/2\ (x_o + x_1) \rightarrow x_1$ Starting algorithm

t = 2h $\quad x_2 = x_1 + h/2\ (3\ x_o - x_1)$

t = 3h $\quad x_i = x_{i-1} + h/2\ (3x_{i-1} - x_{i-2})$

This can be expressed as z-transfer function $x(z)/x(z) = \dfrac{h\ (3z - 1)}{2z\ (z - 1)}$

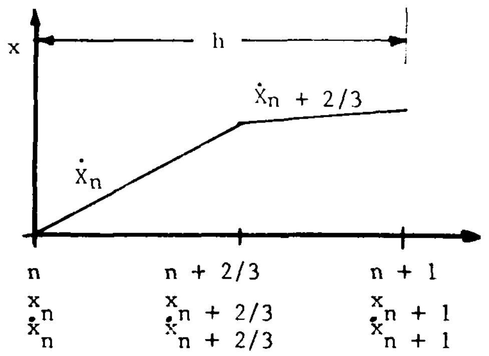
Runge-Kutta 2. order (RK 2)

$x_{n+1} = x_n + \alpha\, k_o + (1 - \alpha)\, k_1$

with $k_o = h\,\dot{x}_n$, $\dot{x}_n = \dot{x}\,(t = t_n,\ x = x_n)$

$k_1 = h\,\dot{x}\,(t = t_n + h/2\,(1 - \alpha),\ x = x_n + h\dot{x}_n\ 1/2\,(1 - \alpha)$

$\alpha = 2/3$



The chosen way of verifying the numerical solution was to repeat runs with a halved step length and to introduce a severe test signal such as a sinusodial perturbance. At the beginning, the difference was considerable. One could think that this was the proof that the smaller stepsize had to be taken. However, it turned out, that also sorting and timing errors contributed to the difference.

(b) Verification of the sorting, timing and sample devices

With simulation languages, there exists an automatic program sorting. ACSL automatically sorts the model definition code that is placed in the DERIVATIVE section. The statements within PROCEDURAL blocks are not sorted. The PROCEDURAL block is positioned according to the input and output variables of the PROCEDURAL header. Sorting errors result especially in that case where not all output and input variables are in the header. On the other hand, too many variables in the header may result in "program unsortable." Another problem is the simulation of digital algorithms. A large missile simulation contains multiple sampling devices and different time delays.

The following verification methods have been used:

- Runs with "no preset = zero" to detect initialization errors.

- Verification of the repeatability; 2 runs,' which are repeated in one JOB, must yield identical results.

- Runs with additional evaluation of the DERIVATIVE section at each communication interval (which is identical to the integration interval). The comparison with runs where this additional evaluation is omitted must yield identical results.

- detailed "test print outs" for the testing of the different sampling and delay devices.

(c) Verification of the simulation of the non-ideal behavior of the digital controllers

The non-ideal behavior consists of computation delays, asynchronous sampling and delays, errors due to fixed point arithmetic and overflow. Usually, these effects are assumed to be negligible. However, in applications with fast control systems, these effects may cause limit cycles, time matching problems and transient effects and therefore cannot be neglected (Ref 21). Modern simulation languages are improved with respect to these features. The ACSL version 8A now offers the macro SKEDTE (Schedule Time Event), which can be used to simulate a computation delay, ΔT_c, after the Analog-Digital-Conversion (ADC). Such a feature of the simulation language must be verified by means of detailed test print outs.

(d) Verification of multiple integration algorithms and stepsizes

The simulation of fast digital control systems requires small variable stepsizes and special integration algorithms. The slower motion of the missile may be simulated with larger stepsizes and multistep ADAMS algorithms. With ACSL, more than one DERIVATIVE section may be used, each with its own independent integration algorithm and stepsize. Although this technique can save execution time when correctly used, any implementation must be approached with caution since, in general, incorrect answers will be obtained unless the model is divided into blocks with a full understanding of the effects of computation delays for variables that cross block boundaries (Ref 22).

Languages are offered, which perform the necessary syntax analysis and partition the problem into a sequential and a parallel part. Special methods must be developed to verify these sophisticated simulations.

5.2.3.8 Logic tests - Branch/Patch tests

In designing an appropriate structure and logic of the overall program, the "top-down" design is recommended. The more sophisticated modules are substituted by program stubs. Special logic drivers and logic verification tests may be used.

5.2.3.9 Integration tests

The verification of the overall missile system must be based on verification tests generated by means of the subsystem simulations. Special testing aids (switches, test drivers and test-print-outs) are helpful in comparing the overall program against the subsystem program.

5.2.3.10 Stochastic verification

The aim of stochastic verification is to verify:

- the statistical behavior of the input noise(variance, bandwidth)

- the statistical behavior of the output noise against the theory (if possible) or other programs

- whether the number of statistical runs is sufficient

The following two examples illustrate this verification:

First, Fig 5.10 shows the "measured" mean squared value as a function of the number of runs N_s. The theoretical value is 100 m. It turns out, that N_s = 100 yields errors up to 10% and is to small.

Finally, Fig 5.11 shows the noise reduction of a typical Kalman-Filter with time-dependent gain. If 1000 runs are evaluated, the simulation results agree well with the theory.

What we can learn from these examples, is that a statistical verification is extremely expensive. On the other hand, if only a few statistical runs are performed, the statistical errors may be considerable.

5.3 VALIDATION METHODS

5.3.1 The General Validation Problem

Validation represents, perhaps, the biggest challenge facing today's simulationist. Reviewing the definition on validation presented in Section II, one can see the difficulty of the task. Validation tests the agreement between the model and the real world system being modeled. The two key words in this definition are tests and real world. The requirement to apply mathematical tests to compare the model and real world separates validation from assessment. This also makes validation an objective process, separating it from assessment - a subjective process. The definition of validation by implication requires the existence of a real world data base before validation comparisons are possible. This real world data base is generally constructed through four primary methods: (1) Bench tests; (2) HIL tests; (3) Field measurements (target signatures, etc.); and (4) Flight trials (captive and free-flight). It should also be noted that for model validation to be possible, the real world data base used for comparison must be independent from that used in the original model development process. During early stages of missile system development, model validation and system testing for assessment purposes become inseparable. Flight tests, once exclusively reserved for system performance demonstration, are now often aimed at providing real world data for model/simulation validation. Even with the increasing emphasis on the availability of independent data sources for validation, the process is not absolute. Simulation predicted system performance will never exactly match actual system performance under all conditions. Results being reported in current literature on missile simulation validation suggests that the proper question is not "Is the model valid"? but "How valid is the model"? Some measure of acceptable error between the model and real world must be established. It is precisely this problem that gives rise to numerous mathematical techniques to quantify the acceptable error for validation purposes. Several of the most popular validation techniques used in guided missile simulations are discussed below.

5.3.2 Specific Validation Methods

5.3.2.1 Pilot Overlays and Graphical Comparisons

Plot overlay and graphic techniques involve the comparison of real world data to that produced by the simulation using plots. It is by far the most popular validation technique. Graphical comparisons and cross plotting of variables is limited only by the imagination of the simulationist and the availability of real world data. The major difficulty with the technique is "How close is close enough"? Generally, "expert opinion" must decide this issue. Common practice using Overlay and Graphical validation techniques is to plot real world test data and establish a somewhat arbitrary, but small allowable error that the simulation must stay within to be considered valid. Expert judgment is usually the basis for establishing the allowable error. Acceptable error is generally established between one and ten percent depending on the particular state variables being considered and the intended application of the simulation. The major advantage of this technique, aside from its simplicity, is the automatic and very graphic validation audit trail developed for the simulation/model documentation package. For additional information and examples of this technique, consult (Ref 23).

5.3.2.2 Correlation Coefficients

The generation of correlation coefficients is a well known mathematical technique to measure the time correlation of two processes. If, for example, the time history of a missile system state variable or internal subsystem variable were recorded during a live flight test and the time history of the same variable generated by the simulation were processed to obtain a favorable correlation coefficient (approaching 1.0), the two processes (one real, the other simulated) are considered equal. Only one computer run and free-flight trial is necessary to conduct this test. Generally speaking, correlation coefficients measure the degree to which two time-varying signals compare. A perfect match with flight test data obviously represents a valid computation of a variable. The application of correlation mathematics represents a strenuous test for simulation validity. Not only must amplitude characteristics match to obtain a good correlation coefficient, but phase is also extremely important. Very small deviations in missile system models will create significant phase changes on some state variables, especially for non-linear models, resulting in rather dramatic shifts in correlation coefficients. For this reason, correlation coefficient techniques are not often used for validation. A detailed discussion of correlation mathematics and examples of their use may be found in (Refs 23, 24 and 25).

5.3.2.3 Theil's Inequality Coefficient

A technique developed by Theil has been used by economists to validate simulations that include econometric models. Theil's inequality coefficient, "U", provides an index that measures the degree to which a simulation model provides retrospective predictions of observed historical data. "U" varies between zero and one: if U=0, the predictions are perfect, if U=1, the predictions are very bad. Although Theil's theory was developed for economic models, in recent years much success has been demonstrated in its application to dynamic scientific models. Examples of its use for missile simulation validation are contained in (Refs 26, 27 and 28).

5.3.2.4 Chi-Square and Kologorov-Smirov Tests

The chi-square and Kologorov-Smirov tests are two special types of hypothesis tests often used to establish the equivalence of a probability density function of sampled data (in this case, simulation output) to some theoretical density function (in this case, real world data). Each of these tests derive a figure-of-merit to characterize the goodness-of-fit between two probability density functions. The chi-square test general procedure involves the use of a statistic with an appropriate chi-square distribution as a measure of the discrepancy between an observed probability density function and the theoretical density function. A hypothesis of equivalence is then tested by studying the distribution of this statistic. The main problem of the chi-square test is that it is relatively sensitive to non-normality. The Kologorov-Smirov test, on the other hand, is a distribution-free (nonparametric) test. It involves specifying the cumulative frequency distribution of the simulated and actual data. Unlike the chi-square test, the figure of merit or goodness-of-fit is not a statistical variable and is, therefore, not sensitive to normality. For more detail concerning exact implementation procedures for each of these tests see Refs 23, 25 and 29.

5.3.2.5 Monte Carlo Boundary Generation

Monte Carlo Boundary Generation is a well known technique involving multiple runs of a simulation, including all known noise and error sources, to establish accumulated statistical properties of selected state variables as a function of time. Using the overlay graphics methods discussed in Paragraph 5.3.2.1, the mean and standard deviation of the selected variables are plotted as a function of elapsed simulated missile flight time. If real world flight trial data for similar launch conditions overlays the Monte Carlo generated simulation data within the established bounds (usually one sigma), the simulation is considered valid for that variable. Generating similar plots for all critical state variables and internal system variables will fully validate the simulation. Two major differences distinguish the Monte Carlo Boundary Technique from the Plot Overlay and Graphical Comparison Technique as described in this report. One, the validation boundaries are applied to the simulated data for the Monte Carlo technique with flight test data overlayed within the defined boundaries to establish simulation validity, and two, the validation boundaries represent statistically generated error properties based on multiple simulation runs instead of a single flight test trial with small, but somewhat arbitrary error boundaries applied as is the case for the Plot Overlay and Graphical Validation Method. The major disadvantage of the Monte Carlo Validation Technique should be rather obvious by now; that is, computer costs. Hundreds of runs are sometimes required to obtain accurate statistical properties and to assure adequate confidence levels are obtained. The application of special programs for statistical analysis is also sometimes required.

5.3.2.6 Spectral Analysis

Spectral analysis is a class of mathematical processes that consider the spectral content of data. These include techniques as Power Spectral Density, Cross Spectral Density, and others. Spectral analysis provides a means of objectively comparing time series data generated by a computer simulation with an observed time series obtained from real world data collection. Spectral analysis is aimed at the quantification and evaluation of uncorrelated data after the data has been transformed into the frequency domain. By comparing the computed spectra of simulation output data and corresponding real world data, it can be inferred how well the simulation resembles the system or subsystem it is intended to emulate. Unlike many of the other validation techniques discussed in this report, spectral analysis does not inherently produce figures-of-merit to quantify goodness-of-fit between simulation output and real world data. Spectral analysis is extremely dependent on expert judgment to answer the question "How close is close enough"? For more discussion covering the many spectral analysis techniques and examples of their use, consult Refs 30, 31, 32 and 33.

5.4 <u>ASSESSMENT:</u>

5.4.1 The General Assessment Problem

Assessment, as defined and used in this report, includes all activities involving the application of subjective judgment (i.e., expert opinion) to answer the question "Will the system/subsystem design meet specifications"? Assessment involves subjective evaluations of all aspects of the weapon system development process, including, but not limited to: system simulations, hardware bench tests, captive flight tests, free-flight tests, hardware-in-the-loop tests, etc. Many of the mathematical processes and techniques applied to missile simulation validation are useful in acquiring data to help answer the assessment question. The key to understanding the difference between validation and assessment is in the use of the data. That is, validation concentrates on the performance of the simulation (i.e., Does the simulation properly emulate the design?), while assessment concentrates on the performance of the system begin simulated (i.e., Does the design meet specifications?) Assessment begins very early in the weapons development process and continues throughout the life of the project. As illustrated in Figure 5.1, there are two distinct types of assessment. One, in-house accomplished by the system prime contractor, and the other, third party accomplished by the customer or a third party contractor employed by the customer.

In-house assessment is identified in Figure 5.1 by the boxes representing four major data sources; the mathematical simulation, the HIL simulation, missile design data, and missile hardware data. The figure clearly illustrates that in-house assessment plays a key role in the system/subsystem design process. Although subjective in nature, in-house assessment offers the first opportunity to compare simulation predicted performance (generally generated during the simulation verification and validation process) to customer requirements (system design specifications) and feeds back information to the design process. This feedback often results in system design modifications which in turn result in simulation modifications creating the need for additional passes through the simulation verification and validation process. During system/subsystem design activities, in-house assessment and simulation verification and validation are inseparable, each feeding the other until a design is finalized.

Third party assessment, on the other hand, involves independent evaluation of the entire missile system/subsystem development and demonstration process, and as such, has little direct impact on the system design. Some indirect influence is present, however, due to the use of third party assessment data for customer management decisions and the impact these decisions may have on system specifications. As pointed out by Richard Richels of the Electric Power Research Institute, "Decision makers are becoming increasingly annoyed that different analyses get quite different answers to the same problem." When this happens, it is natural to want to take a closer look at the simulation models employed and find out why such differences result. This all to familiar situation has led to the increasing use of third party assessment to provide an independent check and balance on the weapons development process and to provide customer management (i.e., decision makers) with unbiased nonparochial information on system/subsystem designs and performance limitations.

VI SUMMARY AND RECOMMENDATIONS

In summary, the Working Group found that the conclusion reached in AGARD-AG-279, "Survey of Missile Simulation and Flight Mechanics Facilities in NATO" regarding the general lack of uniform accepted missile simulation validations methods was indeed correct. In addition, the need for such methods is becoming increasingly apparent in all of the participating Working Group countries. This was clearly evident in a very recent article in AGARD Highlights, "Missile System Simulation and Validation," by Dipl.-Ing Roland Cauggel of BGT, March 1984. (It is interesting to note that this paper discusses "Simulation Levels" very similar to the "CLIMB Levels" in Section III, apparently conceived independently from the Working Group activity.)

It is one thing to establish that uniform validation methods are needed; yet another thing to develop these methods. As stated in Section I Introduction, the Working Group found the latter a formidable task. Nonetheless, the general framework of a validation, or more precisely, "simulation confidence building procedures" is presented in Section III. It was very difficult to overcome the temptation to associate this approach with "documentation alone." While the appendices do show the documentation application, the concept is in fact much broader.

Numerous discussions by Working Group members were required to develop a "unified" understanding of the process itself and its broader use. Even as this report was written, some members felt that the process may be too complex or too difficult to understand, and therefore, perhaps will never be used. On the other hand, almost all of the Working Group members felt that a primary activity by the group is to "preach" the "process" to others in their respective countries. Publication of this Advisory Report should support this recommendation.

Other specific aspects of missile system validation were covered (e.g., computer languages, verification/validation/assessment methods) in Sections IV and V. However, these treatments were of necessity quite limited. Each area covered, in fact, serves as a topic for additional Working Groups or, perhaps, AGARDographs. But before recommending any specific actions of this type to the Flight Mechanics Panel, the Working Group felt this report should be published and widely circulated within AGARD for comment on the overall topic. Only after feedback from experts and potential users should additional action be taken by the panel.

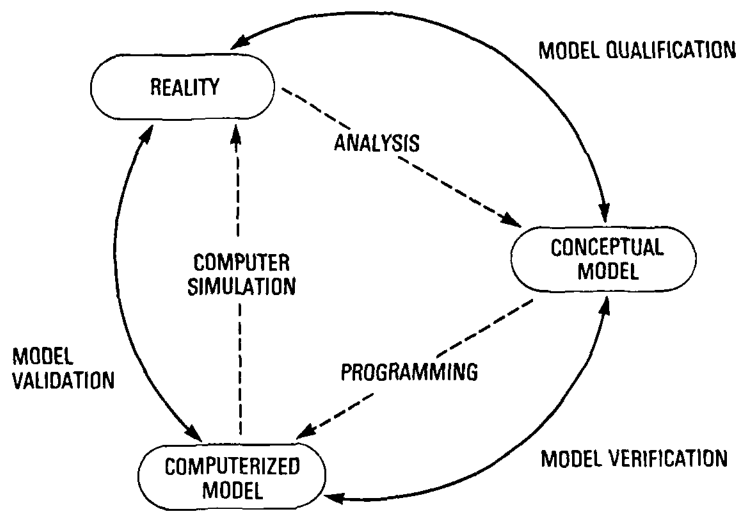


Fig 2.1 Credibility Framework

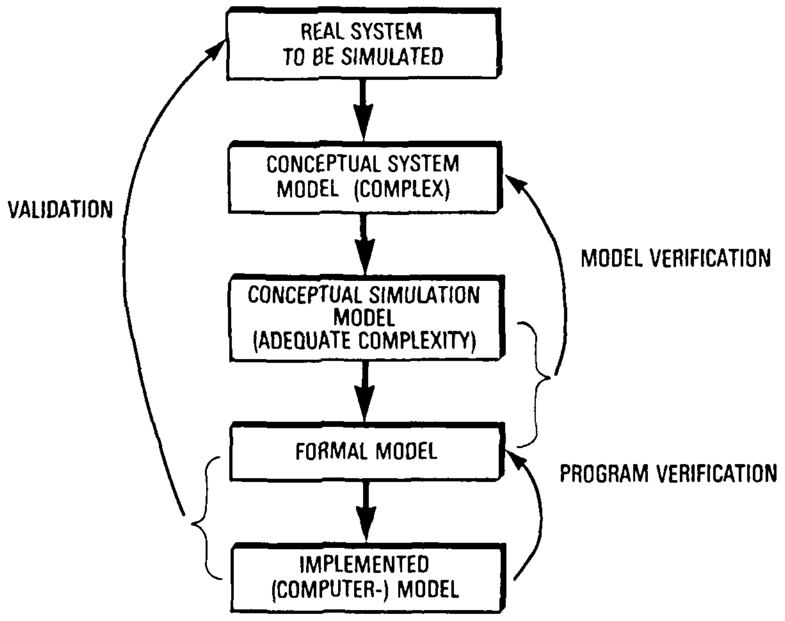

REAL SYSTEM TO BE SIMULATED
VALIDATION
CONCEPTUAL SYSTEM MODEL (COMPLEX)
MODEL VERIFICATION
CONCEPTUAL SIMULATION MODEL (ADEQUATE COMPLEXITY)
FORMAL MODEL
PROGRAM VERIFICATION
IMPLEMENTED (COMPUTER-) MODEL

Fig 2.2 Model Building Process and Associated Documentation
if Simulated System Really Exists

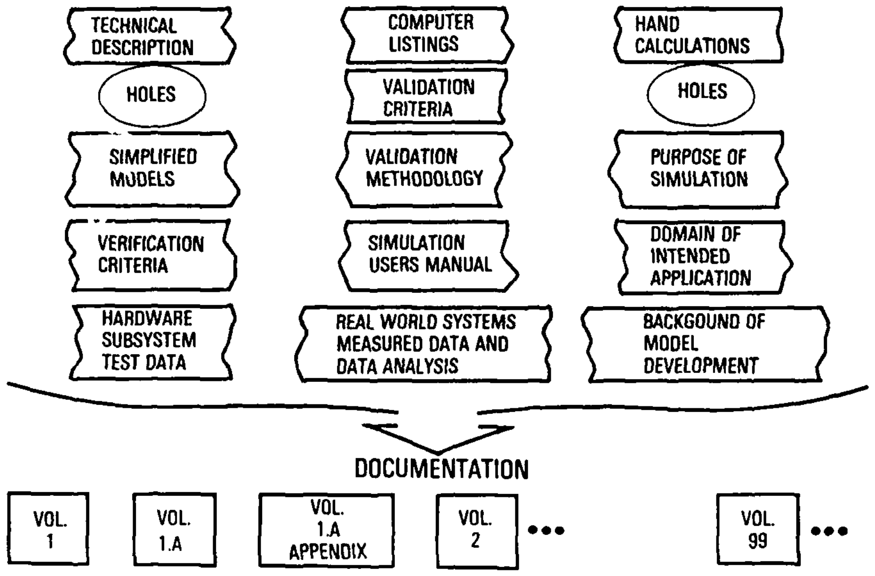


Fig 3.1 Typical Documentation of Simulation Models

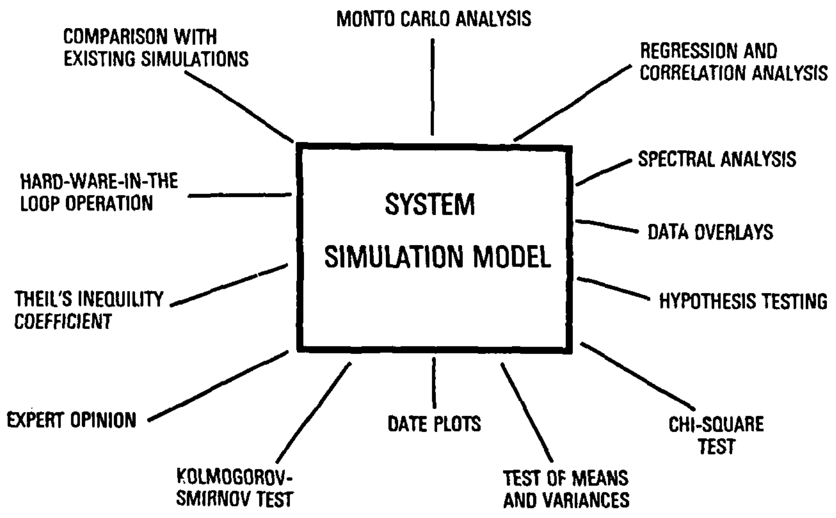

MONTO CARLO ANALYSIS
COMPARISON WITH EXISTING SIMULATIONS
REGRESSION AND CORRELATION ANALYSIS
SPECTRAL ANALYSIS
HARD-WARE-IN-THE LOOP OPERATION
SYSTEM SIMULATION MODEL
DATA OVERLAYS
THEIL'S INEQUILITY COEFFICIENT
HYPOTHESIS TESTING
EXPERT OPINION
DATE PLOTS
CHI-SQUARE TEST
KOLMOGOROV-SMIRNOV TEST
TEST OF MEANS AND VARIANCES

Fig 3.2 Developing Simulation Model Credibility
Using the Random Tool Box Approach

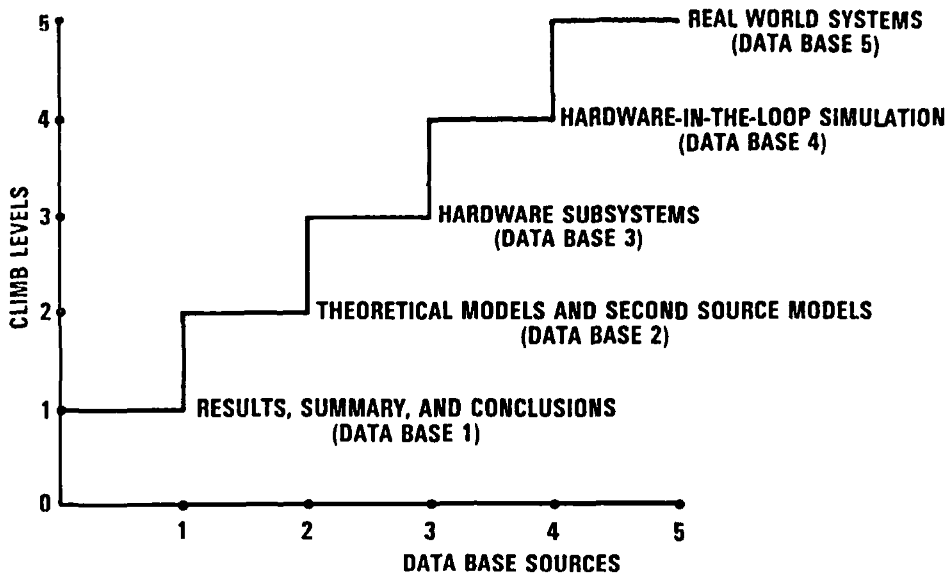


Fig 3.3 Climb Process for Developing Model Credibility

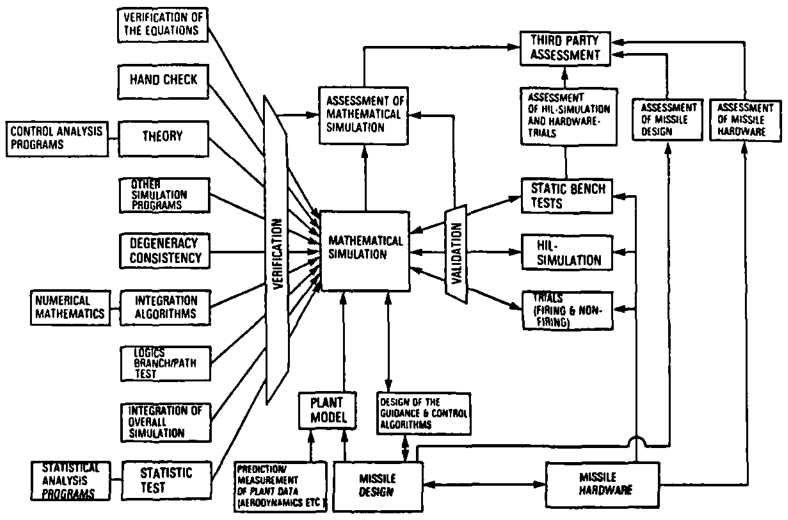

VERIFICATION OF THE EQUATIONS
HAND CHECK
CONTROL ANALYSIS PROGRAMS
THEORY
OTHER SIMULATION PROGRAMS
DEGENERACY CONSISTENCY
NUMERICAL MATHEMATICS
INTEGRATION ALGORITHMS
LOGICS BRANCH/PATH TEST
INTEGRATION OF OVERALL SIMULATION
STATISTICAL ANALYSIS PROGRAMS
STATISTIC TEST
VERIFICATION
ASSESSMENT OF MATHEMATICAL SIMULATION
MATHEMATICAL SIMULATION
VALIDATION
PLANT MODEL
DESIGN OF THE GUIDANCE & CONTROL ALGORITHMS
PREDICTION/ MEASUREMENT OF PLANT DATA (AERODYNAMICS ETC)
MISSILE DESIGN
MISSILE HARDWARE
THIRD PARTY ASSESSMENT
ASSESSMENT OF HIL-SIMULATION AND HARDWARE-TRIALS
ASSESSMENT OF MISSILE DESIGN
ASSESSMENT OF MISSILE HARDWARE
STATIC BENCH TESTS
HIL-SIMULATION
TRIALS (FIRING & NON-FIRING)

Fig 5.1 Interactions Between Verification, Validation and Assessment

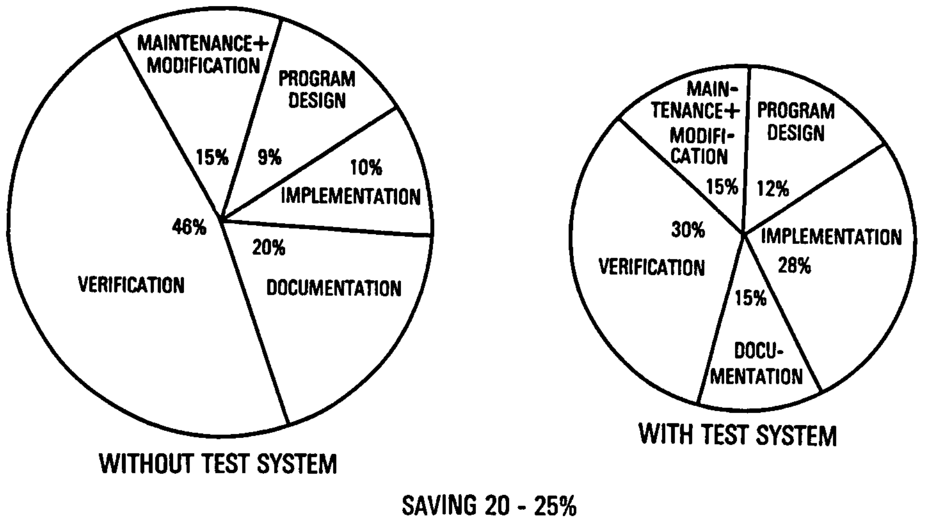


Fig 5.2 Importance of Testing

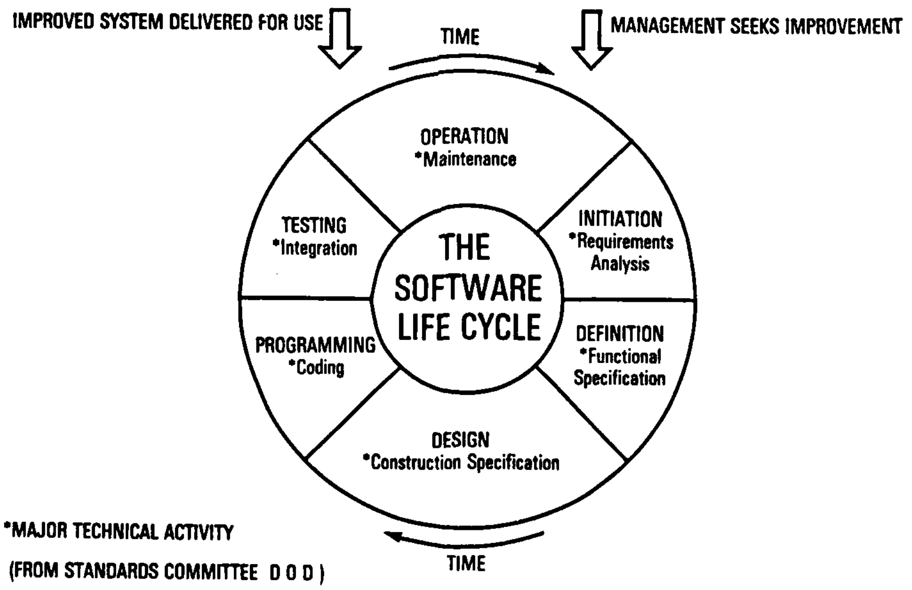

IMPROVED SYSTEM DELIVERED FOR USE
TIME
MANAGEMENT SEEKS IMPROVEMENT
OPERATION
•Maintenance
TESTING
•Integration
THE SOFTWARE LIFE CYCLE
INITIATION
•Requirements Analysis
PROGRAMMING
•Coding
DEFINITION
•Functional Specification
DESIGN
•Construction Specification
•MAJOR TECHNICAL ACTIVITY
(FROM STANDARDS COMMITTEE D O D)
TIME

Fig 5.3 The Software Life Cycle

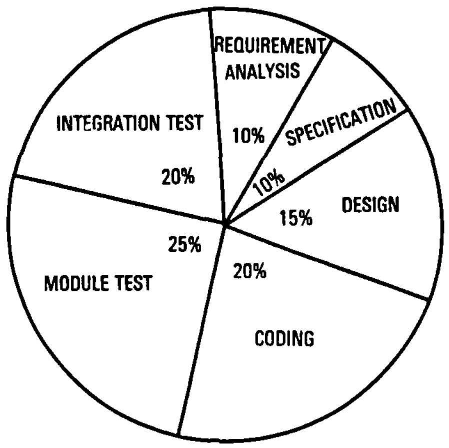


Fig 5.4 Development Effort Distribution
(Excluding Maintenance)

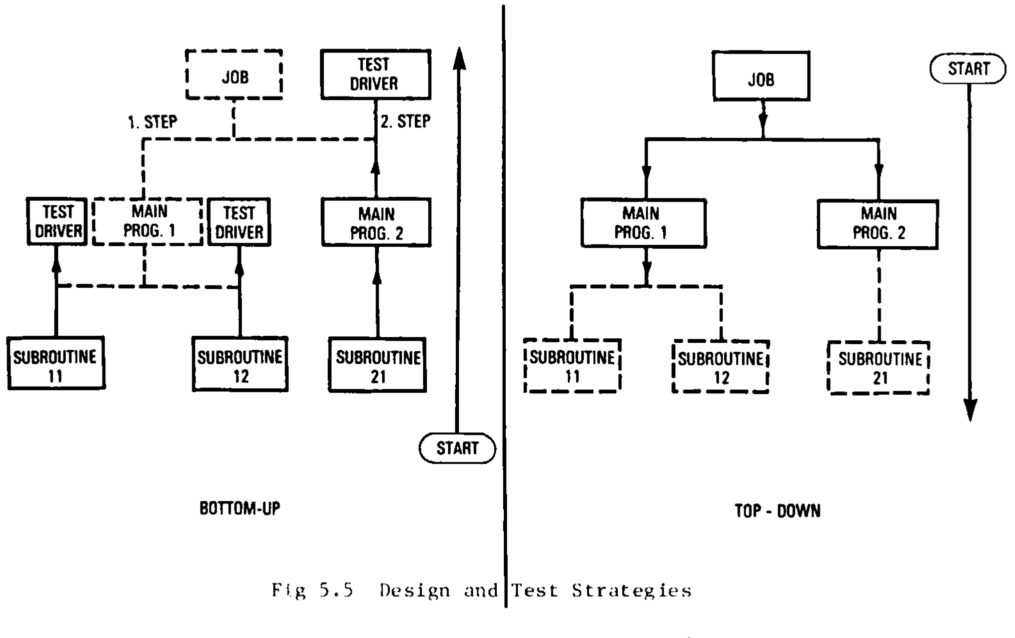


Fig 5.5 Design and Test Strategies

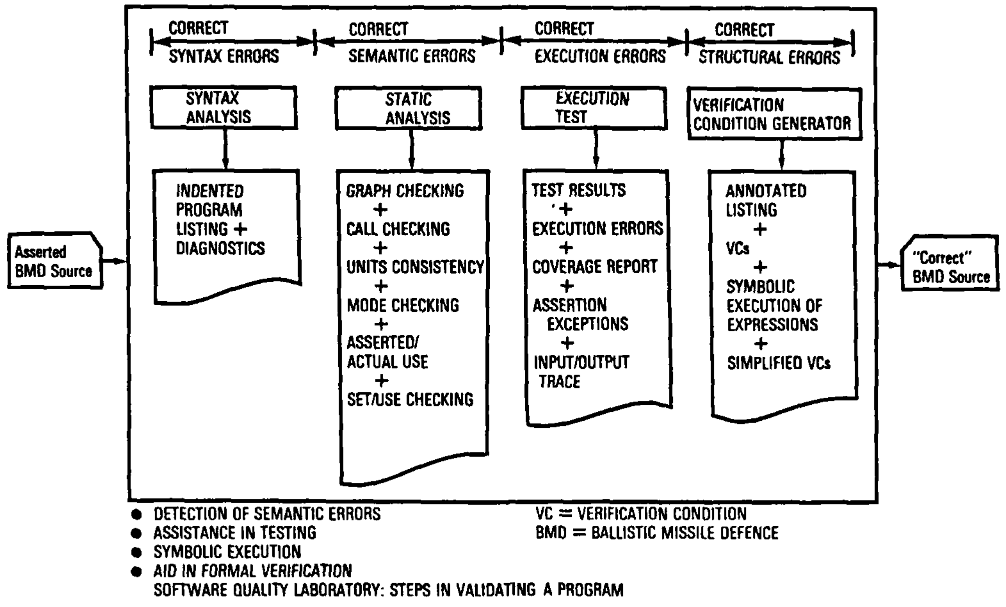

CORRECT SYNTAX ERRORS
CORRECT SEMANTIC ERRORS
CORRECT EXECUTION ERRORS
CORRECT STRUCTURAL ERRORS
SYNTAX ANALYSIS
STATIC ANALYSIS
EXECUTION TEST
VERIFICATION CONDITION GENERATOR
INDENTED PROGRAM LISTING + DIAGNOSTICS
GRAPH CHECKING + CALL CHECKING + UNITS CONSISTENCY + MODE CHECKING + ASSERTED/ ACTUAL USE + SET/USE CHECKING
TEST RESULTS + EXECUTION ERRORS + COVERAGE REPORT + ASSERTION EXCEPTIONS + INPUT/OUTPUT TRACE
ANNOTATED LISTING + VCs + SYMBOLIC EXECUTION OF EXPRESSIONS + SIMPLIFIED VCs
Asserted BMD Source
"Correct" BMD Source
● DETECTION OF SEMANTIC ERRORS
● ASSISTANCE IN TESTING
● SYMBOLIC EXECUTION
● AID IN FORMAL VERIFICATION
SOFTWARE QUALITY LABORATORY: STEPS IN VALIDATING A PROGRAM
VC = VERIFICATION CONDITION
BMD = BALLISTIC MISSILE DEFENCE

Fig 5.6 Test Systems

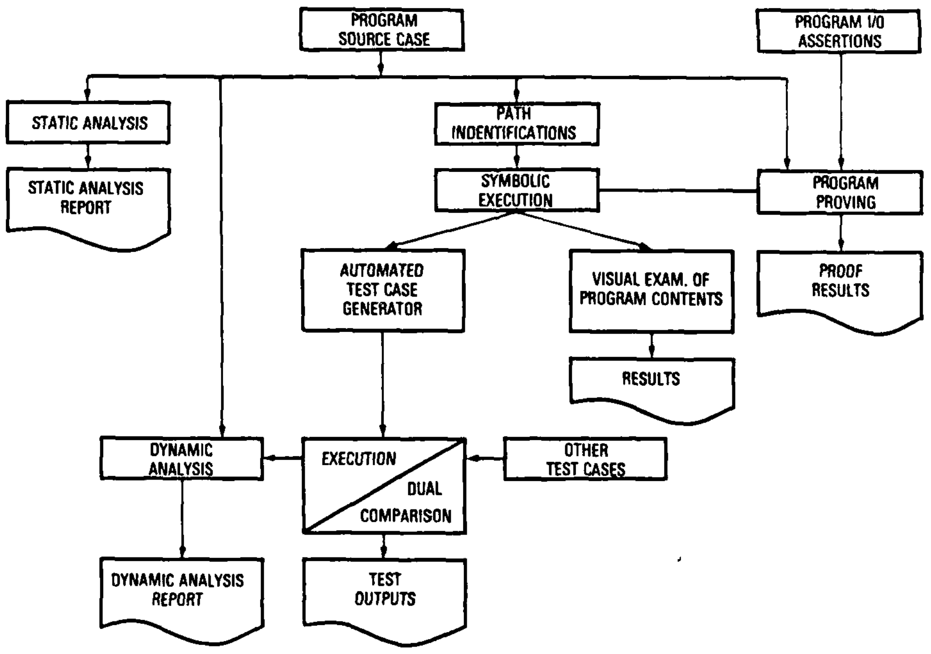


Fig 5.7 Test Systems

AIMS:
- AUTOMATISATION OF TESTING USING TOOLS
- VALIDATION OF TEST RESULTS WITH AUTOMATIC COMPARISONS
- SECURITY OF THE FUNCTION USING REDUNDANT SOLUTIONS

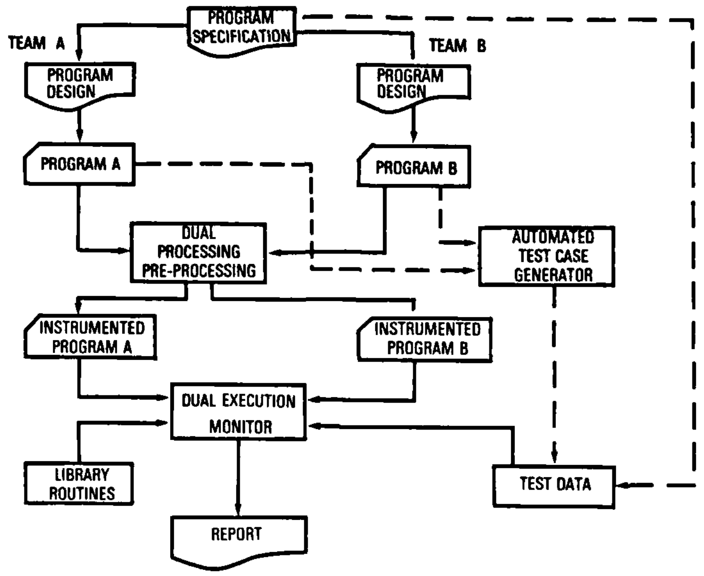


Fig 5.8 Dual Program Analysis (NASA)

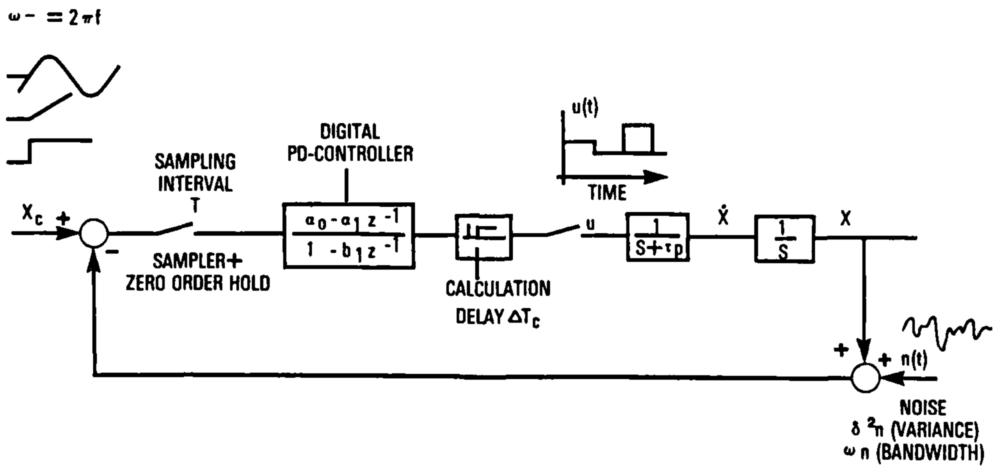


Fig 5.9 Sample-data Control System

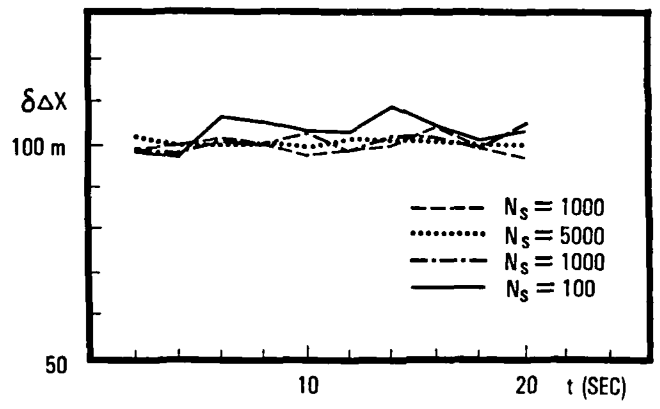


Fig 5.10 "Measured" Variance as a function of the Number of Runs N_s

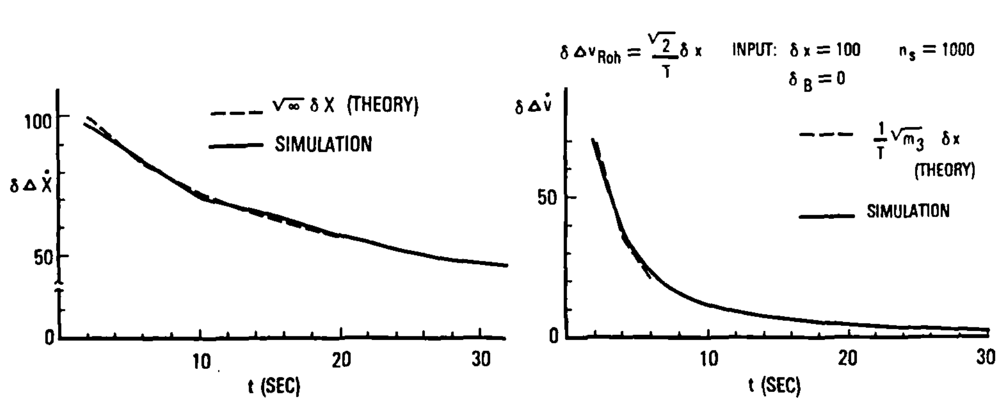


Fig 5.11 Noise Reduction of a Typical Kalman Filter. Comparison of Theory Against Simulation.

VII REFERENCES

1. Holmes, Willard M., "Survey of Missile Simulation and Flight Mechanics Facilities in NATO, AGARDograph No 279, April 1983

2. Sneed Harry, M., Systematisches Programmtesten. Ein muhsames Geschaft. Sneed Software Service. Fachseminar uber Software-Testverfahren, Munchen, 1981

3. Bohem, Barry, "Software and Its Impact: A quantitative assessment", in DATAMATION, May 1973

4. Hellerman, H., Digital Computer System Principles, New York, 1967

5. Miller, E., "Program Testing: Art Meets Theory", Computer, July 1977

6. Pietrasanta, A., "Management Control in Progarm Testing", IBM-SDD Report, Poughkeepsie, 1966

7. Mills, H., "Top-Down Programming in Large Systems", Debugging Techniques in Large Systems, New York, 1971

8. Meyers, G., Software Reliability-Principles and Practice, New York, 1977

9. Dijkstra, E., NATO Conference on Software Engineering, Garmisch, 1968

10. Constantine, L., Structured Design, New York, 1976

11. Howden, W., "Reliability of the Path Analysis Testing Strategy", IEEE Transactions on Software Engineering, September 1976

12. Goodenough/Gerhart, "Toward a Theory of Test Data Selection, IEEE Transactions on Software Engineering, June 1975

13. Gewald/Haake/Pfadler, "Software Engineering", Grundlagen und Technik rationeller Programmentwicklung, Munchen, 1977

14. Schlesinger, S. et al, "Developing Standard Procedures for Simulation Validation & Verification". The Aerospace Corporation, EL Segundo, California, 1974, page 927-933

15. Ragazzini, John R.; Franklin, Gene F., "Sampled-data Control Systems" McGraw-Hill Book Company, Inc. 1958

16. Cellier, F.; Rimvall, M., "Computer Aided Control Systems Design", First European Simulation Congress ESC 83, Aachen, Sept 1983, Springer-Verlag, page 1-21

17. Blakelock, J. H., "Design and Analysis for a Digitally Controlled Integrated Flight/Fire Control System," J. Guidance, 1983, Page 251-257

18. Petzold, Linda R., Applied Mathematics Division, Sandia National Laboratories, Livermore, CA, "Multistep Methods: An Overview of Methods, Codes and New Developments", Proceedings SCSC, Vancouver, 1983, Page 1024-1031

19. Bub, W., "Zur digitalen Simulation von linearen Übertragungs-systemen," 1967

20. Goult, F. J. et al, Applicable Mathematics, The Macmillan Press LTD 1978

21. Hanselmann, H. Paderborn, "Simulation of Fast Digital Control Systems" First European Simulation Congress ESC 83 Aachen, September 1983 Springer-Verlag, page 405-411

22. Mitchell and Gauthier Assoc., ACSL - Advanced Continuous Simulation Language, Concord, MA 01742, Third Edition, 1981

23. Naylor, T. H.; Finger, J. M., "Verification of Computer Simulation Models, "Management Science, Vol 14, No 2, October 1967, page B-92-101

24. Van Horn, R. L., "Validation of Simulation Results," Management Science, Vol 17, No 5, January, 1971, page 247-257

25. Wright, R. D., "Validating Dynamic Models: An Evaluation of Tests of Predictive Power," Proceedings, 1972 Summer Computer Simulation Conference, 1972, page 1286-1296

26. Kheir, N. A.; Holmes, W. M., "On Validating Simulation Models of Missile Systems," Simulation, April, 1978, page 117-128

27. Kheir, Naim A., "A Validation Procedure for Missile Systems Simulation Models," Seventh Annual Conference on Modeling and Simulation, Pittsburg, 1976, page 534-539

28. Rowland, J. R.; Holmes, W. M., "Simulatino Validation with Sparse Random Data," Computers and Electrical Engineering, Vol 5, No 1, March, 1978, page 37-49

29. Senge, P. M., "Some Issues in Evaluating the Validity of Social System Models," Proceedings, 1973 Summer Computer Simulation Conference, 1973, page 1176-1177

30. Fishman, G. S.; Kiviat, P. J., "The Analysis of Simulatino Generated Time Series," Management Science, Vol 13, No 7, March 1967, page 525-557

31. Hsu, Der-Ann; Hunter, J. S., "Validation of Computer Simulation Models Using Parametric Time Series Analysis," Proceedings, 1974, Winter Simulation Conference, 1974, page 727-728

32. Hunt, A. W., "Spectral Analysis Applied to Digital Simulation Models," Paper Presented at the Tenth American Meeting of TIMS, October 3, 1969, Atlanta

33. Hunt, A. W., "Statistical Evaluation and Verification of Digital Simulation Models Through Spectral Analysis," Ph.D Dissertation. The University of Texas at Austin, 1970

APPENDIX A

DEVELOPING "CONFIDENCE LEVELS IN MODEL BEHAVIOR" OR THE "CLIMB" PROCESS

CLIMB LEVEL 1. MODEL SUMMARY, RESULTS AND CONCLUSIONS

Level 1 includes identification of the model developer, summary of the model, and general description of the model with simplified diagrams. The objective of the simulation, domain of intended application and criteria for model validation is stated here. The model's critical variables and major assumptions used in model development are identified and conclusions on overall model performance is stated.

1. MODEL ORIGIN AND RELATED INFORMATION

   1.1 Total name of simulation model
   1.2 Name of developing organization
   1.3 Address of organization
   1.4 Name of contact for additional information about model
   1.5 Address of contact for model information
   1.6 Telephone number of contact
   1.7 Organization for which model was developed
   1.8 Address of organization
   1.9 Contact person
   1.10 Telephone number of contact
   1.11 Keywords for data base processing

2. OBJECTIVES IN DEVELOPING THE SIMULATION MODEL

   2.1 Objectives of the simulation
   2.2 Background information leading to model development

3. MODEL SUMMARY

   3.1 Definition of terms (omitting all symbols)
   3.2 Conceptual model showing major input/output variables
   3.3 Summary statement and descriptive documentation on model application
   3.4 Nature of model (Discrete, Continuous, Stochastic, etc.)

4. FUNCTIONAL MODEL

   4.1 Description of functional model
   4.2 Simplified functional diagram with major subsystem and major variables identified
   4.3 Definitions of and comments on major variables in functional diagram
   4.4 Critical variables identified for model validation

5. MODEL APPLICATION

   5.1 Domain of intended application of simulation model
   5.2 Major assumptions used in developing the model

5.3 Major known limitation in domain of application
5.4 Nonobvious exclusions from model
5.5 Inputs to and Outputs from Model

6. PHILOSOPHY

6.1 Criteria for validation
6.2 Methodology for validation

7. SUMMARY COMMENTS ON SIMULATION IMPLEMENTATION

7.1 Type computer and operating system for simulation
7.2 Computer language or simulation language used

8. STUDIES OR AREAS WHERE MODEL HAS BEEN USED

8.1 Specific studies where model was used
8.2 Related model background

9. COMMENTS ON MODEL PERFORMANCE

9.1 Summary of validation results
9.2 CLIMB levels achieved
9.3 General conclusions on model performance

10. APPLICABLE DOCUMENTS

<u>CLIMB LEVEL 2.</u> SYSTEM MODELS AND SUBMODELS THEORETICAL AND INDIRECT DATA BASES

Simulation model and submodel performances are compared with theoretical models and/or existing appropriate validated simulation models. Methods of comparing model performances are identified and results given at the level of visual inspection, expert opinion and plot overlays. Analysis methodology with assumptions and deficiencies are identified.

1. SYSTEM MODEL ELEMENTS

1.1 General description of system model
1.2 Block diagram of system model
1.3 Identification of major subsystems
1.4 Assumptions and justifications used system model development

2. IMPLEMENTATION DESCRIPTION

2.1 List of computer variables
2.2 Processing methods used and relevant parameter identification (i.e., integration methods, initialization methods, computer word length, etc.)
2.3 Required program library elements
2.4 Required computer resources

3. SYSTEM MODEL VERIFICATION

- 3.1 Criteria for model verification
- 3.2 Identify methods used for model verification
- 3.3 Identify data base used for model verification
- 3.4 Results from model verification

4. VALIDATION OF SYSTEM MODEL'S STOCHASTIC COMPONENTS

- 4.1 Identification of stochastic components
- 4.2 Criteria for achieving validation
- 4.3 Validation methods and techniques used (Comparisons of means, variances, distributions, etc.)
- 4.4 Data bases used for validation
- 4.5 Results from validation effort

5. VALIDATION AGAINST OTHER EXISTING MODELS

- 5.1 Identification of existing models used
- 5.2 Criteria for achieving validation
- 5.3 Validation methods and techniques used (Comparisons of means, variances, distributions)
- 5.4 Data bases used for validation
- 5.5 Results from validation effort

6. SUBSYSTEM CHARACTERIZATION AND BRIEF DESCRIPTION OF SUBSYSTEM MODELS

- 6.1 General description of model
- 6.2 Block diagram of subsystem model
- 6.3 Criteria for validation
- 6.4 Validation methods used
- 6.5 Validation results

7. BENCHMARK TEST CASE

- 7.1 Description of benchmark test case
- 7.2 Input data and computer configuration for test run
- 7.3 Output data or sample results for critical variables
- 7.4 Criteria for acceptability of benchmark results

8. COMPUTER PROGRAMS

- 8.1 User instructions
- 8.2 Computer listing

9. PROGRAM VERIFICATION

- 9.1 Criteria for verification
- 9.2 Identify methods for verification
- 9.3 Identify data base used for program verification
- 9.4 Results from program verification

10. APPLICABLE DOCUMENTS

CLIMB LEVEL 3. SUBSYSTEM REAL WORLD DATA BASE

Real world data is usually available on at least one major subsystem for comparing with simulated model results. Schematics and technical documentation of the total complex model is included only if the need arises resulting from the validation efforts described here. Statistical, logical, or deterministic methods are identified for achieving validation of the subsystem model. Acceptability of the submodel is noted.

1. REAL WORLD SUBSYSTEM DATA

1.1 Identification of subsystem.

1.2 List of variables for which measured data exist. (Real world data recorded in a format consistent with the format of the simulated generated data.)

1.3 Data in hard copy, i.e. charts, graphs, plots, etc. provided to correspond to the critical variables identified in CLIMB LEVEL 1. The source should be identified of any additional data available.

2. EXPERIMENTAL TEST ENVIRONMENT

The description includes information at the subsystem level not shown in CLIMB LEVEL 2 diagrams, i.e., inputs, outputs, test points, scale factors, submodel linkages, etc.

2.1 Scenario used to excite subsystem

2.2 Description of test experiment

3. METHODS AND TECHNIQUES USED IN COLLECTING REAL WORLD DATA

3.1 Data collecting methods

3.2 Error sources associated with input and output measurements

3.3 Analysis performed on input and output measured data

3.4 Contact for further information on measured data

- Name
- Company address
- Telephone number

4. THE APPROACH USED FOR VALIDATING THE SUBMODEL USING THE REAL WORLD DATA

   4.1 Criteria for validation

   4.2 Validation methods used

   4.3 Validation results

   4.4 Model changes due to validation effort

5. TECHNICAL DOCUMENTATION FOR EXCITATION METHODS (EXCITATION SOURCES MAY BE DIFFERENT FOR SUBMODEL AND REAL WORLD SUBSYSTEMS)

   5.1 Description of excitation method(s)

   5.2 Documentation of excitation data

   5.3 Documentation of real world subsystem response

   5.4 Documentation of submodel response

   5.5 Computer program listing of excitation methods (The computer listing of the submodel simulation will be shown if different than the listing shown in CLIMB LEVEL 2)

   5.6 Excitation program for hardware test configuration

6. USER INSTRUCTION FOR TEST SET UP

7. BENCHMARK FOR TEST SET UP

   7.1 Description of benchmark test case

   7.2 Initial conditions for test set up

   7.3 Output data or sample results for critical variables

   7.4 Criteria for acceptable benchmark results

8. APPLICABLE DOCUMENTS

<u>CLIMB LEVEL 4</u>. HARDWARE-IN-THE-LOOP OPERATION

A data base is available from a hardware-in-the-loop operation using a major subsystem hardware component, e.g., for missile systems, an autopilot, sensors/seekers, embedded computers, actuators, etc. A typical process is to include hardware used to collect data from CLIMB LEVEL 3. Results from the model versus hardware performance comparison is reported with specifics on methods used for data comparisons and performance validation. Included are specifics on any additional model development environment for RF/EO/IR seekers.

1. DESCRIPTION OF HARDWARE-IN-THE-LOOP (HWIL) SYSTEM

   1.1 Description of hardware to be used for HWIL operation.

   1.2 Description of computer system (Analog, Digital, Hybrid) used for HWIL operation. Specifically, was the all digital simulation program partitioned between a digital and analog computer?

2. PARTITIONED MODEL FOR HWIL OPERATION

   2.1 Diagram of the partitioned model showing elements of the model to be replaced with hardware (Hardware is defined as any outside element connected to the digital, analog, or hybrid computer relating to the real world system.)

   2.2 Assumptions and criteria for selecting the particular partitioned configuration of the model.

   2.3 Model variables partitioned between the digital and connecting systems, including the analog computer.

   2.4 Identify the model variables showing range and scale factors for the connecting systems.

3. RESULTS OF HARDWARE IN THE LOOP OPERATION

   3.1 Time history plots of critical variables showing the all digital computer results and HWIL results (Method of showing results includes means and variances for systems with random components.)

   3.2 Identify any data analysis performed for comparing the all digital and HWIL results, i.e., time correlation analysis, distribution function testing, power spectral density testing, etc.

   3.3 Identify and change the all digital simulation model based on results from HWIL operation.

4. COMPUTER PROGRAM

   4.1 Partitioned digital program

      4.1.1 Identify changes or modification to the digital computer program required for HWIL operation, i.e., use of real time library subroutines, integration methods, etc.

4.1.2 Input conditions and test scenario for executing the all digital simulation partitioned for HWIL configuration.

4.1.3 Expected output data from test scenario

4.1.4 Identify special program development for HWIL operations, i.e., real time data recording, online data analysis real time interrupt drivers

4.1.5 List special programs required for real time or HWIL operations.

4.2 Connecting Systems

4.2.1 Identify the critical variables between the digital program and connecting systems.

4.2.2 Identify error sources associated with connecting system variables.

4.3 Model Variables

4.3.1 Show verification of the partitioned model against the unpartitioned model

4.3.2 Show verification of the HWIL system against the partitioned model

4.4 Model Validation

4.4.1 Validation of the partitioned model against the unpartitioned model.

4.4.2 Verification of HWIL system against the partitioned model.

CLIMB LEVEL 5. TOTAL REAL WORLD SYSTEMS OPERATIONS

A data base is available from operating the total real world system. As a minimum, results are reported on the validation effort for the system's critical variables operating in the domain of intended application. Specifics on validation methodology and performance comparisons are reported. Evaluation and conclusions are made regarding the system model performance and deficiencies.

1. CONCLUSIONS AND COMMENTS ON MODEL VALIDATION EFFORT USING REAL WORLD SYSTEMS TEST RESULTS

2. DESCRIPTIVE SUMMARY OF REAL WORLD TEST CONDITIONS AND TEST RESULTS

3. SYSTEM TEST ENVIRONMENT

   3.1 Purpose of test

   3.2 Description of test measurement methodology

   3.3 Location of test site and ambient conditions as related to system tests; i.e., temperature, pressure, wind velocity, humidity, etc.

   3.4 Error tolerances in the measurement system

4. DESCRIPTION OF TEST SCENARIO USED TO STIMULATE THE REAL WORLD SYSTEM

   4.1 Description of target or system test driver

   4.2 Target or test driver initial conditions

   4.3 Method used for reconstructing system test driver. (An example for a missile system, reconstructing the target trajectory would be required.)

   4.4 Reconstructed system test driver data

5. REAL WORLD SYSTEMS PERFORMANCE RECONSTRUCTION

   5.1 Initial conditions on system parameters as measured

   5.2 Method of system performance reconstruction

   5.3 Reconstructed system performance data (Example: Missile position and velocity history, time of flight to closest approach, position of closest approach, etc.)

   5.4 Measured data on system's critical variables

6. STRUCTURING OF SIMULATION MODEL FOR SYSTEM TEST CONDITIONS

   6.1 Identify simulation model variables initialized using system test data.

   6.2 Identify assumptions made about simulation model initial conditions for operating with system test conditions.

   6.3 Simulation model generated data using real world system test conditions.

7. ANALYSIS OF SIMULATED MODEL GENERATED DATA AND SYSTEM TEST RESULTS

   7.1 Identify methodology of data comparison.

   7.2 Results of comparing simulation model generated data and system test results.

8. IDENTIFY AND EXPLAIN DISCREPANCIES BETWEEN ADJUSTED MODEL GENERATED DATA AND REAL WORLD SYSTEM TEST RESULTS

9. RECOMMENDATION FOR MODEL IMPROVEMENT

APPENDIX B

EXAMPLES OF THE "CLIMB" PROCESS

The objective of this appendix is to illustrate the use of the Confidence Levels in Model Behavior (CLIMB) process using as examples an actual simulation of an electrical actuation system for a missile. These examples follow the outlines of CLIMB Levels 1, 2 and 3 presented in Appendix A. The simulation was performed at Messerschmitt-Bolkow-Blohm GmbH under the code named ELACT 3.

CLIMB LEVEL 1 EXAMPLE

1. MODEL ORIGIN AND RELATED INFORMATION

1.1 Total Name of Simulation Model

ELACT3: Electrical Actuation System

1.2 Name of Developing Organization

Messerschmitt-Bolkow-Blohm GmbH

1.3 Address of Developing Organization

Messerschmitt-Bolkow-Blohm GmbH
Abteilung AE13
Postfach 801149
D-8000 Munchen 80
W-Germany
Telex: 5287-0 mbb d

1.4 Address of Contact for Model Information

- Person: Werner Bub
- Address: same as Paragraph 1.3
- Phone No: 089-60004125

1.5 Address of Contact for Additional Information About Model

- Persons:

Fridbert Kilger, Phone No 089-60002302
Herman Neubauer, Phone No 089-60006364

- Address:

  Same as para 1.3

1.6 Organization for Which Model was Developed

Same as para 1.3

1.7 Contact Person

- Person:

  Alfred Huber, Phone No 089-60000815

- Address:

  Same as para 1.3

1.8 Keywords for Data Base Processing

Electrical Actuator, Missile Simulation

2. OBJECTIVES IN DEVELOPING THE SIMULATION MODEL

2.1 Objectives of the Simulation

In tactical missile systems, a set of four fins moved by actuators usually constitute the control surfaces of the missile airframe. These are contained in the autopilot loop to produce the three rotational degrees of freedom of the missile.

Objective of the development of the present model was to provide a subsystem model of an electrical actuator system, taking hinge moments into account, that could be included in an overall missile system simulation model.

2.2 Background Information Leading to Model Development

The Model was developed in 1978 to be used for simulation of the EMS Experimental Missile System.

3. MODEL SUMMARY

3.1 Definition of Terms

Commanded fin deflection	Input variable, fin deflection demanded from the autopilot system
External hinge moment	Input variable, hinge moment generated by the aerodynamic fin forces and moments

Device locked	Logical input variable, if true, the actuator is locked in its initial position
Actual fin deflection	Output and state variable, actual angular position of the shaft on which the fin is mounted
Motor speed	Output and state variable, angular speed of the shaft of the electrical motor
Commanded motor current	State variable, output variable of the actuator controller
Actual motor current	Output and state variable, actual motor current generated in response to demand from actuator controller
Motor feeding voltage	Output variable, voltage across the terminals of the electrical actuator motor

3.2 Conceptual Model Showing Major Input/Output Variables

Actual fin deflections, motor speed, motor voltage and motor current are computed as a function of commanded values for desired fin deflections, the moments actually acting on the hinges, and a logical variable which determines whether the device is locked or unlocked. Fig 1 shows the basic functions which model ELACT3 performs.

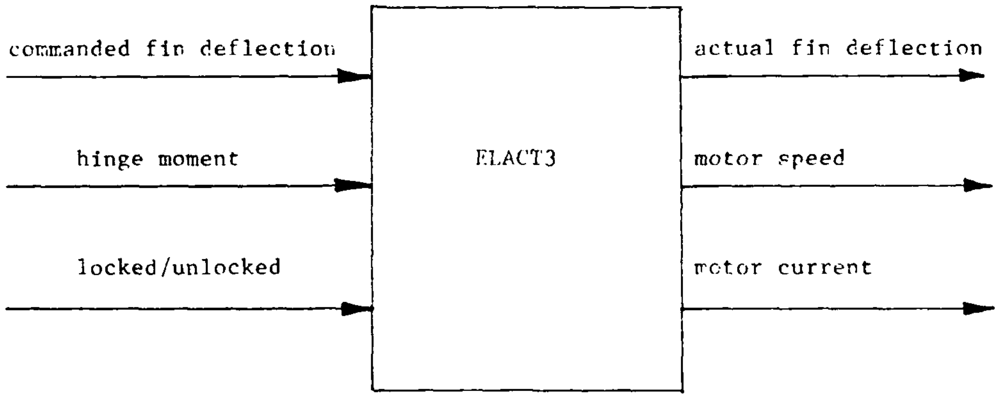


Fig B-1 Basic Model Functions

3.3 Summary Description of Model Application

Model ELACT3 can be used in the scope of missile models if a model of the aerodynamic hinge moments acting on the fins is available. Intended applications are:

- Autopilot studies
- System simulation studies - if hinge moments have a sensible effect on system performance or if an estimate of overall power consumption during a mission has to be obtained
- Actuator design studies for verification of basic design parameters
- Usage as a model of a typical electrical actuator system for other applications where load moments are important

3.4 Nature of Model

The model is continuous, i.e., it is described by three ordinary differential equations. The model is basically of deterministic nature in the sense that it does not contain any internal sources of noise.

4. FUNCTIONAL MODEL

4.1 Description of Functional Model

Since the influence of external hinge moments is taken into account, a physically functional model is necessary. Therefore, the model is composed of the actuator controller, the electrical power amplifier, the electrical shunt motor with gear drive, and pickups for motor speed and fin position. The dynamics of the power amplifier as well as the dynamics and higher order effects of the rotor circuit are neglected. The overall dynamics for small signals corresponds to a third order transfer function. Nonlinear behaviour is the result of limits for motor current, motor voltage, and motor speed, which are represented in the model.

4.2 Functional Block Diagram and Major Variables

The model is composed of the following functional blocks:

- Actuator controller
- Electrical power amplifier
- Electrical motor with associated gear drive
- Sensors for fin position and motor speed

The relationship between these function of blocks are depicted in the following functional block diagram:

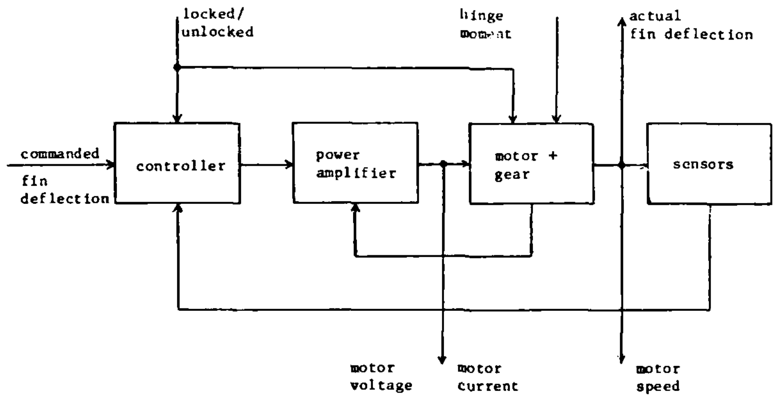


Fig B-2 Functional Block Diagram

4.3 Definition and Comments on Major Variables in Functional Block Diagram

Same as in Paragraph 3.1

4.4 Critical Variables for Model Validation

- Actual fin deflection
- Motor speed
- Actual motor current

5. MODEL APPLICATION

5.1 Domain of Intended Application of Simulation Model

The model can be used without special precautions within the domain defined by its basic design parameters (max fin deflection, max defletion rate, max hinge moments, bandwidth, etc.).

5.2 Major Assumptions Used in Developing the Model

In view of the real actuator system and of the intended applications, the model represents the following features:

- Third order dynamics
- Motor current
- Friction and hinge moments
- Limitations in actual fin deflection, motor speed and motor current
- Rigid body dynamics

5.3 Major Known Limitations in Domain of Application

If the design of the real actuator system is sound, the neglected effects such as backlash, gear efficiency, elasticity of mechanical parts, motor commutation and cogging effects, deterioration of magnetic flux and the dynamics of the power amplifier should not have a sensible effect on the static and dynamic behavior of the device and therefore also on the model.

5.4 Non-obvious Exclusions from Model

Not identified in present model.

5.5 Inputs to and Outputs from Model

5.5.1 Models Providing Inputs

The inputs to model FLACT3 have to be provided by:

- An autopilot model which calculates desired fin deflections

- A model of hinge moments which computes the aerodynamics load moments acting on the actuator hinge as a function of fin incidence.

5.5.2 Models Using Outputs

The principal output of model ELACT3 is actual fin deflection. This output provides data to compute aerodynamic forces and moments acting on the missile body and on the actuator hinges. Additional outputs can be used to monitor actuator performance. The output motor current in conjunction with a power supply model can be used to determine power consumption over missile flight time.

6. MODEL VALIDATION PHILOSOPHY

6.1 Criterion for Validation

Criterion for validation requires that the model response and the response of the real actuator system be matching reasonably well from an engineering point of view using the same kind of system excitation and observing the variables identified in Paragraph 4.4.

6.2 Methodology for Validation

Validation was performed against data obtained from bench test with the real actuation system. A step function for "commanded fin deflection" was applied as an input test function. The system response with respect to the critical variables identified in Paragraph 4.4 were recorded on a multi-channel recorder. The corresponding test was performed with the model and the critical variables were recorded on plots using the same format and scale factors as on the multi-channel recorder. Comparison was performed by visual overlay of the two system responses. Quality of coincidence was judged by engineers experienced in actuator design and in missile modelling. No formal measures for goodness of fit have been used.

7. SUMMARY COMMENTS ON SIMULATION IMPLEMENTATION

7.1 Type Computer and Operating System

The Model is implemented digitally on an CDC 6600 Computer under NDS 1.4, level 552.

7.2 Language

Standard ANSI-FORTRAN IV.

8. STUDIES OR AREAS WHERE MODEL HAS BEEN USED

8.1 Specific Studies where Model was used

The model was used for the purpose mentioned in Paragraph 2.2.

8.2 Related Model Background

8.2.1 Similar Models

Model ELACT3 is a member of a family of several actuator models:

- ACT1: First order general actuator model with position and speed limits.
- ACT2: Second order general actuator model with limits for acceleration, speed and position.
- ELACT3: Third order electromechanical actuator model, taking limits and hinge moments into account.
- ELACT4: Detailed model to be used in electrical actuator design studies.

8.2.2 Model Structure

Model ELACT3 is a stripped version of model ELACT4 that used for design of the actuator. The newly developed actuator system was acceptance tested against results obtained with ELACT4. In ELACT3, only those features are represented that are necessary to meet the objectives mentioned in Paragraph 2.1. The choice has been made by engineers experienced in actuator design and in missile system modelling.

8.2.3 Model Data

The parameters and constants for the model have been taken from model ELACT4 and have been validated by measurements on the actual system during its development.

9. COMMENTS ON MODEL PERFORMANCE

9.1 Summary of Validation Results

Since no device was available which would be capable to apply a defined moment on the hinge of the real system under dynamic conditions, validation was possible only without external load.

For the tests performed, coincidence of the variables "fin deflection" and "motor speed" was very good whereas the motor current of the model matched the current of the real system reasonably well only during acceleration and deceleration phases. A large ripple, which is induced in the real system by motor effects such as cogging, commutation, etc., does not exist in the case of the model since motor effects are not included.

The way mechanical friction was represented in the model was not reasonable. When the model approached a steady state, a limit cycle was generated; the characteristics of which are very sensitive to the implementation parameters (e.g., integration step size).

9.2 CLIMB Level Achieved

CLIMB Level 3 has been achieved.

9.3 General Conclusions on Model Performance

ELACT3 is a reasonable model of a third order actuation system. The static and dynamic performances are well represented. The representation of the motor current in the model allows the correct representation of degradation in dynamic performance when the current reaches its limitation bounds as well as to obtain an estimate of electrical power consumption, whereas, the representation of motor current with respect to time is poor because of the neglected high order effects. Caution has to be observed when using the mechanical friction feature of the model, as explained in Paragraph 9.1.

When the model is used within an autopilot loop, steady state conditions will practically never be reached and the limit cycle will probably never be excited. Therefore, if one wishes to derive an estimate of power consumption of the actuation system, the model could be used taking friction into account if the necessary caution is observed.

10. APPLICABLE DOCUMENTS

ELACT4, Documentation of the Design Model of an Electrical Actuation System.

CLIMB LEVEL 2 EXAMPLE

1. SYSTEM MODEL ELEMENTS

1.1 General Description of System Model

The model is composed of the following functional blocks:

- Actuator controller
- Electrical power amplifier
- Electrical motor with associated gear drive
- Sensors for fin position and motor speed

These can readily be identified in Fig B-3.

1.2 Block Diagram of System Model

Represented by Fig B-3

1.3 Major Subsystems

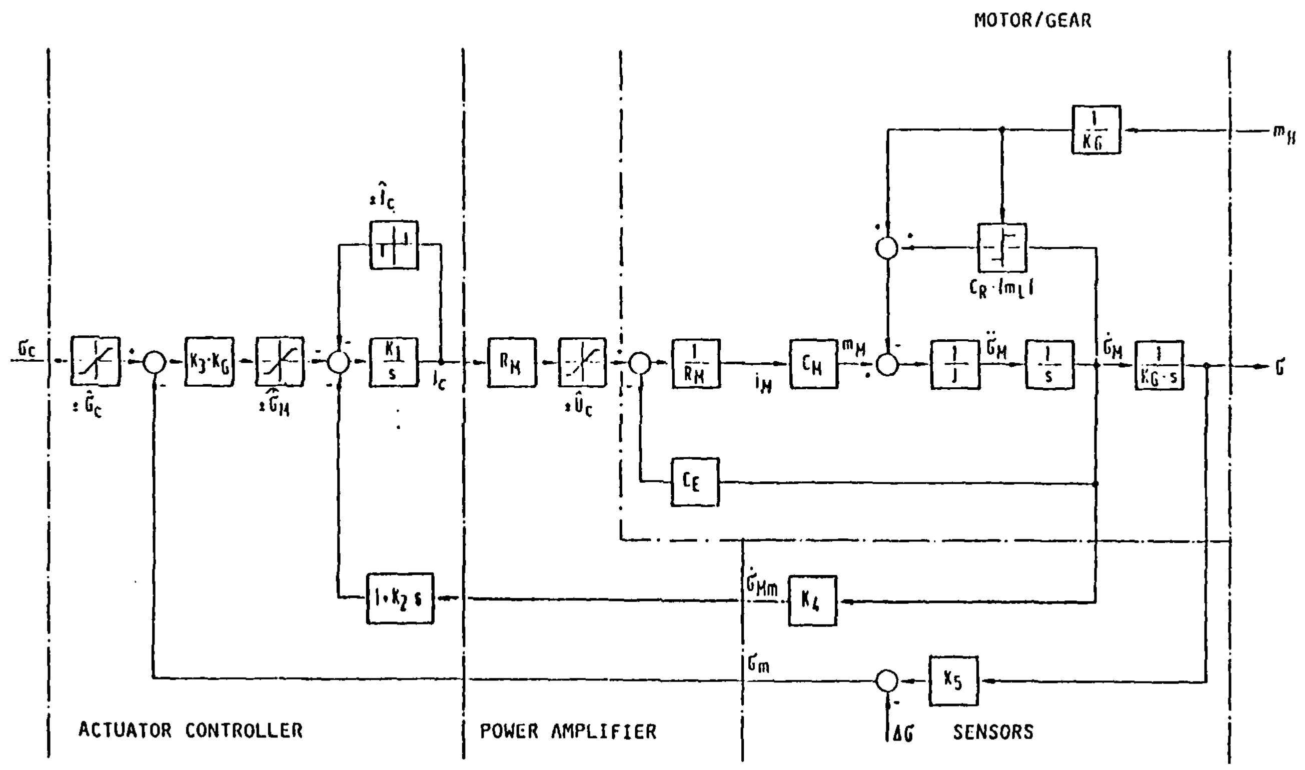


Fig B-3 Detailed Block Diagram

1.3.1 Actuator Controller

The actuator controller is a PID-controller, the three coefficients of which are calculated from the denominator polynomial of the desired third order overall transfer function.

1.3.2 Power Amplifier

The power amplifier provides a current to the motor that is commanded by the actuator controller. Neglecting dynamic effects, it is represented by its steady state behavior taking into account limits for motor current and voltage.

1.3.3 Electrical Motor/Gear/Load

The model of this block uses the basic laws of a dc shunt motor, neglecting the dynamics of the rotor circuit. The gear is represented by its ratio and its coefficient for friction.

1.3.4 Sensors

The pickups for motor speed and fin deflection are modelled by error terms for set-off and scaling errors.

1.4 Model Interface

1.4.1 Model Inputs

Mnemonic Name	Type	Symbol	Dimension	Meaning
LL	LOGICAL	L_L	-	Device locked if true
DT	REAL	-	s	Communication interval
SIC	REAL	σ_c	rad	Commanded fin deflection
MH	REAL	m_H	Nm	Hinge moment

Tab 1. Inputs to ELACT3

1.4.2 Model Outputs

Mnemonic Name	Type	Symbol	Dimension	Meaning
SI	REAL	σ	rad	Fin position
DSI	REAL	$\dot{\sigma}_M$	rad/s	Motor speed
IC	REAL	i_c	A	Commanded motor current
IM	REAL	i_M	A	Actual motor current
UC	REAL	u_c	V	Motor feeding voltage

Tab 2 Outputs from ELACT3

1.5 Assumptions and Justifications Used for System Model

See Paragraph 10, B-1.

1.6 Mathematical Model

In the following paragraphs, the mathematical model of the electromechanical actuation system will be described. The model variables are listed in Tab 3, the model constants and parameters are listed in Tab 4, and the detailed block diagram of the system is shown in Fig B-3.

The following conventions have been used for notation:

- Constants and parameters: Capital letters
- Variables: Small letters
- Subscript "M" stands for "Motor"
- Subscript "m" stands for "Measured value"
- Subscript "c" stands for "Commanded value"
- "V" indicates "value of limitation for variable v"

Mnemonic Name	Type	Symbol	Dimension	Meaning	Remarks
IC	REAL	i_c	A	Commanded motor current	State variable
SIC	REAL	σ_c	rad	Commanded fin deflection	Input variable
SIM	REAL	σ_m	rad	Measured fin deflection	
DSI	REAL	$\dot{\sigma}_M$	rad/s	Motor speed	State/output var.
DSIM	REAL	$\dot{\sigma}_{M_m}$	rad/s	Measured motor speed	
IM	REAL	i_M	A	Actual motor	State/output var.
UC	REAL	u_c	V	Motor feeding voltage	Output variable
UE	REAL	u_E	V	Motor EMF	
SI	REAL	σ	rad	Actual fin deflection	State/output var.
LL	LOGICAL	L_L	logical	"Device locked"	Input variable
MM	REAL	m_M	Nm	Motor moment	
MH	REAL	m_H	Nm	External hinge moment	Input variable

Tab 3. Model Variables

Mnemonic Name	Type	Symbol	Dimension	Value	Tolerance	Definition by	Meaning
OM	REAL	ω	1/s	185	-	DATA	System design parameter
B	REAL	b	-	3.1	-	DATA	System design parameter
C	REAL	c	-	4.0	-	DATA	System design parameter
KG	REAL	K_G	-	96	-	DATA	Gear ratio
K1	REAL	K_1	As	$b\ \omega^2\ J/C_M$	± 6%	DATA	Controller parameter
K2	REAL	K_2	s	$c/(b\ \omega)$	± 6%	DATA	Controller parameter
K3	REAL	K_3	1/s	ω/b	± 6%	DATA	Controller parameter
K4	REAL	K_4	-	1	± 2%	DATA	$\dot{\sigma}$-sensor, scaling error
K5	REAL	K_5	-	1	± 0.5%	DATA	σ-sensor, scaling error
DELSI	REAL	$\Delta\sigma$	rad	0	± 0.006	DATA	σ-sensor, set off error
SDMAX	REAL	$\hat{\dot{\sigma}}_M$	rad/s	288	± 5%	DATA	Controller, limit
SIMAX	REAL	$\hat{\sigma}$	rad/s	0.349	± 3%	DATA	Limit of commanded fin deflection
RM	REAL	R_M	Ω	5.4	5.4..6.8	DATA	Motor resistance
CE	REAL	C_E	Vs/rad	0.0707	± 10%	DATA	Coefficient of EMF
CM	REAL	C_M	Nm/A	0.0707	± 10%	DATA	Motor constant
IMG	REAL	J	Nms^2	10^{-5}	± 10%	DATA	Moment of inertia
ICMAX	REAL	$\hat{I}_c$	A	6.76	± 5%	DATA	Limit commanded motor current
UCMAX	REAL	$\hat{U}_c$	V	56		DATA	Power amplifier max. output voltage
CR	REAL	C_R	-	0.2	± 50%	DATA	Coefficient of friction
MR	REAL	M_R	Nm	0.02	± 50%	DATA	Friction moment
MHMAX	REAL	$\hat{M}_H$	Nm	$\hat{I}_C\ C_M\ K_G$		DATA	Max. hinge moment

Tab 4. Model Parameters and Constants

1.6.1 Actuator Controller

The differential equation for the PID-controller, taking the gear ratio K_G into account, is as follows:

$$(i_c) = K_1 \cdot K_3 \cdot K_G \quad (\sigma_c - \sigma_m) - K_1 \cdot \dot{\sigma}_{Mm} - K_1 \cdot K_2 \cdot \dot{\sigma}_{Mm}$$

with the limits

$$|K_3 \cdot K_G (c_c - \sigma_m)| < \hat{\dot{\sigma}}_M$$

$$i_c < I_c$$

If L_L = true (device locked): $i_c = 0$.

In the real system $\dot{\sigma}_{Mm}$ is generated by a differentiating network (Ref Fig B-3). For the model, σ_{Mm} is generated from the equilibrium of moments, Ref Paragraph 1.6.3. σ_{Mm} is formed by multiplication with K_4, Ref Paragraph 1.6.4.

1.6.1.1 Computation of Controller Parameters

The overall transfer function of the actuator system, neglecting the dynamics of the motor current circuit, is:

$$\sigma(s) = \frac{1}{\frac{J}{K_1 K_2 C_M} \cdot s^3 + \frac{K_2}{K_3} \cdot s^2 + \frac{s}{K_3} + 1} \cdot \sigma_c(s)$$

Given the transfer function of the desired behavior of the actuator system

$$\sigma(s) = \frac{1}{(\frac{s}{\omega})^3 + c \quad (\frac{s}{\omega})^2 + b \quad (\frac{s}{\omega}) + 1} \cdot \sigma_c \ (s)$$

we get by comparison of coefficients:

$$K_1 = b \cdot \omega^2 \cdot J/C_M$$

$$K_2 = c/(b\ \omega)$$

$$K_3 = \omega/b$$

J is the total moment of inertia of the motor/gear/fin assembly, defined at the motor side of the assembly. This way it is possible to calculate the coefficients of the actuator controller, given the dynamic design parameters ω, b, c.

1.6.2 Power Amplifier

The task of the power amplifier is the transformation of the commanded motor current, as computed by the actuator controller, into real motor current. The bandwidth of the real device is around 600 H_z. This is large compared with the bandwidth of the overall actuator system and can therefore certainly be neglected for the intended purpose of this model.

The steady state equations of the power amplifier are:

$$U_E = C_E \cdot \sigma_M$$

$$U_c = i_c \cdot R_M$$

$$\text{if } |u_c| > \hat{U}_c:\ u_c = U_c \cdot \text{sign}(i_c)$$

$$i_M = (u_c - u_E)/R_M$$

1.6.3 Motor/Gear/Load

Moment, generated by the motor:

$$m_M = C_M \cdot i_M$$

Equilibrium of moments:

$$J\sigma_M = m_M + m_L - C_R \cdot |m_L| \cdot \text{sign}(\sigma_M) - M_R \cdot \text{sign}(\dot{\sigma}_M)$$

$$m_L = m_H / K_G$$

$$\dot{\sigma}_M = \int \sigma_M \, dt + \dot{\sigma}_{Mo}$$

$$= \int (\dot{\sigma}_M / K_G) dt + \sigma_o$$

If L_L = true (device locked): $\dot{\sigma}_M = \dot{\sigma}_{Mo} = 0$

$$\sigma = \sigma_o = \Delta\sigma$$

1.6.4 Sensors

The feedback values for the controller, motor speed and actual fin deflection, are measured by a tacho generator and a potentiometer. The tacho generator is represented by a scaling error:

$$\dot{\sigma}_{Mm} = K_4 \cdot \dot{\sigma}_M$$

$$\sigma_{Mm} = K_4 \cdot \sigma_M$$

The position pickup is represented by set-off and scaling error:

$$\sigma m = K_5 \cdot \sigma - \Delta\sigma$$

2. IMPLEMENTATION DESCRIPTION

The model is implemented digitally as a single subroutine named ELACT3, written in Standard ANSI FORTRAN IV.

2.1 List of Computer Variables

The computer variables, apart from temporary variables, are listed in the following tables:

Input Variables	Tab 1
Output Variables	Tab 2
Model Variables	Tab 3
Model Parameters and Constants	Tab 4
Implementation Variables	Tab 5
Implementation Parameters	Tab 6

Mnemonic	Type	Definition by	Meaning
LL	LOGICAL	input	true = initialization false = integration identical to "device locked"
IERR	INTEGER	condition within ELACT 3	Error indicator
IR	INTEGER	ELACT 3	Counter for integration control

Tab 5. Implementation Variables

Mnemonic	Type	Definition by	Meaning
DT	REAL	input	Communication interval
DTRM	REAL	ELACT3	Internal integration step
DTRMO	REAL	DATA	Upper bound for integration step
IRM	INTEGER	ELACT3	Number of integration steps per communication interval

TAB 6. Implementation Parameters

2.2 Processing Methods

2.2.1 Function Allocation

The functions are performed as described in Paragraph 1.6. There is a clear correspondence between the implemented code and those functions.

2.2.2 Integration Method

For integration of the differential equations, the Euler method is used. Because of the dynamics of the modeled device, usually a smaller integration step, DTRM, than the communication interval DT has to be used. During initilization, ELACT3 computes a suitable integration step size, assuring that it is an integral fraction of the communication interval not greater than DTRMO:

$$DTRM = DT/n < DTRMO$$

with n = suitable integer:

$$n = \text{int}\ [DT/DTRMO + 0.5]$$

At every call to ELACT3, n integration steps are performed. This is controlled through the variable IR and parameter IRM (= n).

2.2.3 Model Parameters and Constants

Model parameters and constants are defined by DATA-statements within ELACT3. They cannot be altered by calling the subroutine. There is no stochastic variation of model parameters implemented.

2.2.4 Initialization

The first call to ELACT3 must be an initialization call. This is performed by calling it with the input variable LL = TRUE . State variables are set to initial conditions, the controller coefficients are computed as well as the internal integration step size, and the error indicator IERR is reset. Initialization calls can be repeated.

2.2.5 Error Detection

The error indication is set to 1 if the external hinge moment exceeds 80% of the maximum hinge moment defined as

$$\hat{M}_H = \hat{T}_c \cdot C_M \cdot K_G$$

2.3 Required Program Library Elements

Apart from standard run time, library routine ELACT3 does not call any subroutine.

2.4 Required Computer Resources

2.4.1 Required Peripheral Equipment and Data Files

The program does not need any peripheral equipment or data files.

2.4.2 Memory Requirements

Subroutine ELACT3 occupies 152 60-bit-words of main memory on CDC 6600.

2.4.3 Running Time

The execution time per call to ELACT3 with 5 internal integration steps on a CDC 66000 Computer is 480 sec.

3. SYSTEM MODEL VERIFICATION

3.1 Criteria for Model Verification

The following criteria have been used for model verification:

(a) Model responses to step input functions should be as predicted by theory and as expected due to an expert's understanding of the system (plausibility).

(b) Insensitivity of model behavior with respect to digital integration parameters.

3.2 Methods Used for Model Verification

The methodology used for model verification is summarized in Tab 7.

3.2.1 Dynamic Model Behavior Test

Objective was to assure that the model behaves dynamically as expected from theory and from an experts' experience.

3.2.2 Sensitivity with Respect to Implementation Parameters

Objective was to assure that model behavior does not depend on implementation particularities and parameters.

3.3 Data Bases used for Verification

3.3.1 Dynamics Model Behavior Tests

3.3.1.1 Reference Data Base

3.3.1.1.1 Model Behavior without Mechanical Friction

Neglecting mechanical friction ($M_R = C_R = 0$), the following tests have been performed.

Type of Verification Test	Data Bases		Verification Test		
	Model Generated Data Base	Reference Data Base	Method	Criteria	Results
Dynamic Model Behavior	Various Model Responses on Step Inputs Plots of Critical Variables	Linear Domain: Analytical Solution of Time Response of 3rd order Transfer Function Nonlinear Domain: Precalculation of Typical Response Quantities Plots of Critical Variables	Visual Inspection by Experts Plot Overlays	Reasonable Coincidence from an Engineering viewpoint (Expert Opinion) No anomalies or Unexplainable Effects	o.k. o.k.
Sensitivity wrt. Implementation Parameters	as (1)	as (1)	as (1)	Reasonable Insensitivity wrt. Integration Step Size Δt	o.k. for $\Delta t < 0.001$ sec

Tab 7. Methodology of Model Verification

Test A: Operation in nonlinear domain.
Rectangular input, $\sigma_c = \pm 0.26$ rad, no external load, change in σ_c every 400 ms.
The transition slope of σ is required to correspond to maximum motor speed divided by gear ratio:

$$\dot{\sigma}_{Mmax} = 288/96 = 3 \text{ rad/sec}$$

Therefore, neglecting the dynamics of the system, the transition time of σ between steady state values should be approximately

$$t = \Delta\sigma/\dot{\sigma}_{Max} = 0.52/3 = 0.173 \text{ sec}$$

The motor current i_M is required to overcome mechanical inertia during the acceleration/deceleration phases. As soon as motor speed $\dot{\sigma}_M$ reaches its saturation value of 288 rad/sec or the value zero, motor current i_M has to go to zero.

Test B: Operation at the limits of linearity.
Rectangular input, $\sigma_c = \pm 0.07$ rad, no external load, change in σ_c every 400 ms.

Test C: Operation in the linear domain.
Rectangular input, $\sigma_c = \pm 0.017$ rad, no external load, change in σ_c every 400 ms.

The required model response can be calculated analytically, using the desired transfer function (see Paragraph 1.4.1.1):

$$(s) = \frac{1}{\left(\frac{s}{\omega}\right)^3 + b\cdot\left(\frac{s}{\omega}\right)^2 + c\cdot\frac{s}{\omega} + 1}\,\sigma_c\,(s)$$

with parameters ω, b, c, according to Tab 4.

For this small step input, no one of the limitation values is reached and the implemented model should reproduce the analytical solution with high accuracy.

Test D: Nonlinear operation with external hinge moment. Rectangular input, $\sigma_c = \pm 0.26$ rad, $m_H = -15$ Nm, change in σ_c every 400 ms.

As long as neither currents nor voltages reach their saturation values, the time histories of σ and $\dot{\sigma}_M$ should be the same as with Test A. Also i_M should show the behavior as in Test A, but with a constant offset value which is necessary to compensate for the external load. This offset value should be

$$i_M = \frac{-m_H}{C_M \bullet K_G} = \frac{15}{0.0707 \bullet 96} = 2.21 \text{ A}$$

3.3.1.1.2 Model Behavior Including Mechanical Function

Above Tests A through D have to be repeated with mechanical friction, i.e., $M_R = 0.02$ Nm and $C_R = 0.2$. In test cases without external load, the behavior of σ and $\dot{\sigma}_M$ should be identical to the results without friction. During non-steady states, motor current should be increased by

$$\Delta i_M = \frac{M_R}{C_M} \bullet \text{sign}(\dot{\sigma}_M) = \frac{0.02}{0.0707} \bullet \text{sign}(\dot{\sigma}_M) = 0.28 \bullet \text{sign}(\dot{\sigma}_M)$$

In test case D (with external load) the motor current is increased by

$$\Delta i_M = \frac{M_R + m_H \bullet C_R/K_G}{C_M} \text{ sign}(\dot{\sigma}_M) = \frac{0.02+15 \bullet 0.2/96}{0.0707} \text{ sign}(\dot{\sigma}_M) = 0.725 \bullet \text{sign}(\sigma_M$$

3.3.1.2 Model Generated Data Base

The integration step size used was 1 ms. The data is recorded in form of plots in a format similar to the one obtained from a multi-channel recorder. The first line shows the input variable σ_c, the following ones the critical variables defined in Section 10, B-1.

3.3.1.2.1 Model Behavior without Mechanical Friction

Test A: Operation in nonlinear domain, Fig B-4 (At end of CLIMB Level 2).

Test B: Operation at the limits of linearity, Fig B-5.

Test C: Operation in the linear domain, Fig B-6.

Test D: Nonlinear Operation with external hinge moments, Fig B-7.

3.3.1.2.2 Model Behavior Including Mechanical Friction

Test A: Operation in the nonlinear domain, including mechanical friction, i.e., $M_R = 0.02$ and $C_R = 0.02$, Fig B-8.

3.3.2 Sensitivity with Respect to Implementation Parameters

Using Test C as test case, the sensitivity of model response with respect to integration step size DTRM has been investigated.

3.3.2.1 Reference Data Base

Test C: Operation in the linear domain, without mechanical friction, analytical solution, Fig B-9.

3.3.2.2 Model Generated Data Base

Test C: Operation in the linear domain, without mechanical friction, results of runs with DTRM = 0.0005 s, 0.001 s and 0.002 s, Fig B-10.

Test C: As above, but with DTRM = 0.002 s, 0.003 s and 0.004 s, Fig B-11.

Using the same test case, the model, including mechanical friction was exercised. Fig B-12 shows the results in the case of DTRM = 0.001 s, Fig B-13 with DTRM = 0.002 s.

3.4 Results from Model Verification Efforts

3.4.1 Dynamic Model Behavior Tests

Criterion was reasonably coincidence between the two data bases from an engineering expert's viewpoint. Plot overlays have been used, but no quantitative measures for goodness of fit were used. In addition, the model generated data did not expose any anomalies or unexplainable effects.

The results of the comparison are summarized in the following:

3.4.1.1 Model Behavior Without Mechanical Friction

Test A: Operation in nonlinear domain, Fig B-4.
The transition slope of , as retrieved from the plot, is:

$$\dot{\sigma}_{Mmax} = 0.52/0.172 = 3.02 \text{ rad/sec}$$

which matches well the theoretical value of 3.0 rad/s. The steady state value for $\dot{\sigma}_M$ in the plot is 290 rad/s, as compared with theoretically 288 rad/s. Motor current i_M behaves as expected, it is proportional to σ_M. The magnitude cannot easily be verified at this stage.

Test B: Operation at the limits of linearity, Fig B-5. $\dot{\sigma}_M$ is just reaching its saturation value. Model performance does not show any anomalies or unexplainable effects.

Test C: Operation in the linear domain, comparison with analytical solution of Fig B-9.

Model behavior, Fig B-6, matches very well the analytical solution. The model shows a little, unsignificantly higher overshoot. Motor current cannot be compared since it has not been calculated in the case of the analytical solution.

Test D: Nonlinear operation with external hinge moments, Fig B-7. The curves are identical with those of Fig B-4 (Test A) with the only difference that i_M shows an offset of 2.25 A in order to compensate for the external hinge moment:

$$m_H = i_M \cdot C_M \cdot K_G = 2.25 \cdot 0.0707 \cdot 96 = 15.27 \text{ Nm}$$

as compared with the applied value of 15.0 Nm.

As long as the motor current i_M is not saturated, external loads do not have any effect on the dynamic behavior of the actuation system.

3.4.1.2 Model Behavior Including Mechanical Friction

Test A: Operation in the nonlinear domain, Fig B-8. Motor current is now required not just during the acceleration/deceleration phases, but in to overcome the friction moment M_R.

The current during constant motor speed, is 0.3 A. This corresponds to a motor moment of

$$m_M = C_M \cdot i_M = 0.0707 \cdot 0.3 = 0.02121 \text{ Nm}$$

which matches very well the supposed value of $M_R = 0.020$. The transition slope, is practically the same as in the case without friction.

However, when the system is approaching a steady state, i.e., $\dot{\delta} \doteq 0$, a limit cycle is generated, having characteristics that are very sensitive to the particular implementation. In the present case of a digital implementation, frequency and amplitude of the limit cycle are highly dependent on the particular integration step size chosen. This means that the proposed model of mechanical friction is not reasonable.

Since the influence of friction on the dynamic behavior of the actuator system is negligible, it is recommended that the model be operated with $M_R = C_F = 0$.
When the model is used within an autopilot loop, steady states will practically never be reached and the limit cycle will probably never be excited. Therefore, if one wishes to derive an estimate of power consumption of the actuator system, the model could be used by taking friction into account.

3.4.2 Sensitivity with Respect to Integration Step Size

The criterion was that the model response should be reasonably insensitive with respect to step size, DTRM, as judged by an expert engineer.

Fig B-10 shows that there is no significant difference between the responses with DTRM = 0.0005 s and DTRM = 0.001 s, whereas a clearly visible divergence can be stated for DTRM = 0.002 s.

Fig B-11 shows that already DTRM = 0.003 s causes the model to become unstable.

Therefore, the value DTRM = 0.001 s appears to be a reasonable choice.

Using the same test case, the model including mechanical friction was exercised. Fig B-12 shows the limit cycle in the case of DTRM = 0.001 s. Fig B-13 shows the results obtained with DTRM = 0.002 s. It demonstrates the sensitivity of the characteristics of the limit cycle with respect to integration step size. This confirms the statement made in Paragraph 2.3.2.2 about the questionable applicability of the model including mechanical friction terms.

4. VALIDATION OF SYSTEM MODEL'S STOCHASTIC COMPONENTS

The present model was basically a deterministic nature in the sense that it does not contain any internal sources of noise. In the real system, most of the parameters describing the model are subject to random variations because of component tolerance, ref Tab 4. This could be taken into account in the model by random variation of relevant parameters prior to each model run. However, this feature has not been implemented in the present model.

5. VALIDATION AGAINST OTHER EXISTING MODELS

No Validation against other models have been executed.

6. SUBSYSTEM CHARACTERIZATION AND BRIEF DESCRIPTION OF SUBSYSTEM MODELS

Not applicable.

7. BENCHMARK TEST CASE

7.1 Description of Benchmark Test Cases

The test cases used for verification of the dynamic behavior of the model as described in Paragraph 3.2.2.1 should be used as benchmark test cases.

7.2 Input Data

Input is a rectangular variation of the variable σ_c (SIC) as described in Paragraph 1.4.1.

7.3 Output Data

The output data is recorded in Figs B-4 through B-8, as described in Paragraph 1.4.2. For the benchmark tests, the data base described in Paragraph 1.6 becomes the reference data base.

7.4 Criteria for Acceptability

Same as for verification tests, described in Paragraph 3.

8. COMPUTER PROGRAM

8.1 User Instructions

The model is implemented as a single subroutine called FLACT3. Its calling sequence can readily be inferred from the listing in the appendix.

CAUTION: The contents of the state variables SI, DSI, IC must not be altered between calls!

For initiatization see Paragraph 2.2.4 and for error detection see Paragraph 2.2.5.

8.2 Computer Listings

The complete source listing is not included.

9. PROGRAM VERIFICATION

9.1 Criteria for Verification

The following criteria have been used for program verification:

(a) Correct implementation of the mathematical model of Paragraph 1.6.

(b) Program code in compliance with Programming Standards.

(c) Portability of the program code.

9.2 Methods used for Program Verification

The methodology used for model verification is summarized in Tab 8. Program verification has been performed by computer code analysis.

Objective was to assure that the model has been correctly translated into portable computer code.

Three types of analyses have been performed:

(a) Source Code Inspection

(b) Inspection of Cross Reference Listing

(c) Source Code Compilation in "ANSI-Mode."

Type of Verification Test	Data Bases		Verification Test		
	Model Generated Data Base	Reference Data Base	Method	Criteria	Results
(a) Source Code Inspection	Computer Listing	Mathematical Model, Programming Standards	Visual Inspection	Computer Listing in Compliance with Mathematical Model and Programming Standards	o.k.
(b) Inspection of CRL	Computer Listing	Programming Standards	Visual Inspection	Compliance with Requirements List Paragraph 10, B-5	o.k.
(c) Portability	Computer Source Program	ANSI-FORTRAN Standard	Compiler Run (Automatic Diagnostics)	Absence of Diagnostic Messages	o.k.

Tab 8. Methodology of Program Verification

9.3 Data Bases used for Program Verification

9.3.1 Source Code Inspection

Model Generated Data Base: Computer Listing

Reference Data Base: Description of the mathematical model of Paragraph 1.6 plus Programming Standards, Paragraph 10, B-3.

9.3.2 Inspection of Cross Reference Listing

Model Generated Data Base: Computer Listing

Reference Data Base: Requirements List of Paragraph 10, B-5.

9.3.3 Source Code Compilation in "ANSI-Mode"

Model Generated Data Base: Diagnostic messages by compilation run in "ANSI-mode", Section 10, B-6.

Reference Data Base: ANSI-FORTRAN Standard as implemented in the compiler of Section 10, B-6.

9.4 Results from Program Verification Efforts

9.4.1 Source Code Inspection

Formal source code inspection was performed by the Quality Assurance Dept of MBB-UA. Criterion required the compliance of the computer listing with the mathematical model as described in Paragraph 1.6 as well as with the Programming Standards of Paragraph 10, B-3. This criterion was fulfilled.

9.4.2 Inspection of the Cross Reference Listing

The cross reference listing, as generated by the compiler, was inspected by the Quality Assurance Dept of MBB. It fulfilled the criterion to comply with the requirements list of Section 10, B-5.

9.4.3 Source Code Compilation in "ANSI-Mode"

Inspection of the source code by the Quality Assurance Dept of MBB has shown that it is in compliance with Paragraph 10, B-4. Compilation of the program in the "ANSI"-mode did not result in any statement flagged as "non-ANSI".

10. APPLICABLE DOCUMENTS

B-1 MODEL DOCUMENTATION, ELACT3
ELECTRICAL ACTUATION SYSTEM, CLIMB 2 Level 1
April 1984

B-2 REPORT on ACCEPTANCE TESTS of the REAL ACTUATOR

B-3 Richtlinien zur Programmerstellung (Programming Standards)
Internal MBB-Paper (in German)

B-4 Sandra Summers, Jean Fox
Writing Machine Independent FORTRAN
Software World Vol 9, No 2

B-5 Checklist for Inspection of Cross Reference Listing, as Generated by CDC-FORTRAN-Compilers.
Internal MBB-Paper

B-6 CDC-FORTRAN-EXTENDED VERSION 4
REFERENCE MANUAL 60997800

B-7 CDC-NOS VERSION 1
REFERENCE MANUAL 60435400

CLIMB LEVEL 3 EXAMPLE

1. <u>REAL WORLD SUBSYSTEM DATA</u>

1.1 <u>Identification of Subsystem</u>

Electrical actuator system to drive the fins of the experimental EMS missile system.

1.2 <u>List of Variables for Which Measured Data Exist</u>

Symbol		Variable
Model	Real System	
σ_c	σ_{RC}	commanded fin deflection
σ	σ_{RM}	actual fin deflection
$\dot{\sigma}_M$	$\dot{\sigma}_{TM}$	motor speed
i_M	i_{AM}	actual motor current

1.3 Data in Hard Copy Form

Attached to the end of CLIMB Level 3 in Figs B-14 through B-16.

2. EXPERIMENT. ST ENVIRONMENT

2.1 Scenario Used to Excite Subsystem

Laboratory bench tests have been performed with the real actuator system by applying step inputs. Test conditions A through C as identified Paragraph 8, B-9, have been used. Since no device was available which was capable to apply a defined moment on the hinge under dynamic conditions, Test D could not be made.

2.2 Description of Test Experiment

The test experiment is outlined in Fig B-17. The real actuator system consists of two subassemblies:

(a) The actuator electronics, including the actuator controller plus the power amplifier

(b) The mechanical parts: DC-motor, gear and sensors

Three power supplies were used to feed the actuator electronics:

- 2 each Hewlett Packard HP 6012A,
  providing power to the power amplifier.
  Voltage setting: ± 56 V, precision ± 0.5 V
  Current limitation value: 10 A, precision ± 1.0 A

- 1 Dual Power Supply Hewlett Packard HP 6227 B,
  providing power to the actuator controller.
  Voltage setting: ± 15 V, precision ± 0.5 V
  Current limitation: 0.2 A, precision ± 5 mA

The input step function to the actuator system was provided by an EXACT Function Generator, Type 255. The control setting was:

- Output:	according to desired square wave amplitude, scale factor 28.65 V/rad for actuator input
- Frequency:	4 Hz
- Waveform:	rectangular

The variables were recorded on a Gould Brush 4 Channel Recorder, Type 2400.

Scale factors and settings were as follows:

- Paper speed: 250 mm/s
- Recordings:

Channel No	Variable	Scale Factor	Channel Setting f.s. = 25 lines
1	σ_{RC}	0.5 V/deg = 28.65 V/rad	9.0 V = 0.312 rad
2	σ_{RM}	0.5 V/deg = 28.65 V/rad	9.0 V = 0.312 rad
3	$\dot{\sigma}_{TM}$	11.94 mV/rad s^{-1}	6.0 V = 500 rad/s
4	i_{AM}	1 V/A	6.25 V = 6.25 A

The points where above variables have been probed are depicted in Fig B-18.

The recording format corresponds to the one used in the model generated data base for verification, see Paragraph 8, B-9, in order to facilitate comparisons by plot overlays.

3. METHODS AND TECHNIQUES USED IN COLLECTING REAL WORLD DATA

3.1 Data Collection Methods

Laboratory bench test with real actuator system, step input applied using a square wave generator. Output recorded by a Brush 4-channel recorder.

3.2 Error Sources

3.2.1 Input Measurements

The apparent rise time on the input step function is a function of the recorder (bandwidth approx 50 Hz) and not of the generator. This has been verified by using an electronic scope. The accuracy of the recordings is 0.7 percent of full scale.

3.2.2 Output Measurements

The same conditions apply.

3.3 Input/Output Data Analysis

3.3.1 Input Data

No analysis performed.

3.3.2 Output Data

No analysis performed.

3.4 Contact for Further Information on Measured Data

Person:	Rudolf Merz
Address:	Messerschmitt-Bolkow-Blohm GmbH Abteilung AE132 Postfach 801149 D-8000 Munchen 80 W-Germany
Phone No.:	089-60006536

4. VALIDATION APPROACH

4.1 Criteria for Validation

The model responses to various step inputs should match reasonably well the real world data that has been generated under the same conditions. This is to be judged by expert engineers that are experienced in actuator design and missile system simulation. No quantitative measures for goodness of fit have been used.

4.2 Validation Methods Used

The method of comparison of the two data bases was plot overlays and experts' judgment.

4.3 Data Bases Used for Validation

4.3.1 Reference Data Base

Reference data base is the real world data as shown in Figs B-14 through B-16. The test cases correspond to the ones described in Paragraph 8, B-9.

Test A: Operation in the nonlinear domain, Fig B-14.

Rectangular input $\sigma_c = \pm 0.26$ rad, no external load.

The first channel shows the step input. The deviation from an ideal step is due to the limited bandwidth of the recorder (Roughly 50 Hz).

The second channel shows actual fin deflection. The slope of the ramp is

$$\dot{\sigma}_{Mmax} = 0.52/0.168 = 3.095 \text{ rad/sec}$$

The third channel records the output of the tacho generator which serves as the sensor for motor speed. The measured value shows a ripple which is due to the cogging effects of the motor. The frequency of the ripple is:

10 periods/(21 mm: 200 mm/s) = 95 Hz,

Motor speed (revolutions per second) is:

3.095 • 96 = 297.1 rad/s = 47.3 Hz,

which matches ideally taking into account that the motor has 2 pairs of poles.

The fourth channel, motor current, is showing a large ripple which is induced by motor effects such as cogging, commutation, etc., as explained above.

Test B: Operation at the limits of linearity, Fig B-15.

Rectangular input $\sigma_c = \pm 0.07$ rad, no external load.

Test C: Operation in the linear domain, Fig B-16.

Rectangular input $\sigma_c = \pm 0.017$ rad, no external load.

4.3.2 Model Generated Data Base

The model generated data base used for validation is documented in Paragraph 8, B-9.

4.4 Validation Results

Test A: Operation in the linear domain.
Model: See B-9, Paragraph 8.
Real System: Fig B-14
The slope of the ramp $\dot{\sigma}_{Mmax}$ differs by 2.5%. The ripple on top of the variable $\dot{\sigma}_{TM}$ in Fig B-14 is not present in the case of the model because the causing motor effects like commutation, cogging, etc. are not represented in the model.

In the fourth channel, motor current shows the greatest difference, whereas, the dynamic behavior during the acceleration/deceleration phases matches reasonably well. a A large ripple that shows up is induced by motor effects such as cogging, commutation, etc. that do not exist in the case of the model since motor effects are not included.

Test B: Operation at the limits of linearity
Model: See B-9, Paragraph 8.
Real System: Fig B-15
As far as σ and $\dot{\sigma}_M$ are concerned, the model is showing a somewhat higher overshoot. The correspondence of motor currents is not very good due to the effects discussed above.

<u>Test C:</u> Operation in the linear domain
Model: See B-9, Paragraph 8.
Real System: Fig B-16
Again, for c and $\dot{\sigma}_M$ the match is very good with the model showing a little higher overshoot.

A summary of validation results has been given in Paragraph 8 of B-8.

4.5 Model Changes Due to Validation Effort

None.

5. COMPUTERIZED EXCITATION METHODS USED

5.1 Excitation of Real World System

The experiment used is described in Paragraph 2. No computer was involved.

5.2 Excitation of the Model

The model was executed by a main program calling subroutine ELACT3. The different test cases A through C were implemented by subsequent manual changes.

6. USER INSTRUCTIONS FOR TEST SET UP

No special explanations necessary, obvious from description of test set up.

7. BENCHMARK FOR TEST SET UP

7.1 Description of Benchmark Test Cases

The test cases used for the validation experiment as described in Paragraph 2.1 and 4.3.1 should be used as benchmark test cases.

7.2 Initial Conditions for Test Set Up

Ref Paragraph 6. The EXACT Function Generator is set up to generate square waves with a frequency of 4 Hz and various amplitudes corresponding to the test cases A through C. Scale factor is 28.65 V/rad.

7.3 Results for Critical Variables

Ref Paragraph 4.3.1 and Figs B-14 through B-16.

7.4 Criteria for Acceptability of Benchmark Results

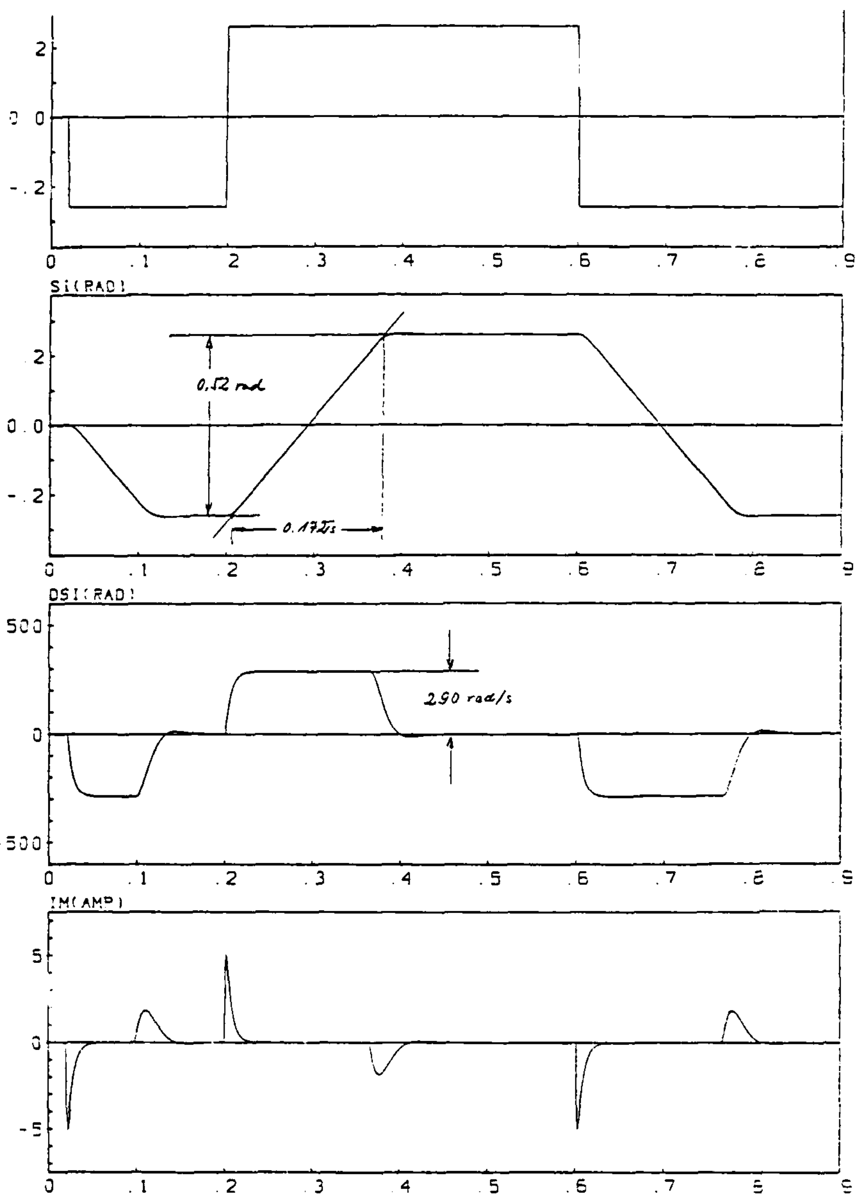
Reasonable correspondence with data Figs B-14 through B-16, judged from an engineer's point of view.

Method of comparison: Plot overlays.

8. APPLICABLE DOCUMENTS

B-8 MODEL DOCUMENTATION, ELACT3, ELECTRICAL ACTUATION SYSTEM, CLIMB
Level 1
April 1984

B-9 dto., CLIMB Level 2
April 1984



Time-sec

Fig B-4. Nonlinear Domain

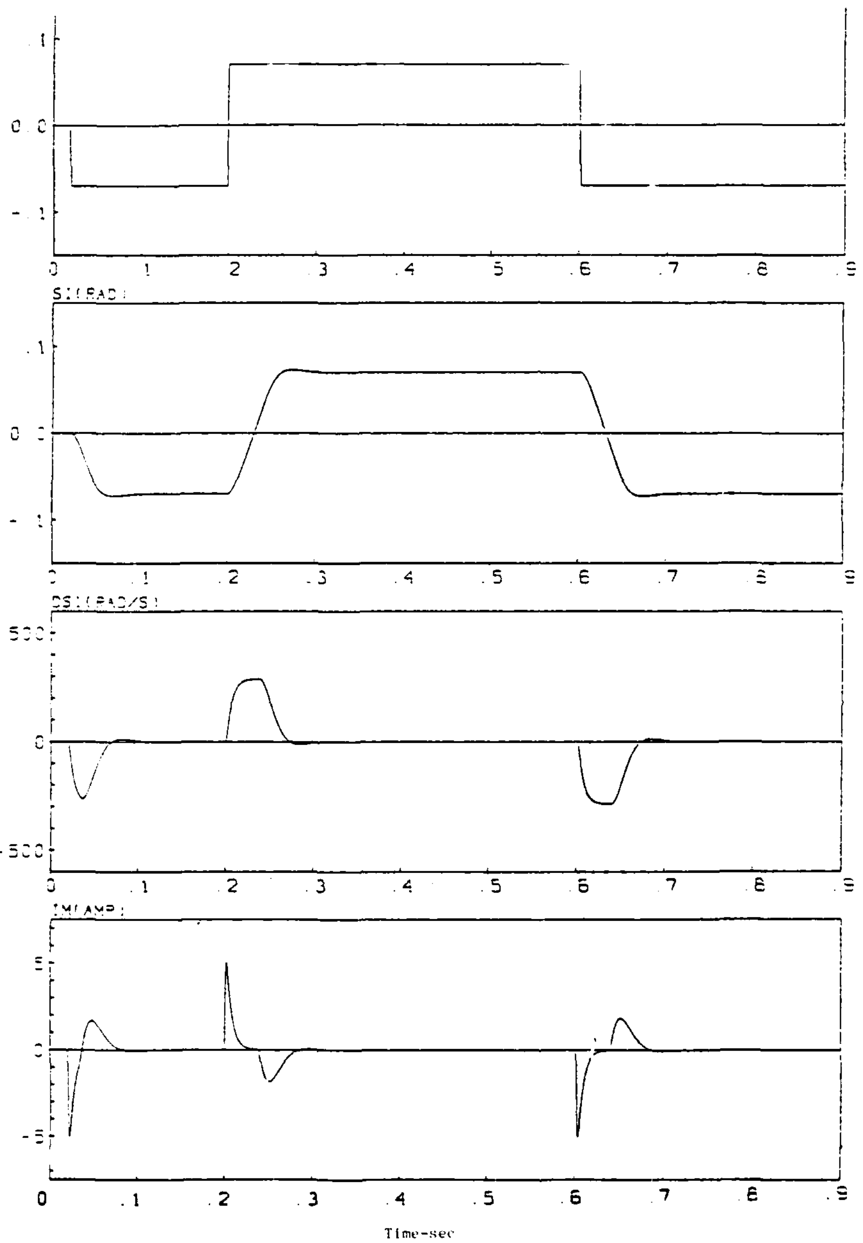
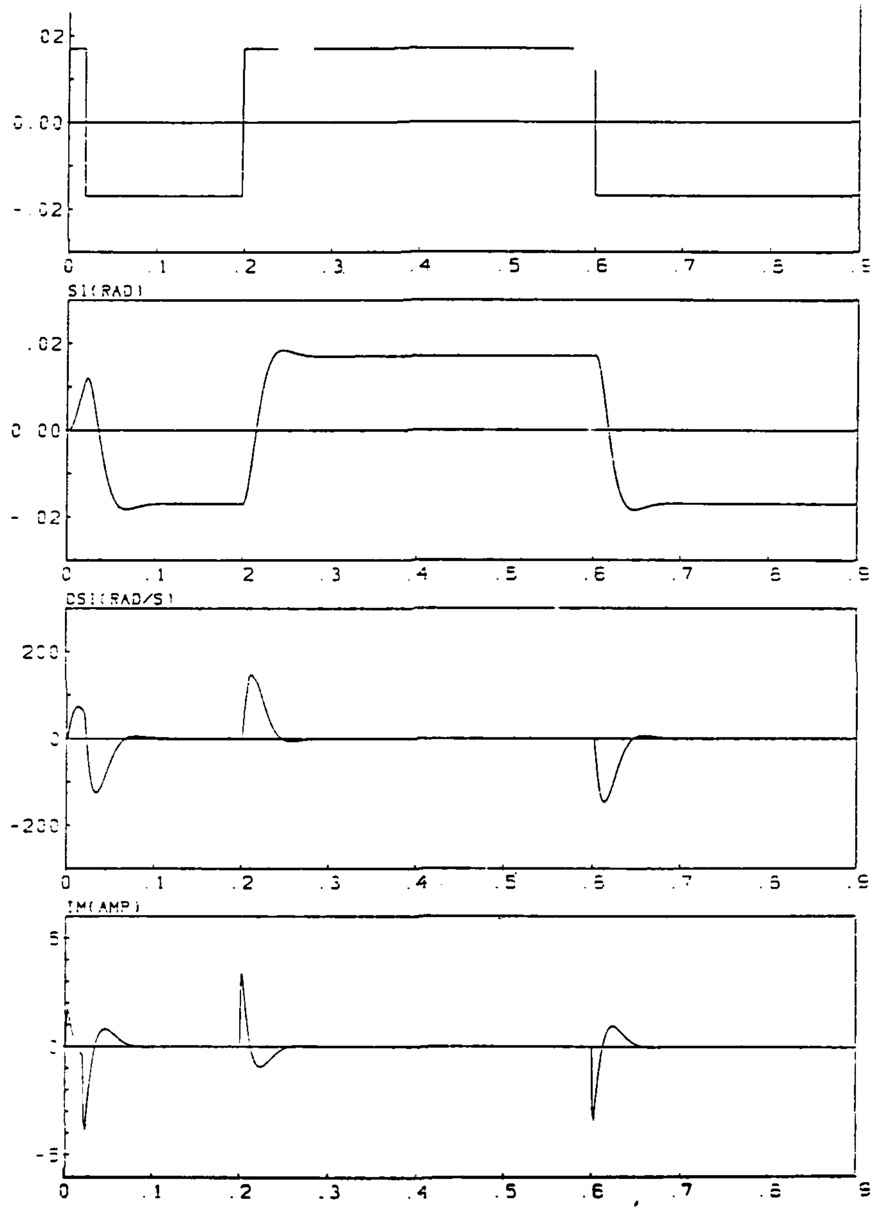


Fig B-5. Limits of Linearity



Time-sec

Fig B-6. Linear Domain

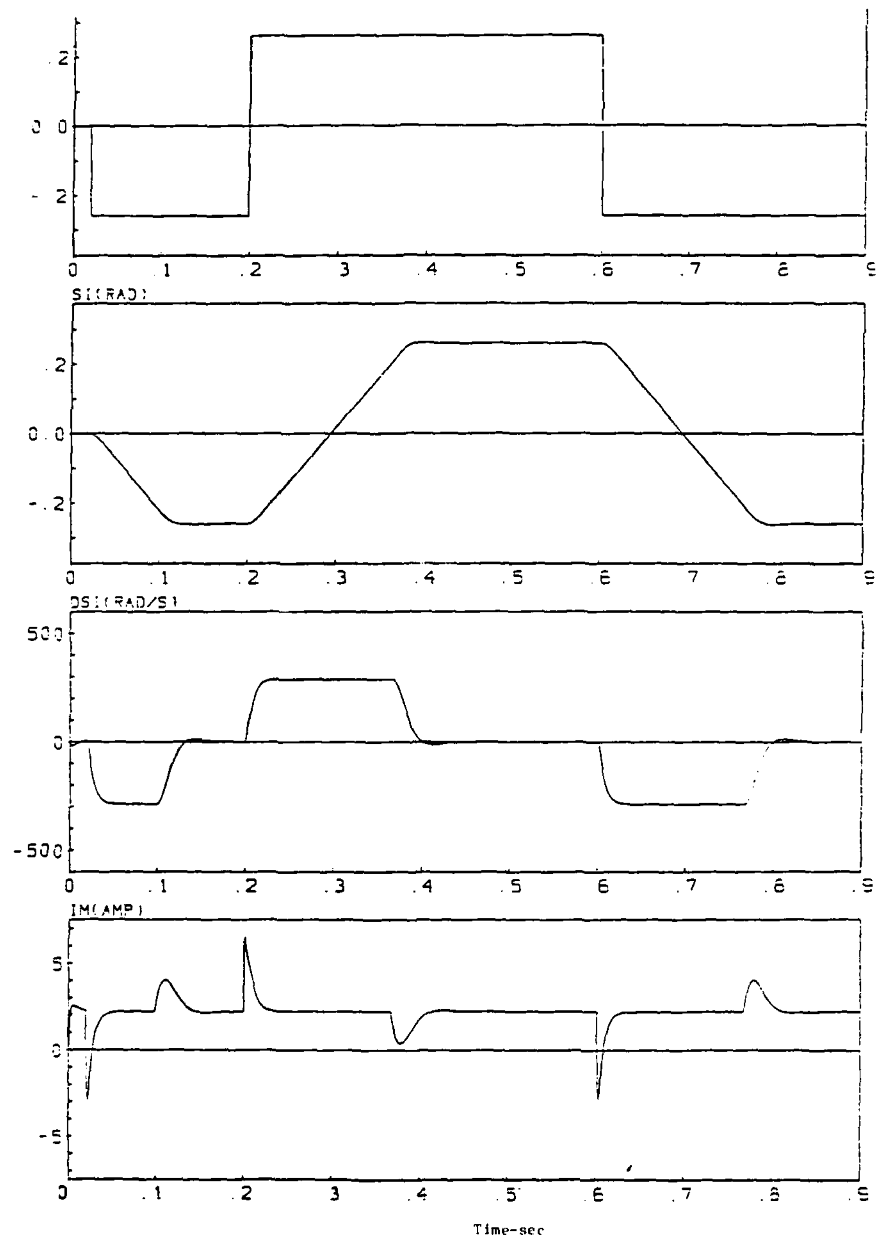
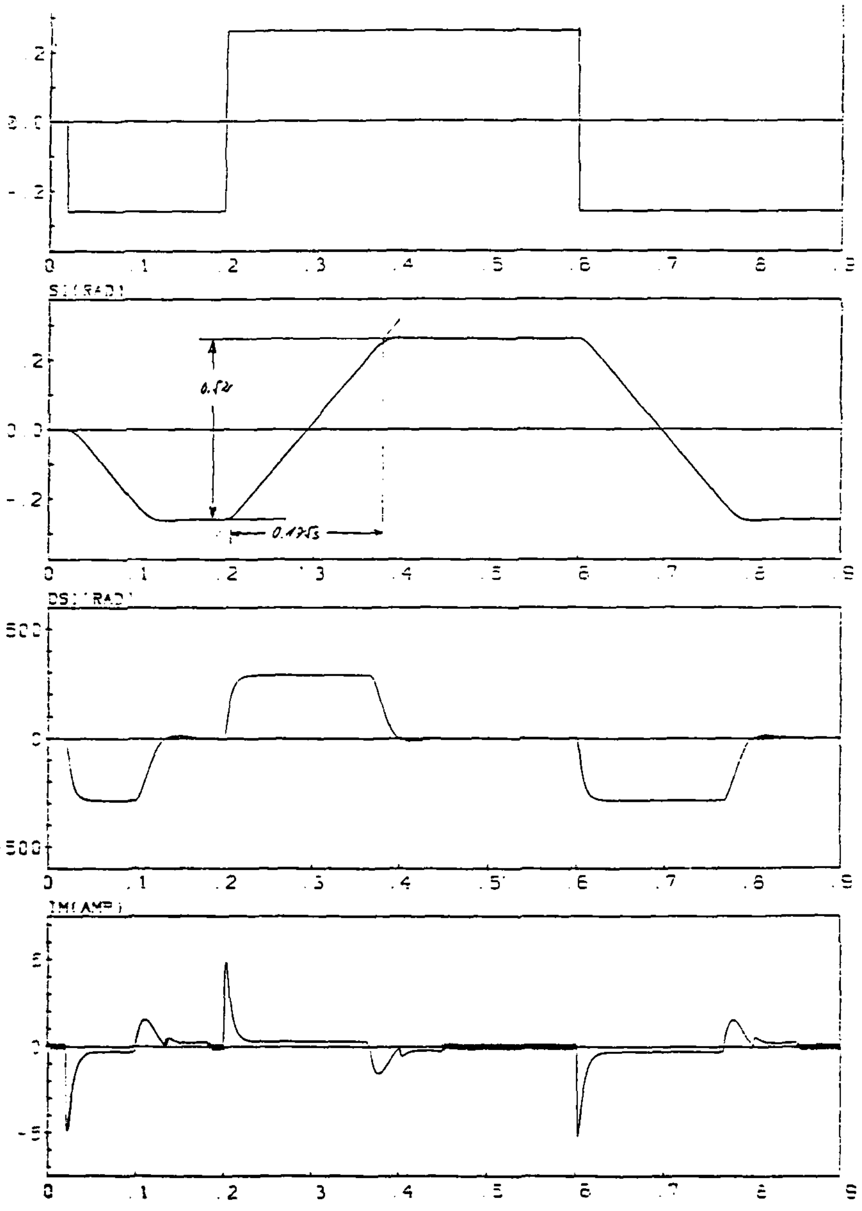
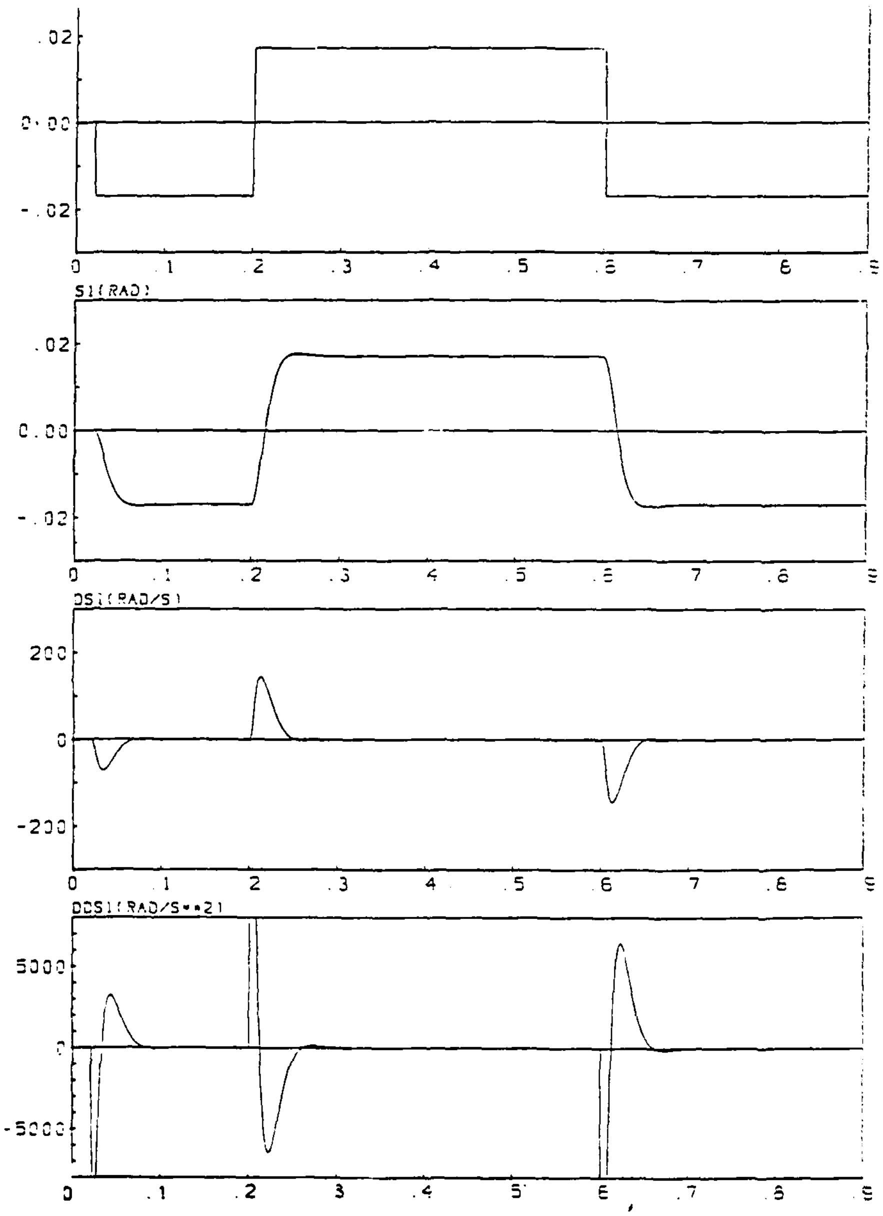


Fig B-7. Nonlinear Operation With External Hinge Moments



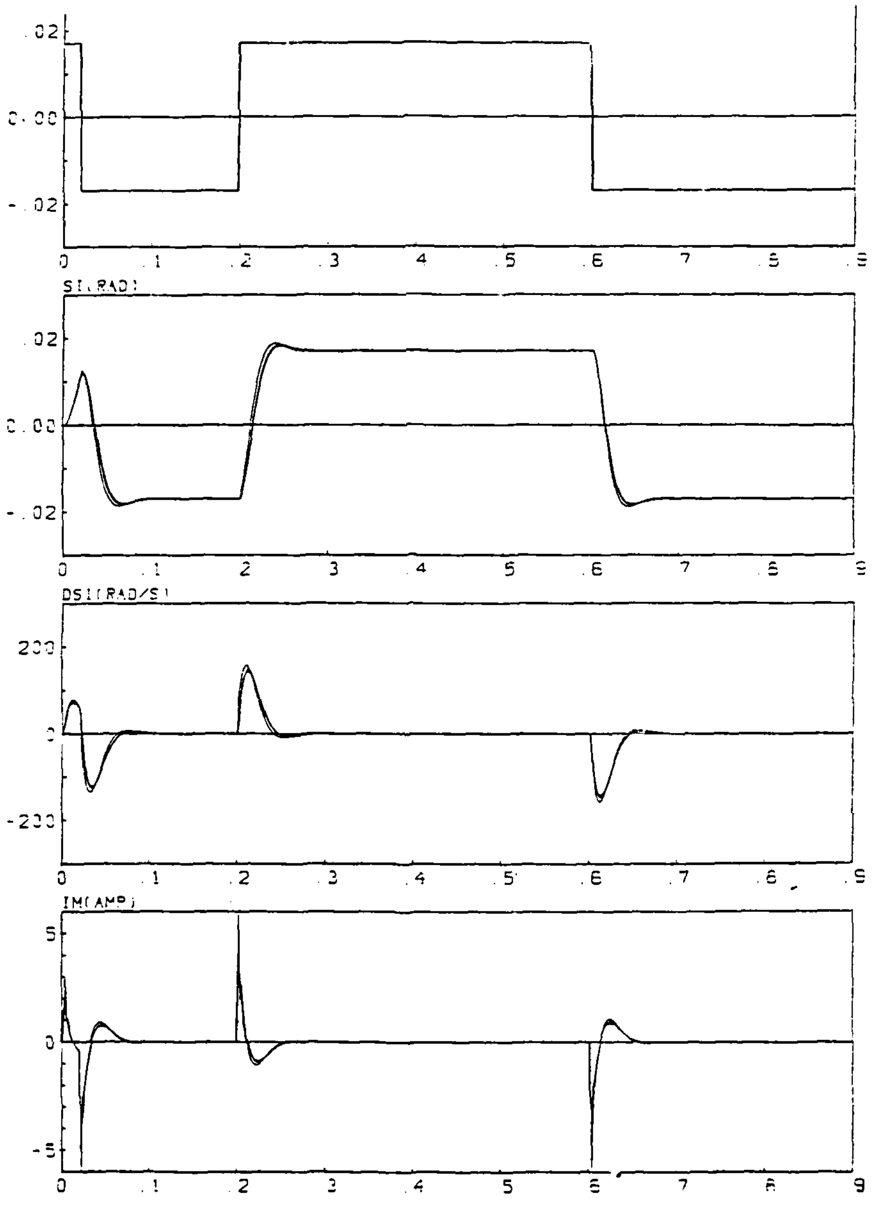
Time-sec

Fig B-8. Nonlinear Operation With Mechanical Friction



Time-sec

Fig B-9. Linear Operation With Mechanical Friction



Time-sec

Fig B-10. Linear Operation Without Friction

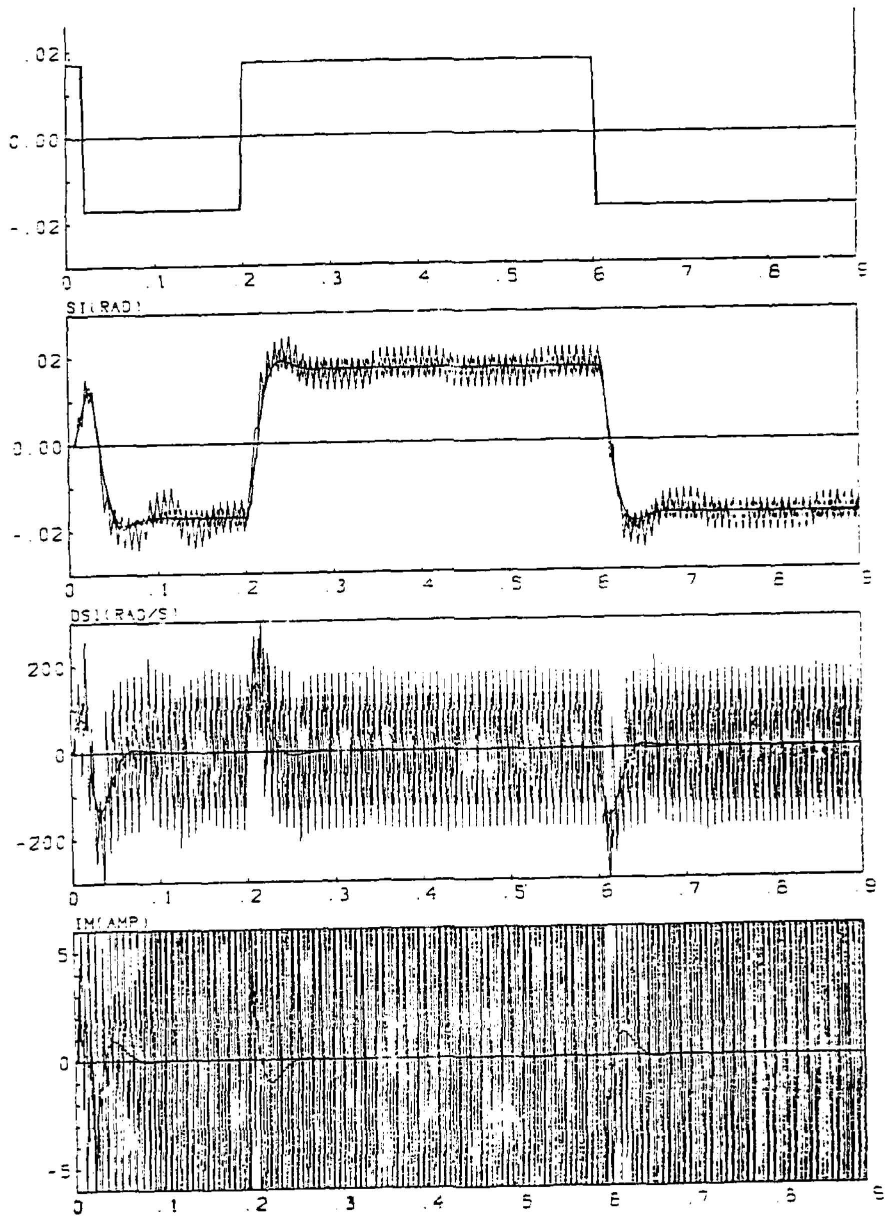
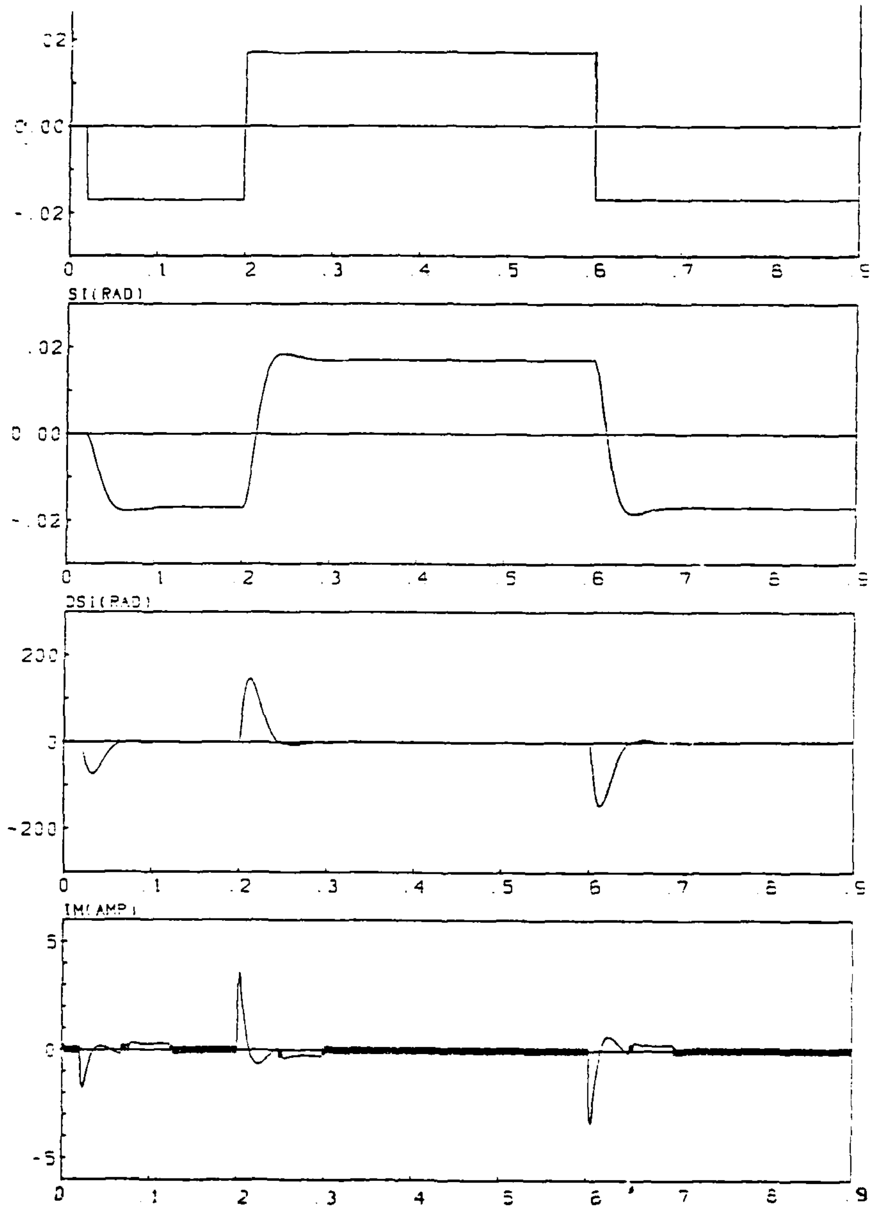
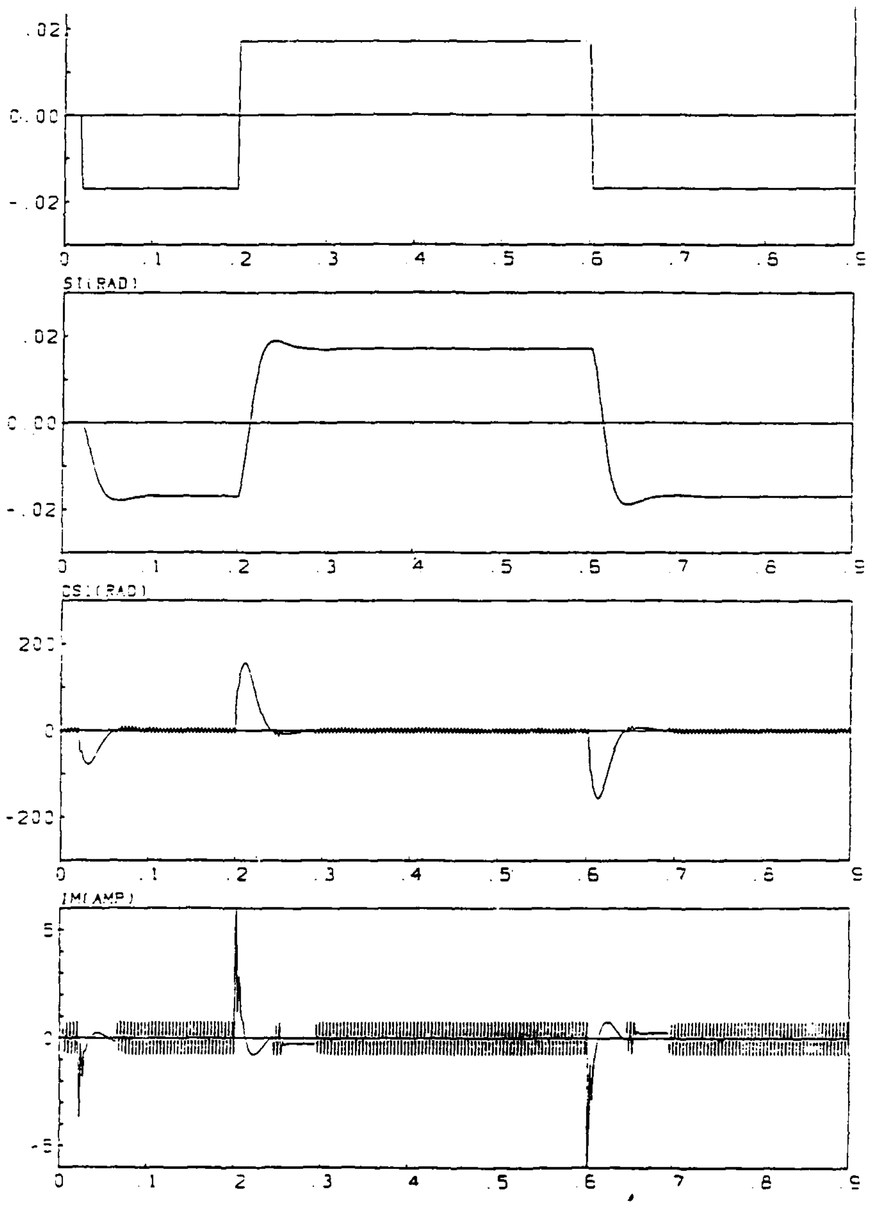


Fig B-11. Linear Domain Without Friction



Time-sec

Fig B-12. Linear Operation With Friction, DTRM = 0.001s



Time-sec

Fig B-13. Linear Operation With Friction, DTRM = 0.0020

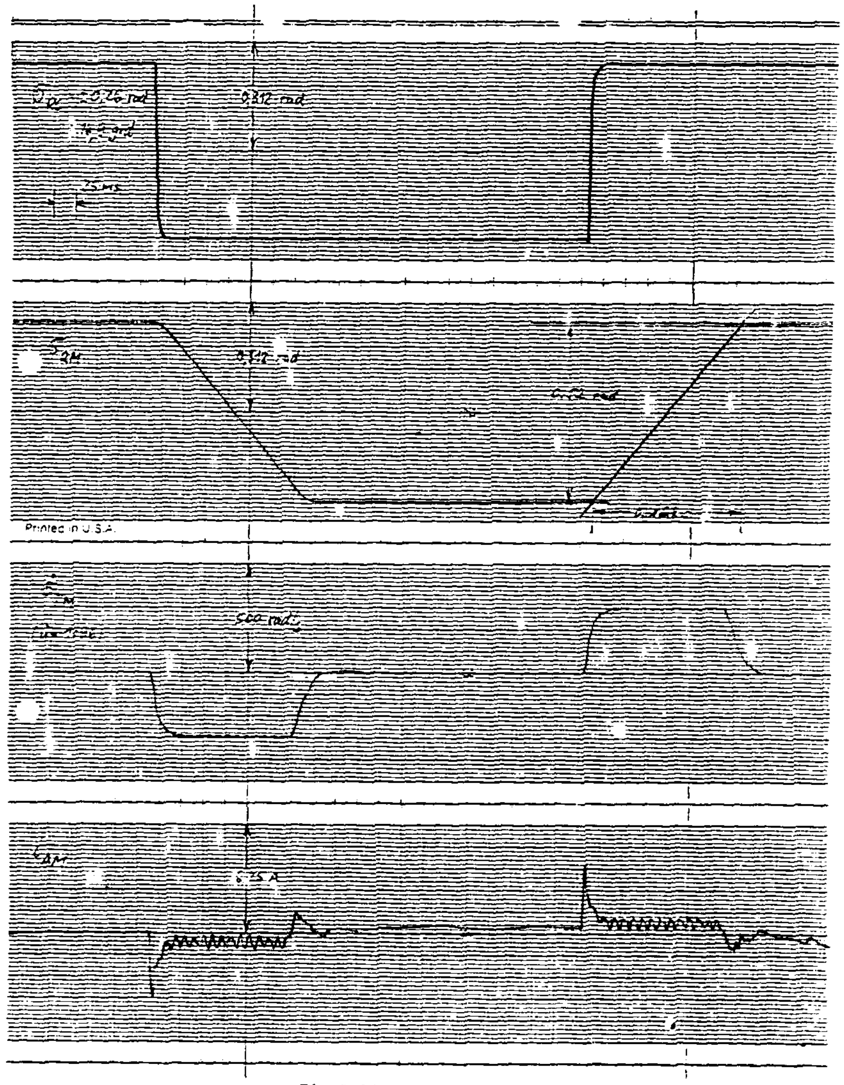


Fig B-14. Test A, Real System

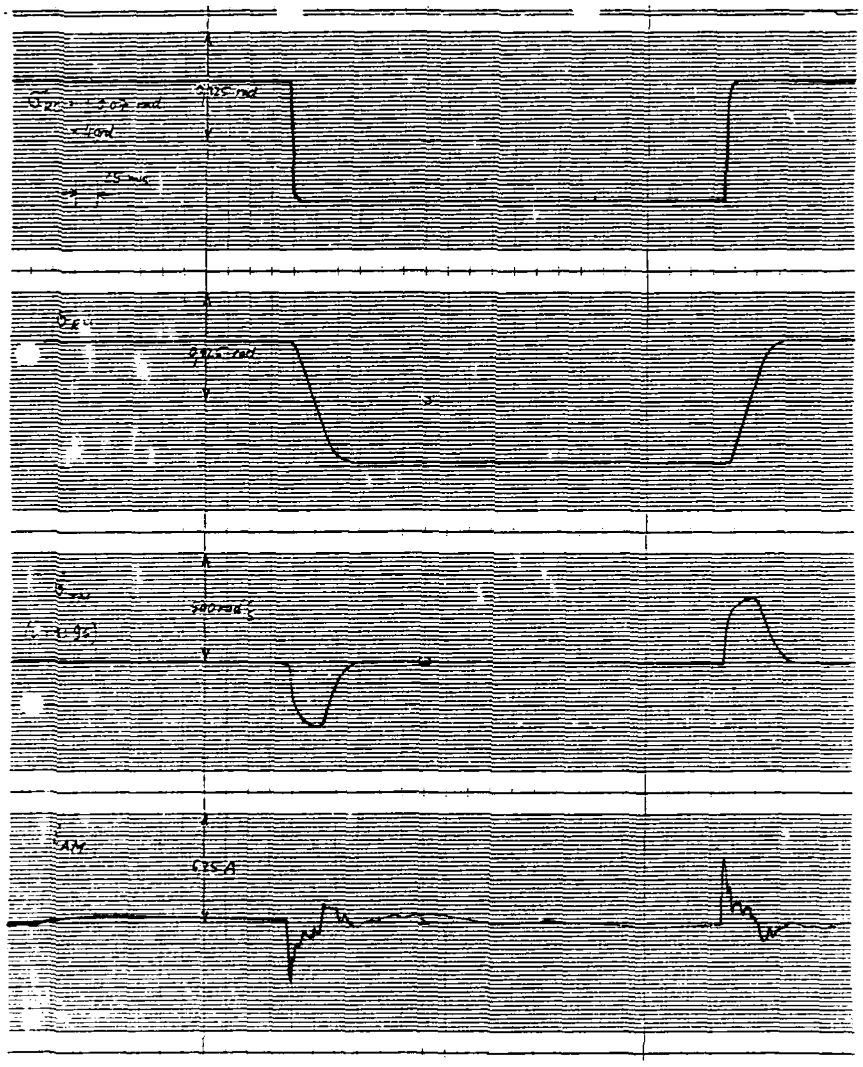


Fig B-15. Test B, Real System

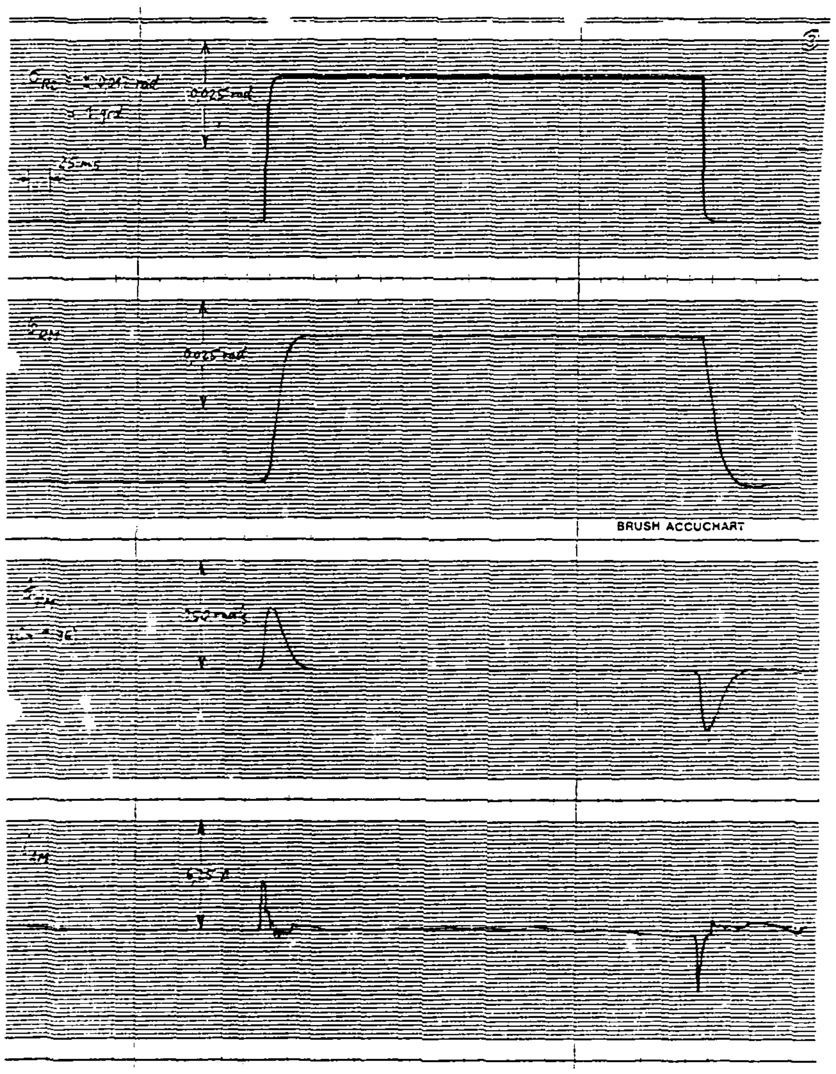
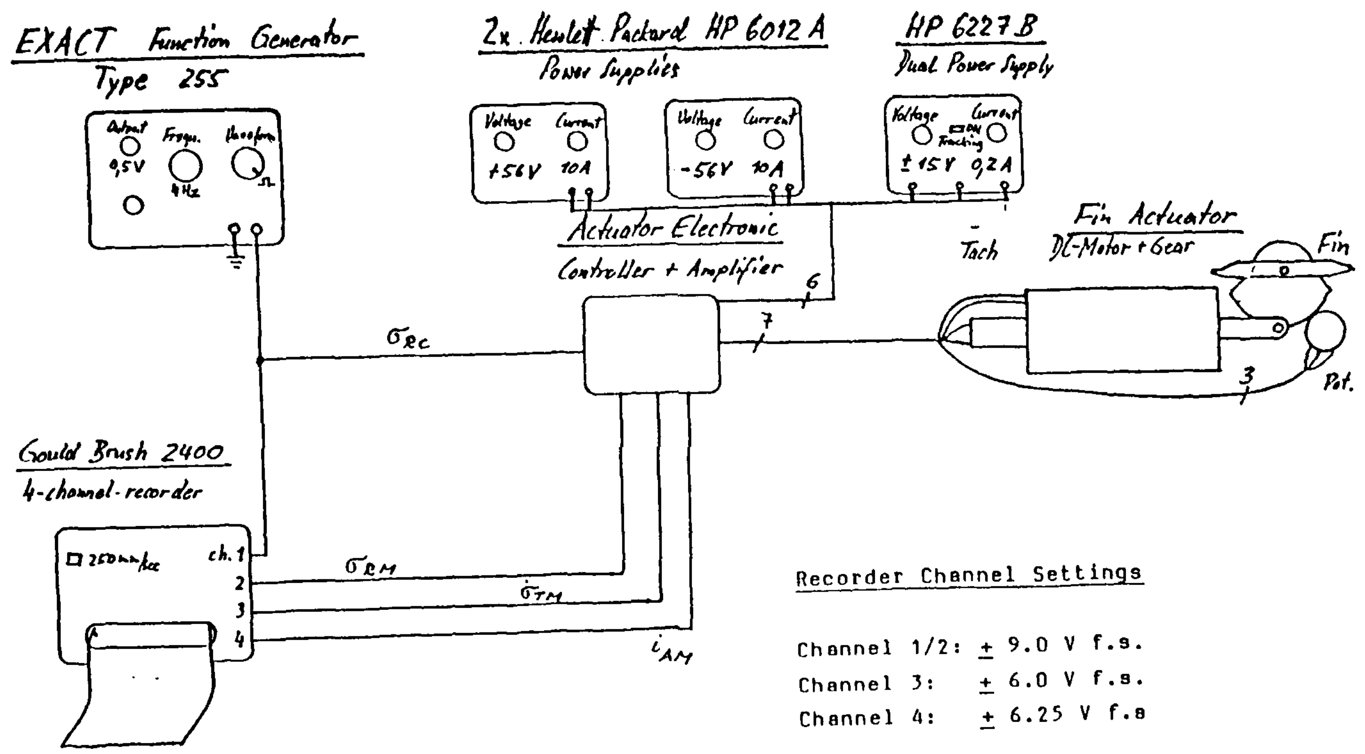


Fig B-16. Test C, Real System



Recorder Channel Settings

Channel 1/2: ± 9.0 V f.s.
Channel 3: ± 6.0 V f.s.
Channel 4: ± 6.25 V f.s

Fig B-17. Test Experiment

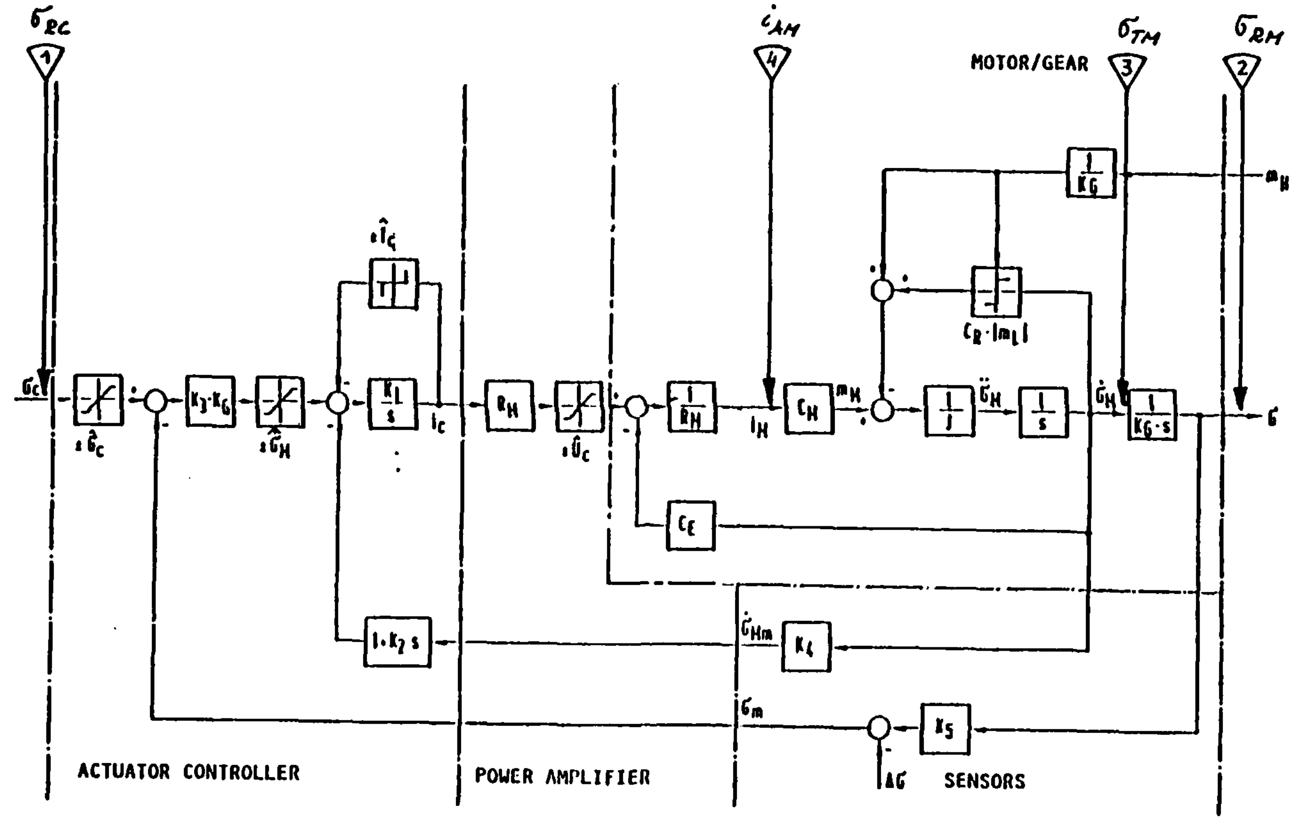


Fig B-18. Points of Measurement in Actuator System

APPENDIX C

PROGRAM DEBUGGING IN ACSL

One of the more important features of the ACSL language is the availability of tools that assist in pinpointing errors. The first thing is to establish a frame of mind that believes in the existence of errors. It is difficult, in general, for the average user who writes a model definition to believe that there are any errors. Accept the fact that all programs have at least one error and part of the joy of coming up with a finished product will be in finding it.

As the program is written, prepare the first run for debugging. Set the stop condition (TERMT) for the first run to a small value (typically one communication interval will suffice) so that no time will be wasted calculating the incorrect values. Use the 'D' option in the translator so that the program will proceed to uncover as many errors as possible.

The first run through the translator will produce syntax error indications and probably error messages as well. The translator analyzes each statement in turn and if an error occurs it will be indicated. The way the error is indicated is to write out again the statement in error, including any continuations, with a line of asterisks (*) underneath to indicate the acceptable section. The asterisk should stop just below where the error is located.

Example:

```
        X = Y + (SIN(Y.Y))

   ***SYNTAX ERROR***THE LINE IS LISTED WITH A POINTER TO THE ERROR

        X = Y + (SIN(Y.Y))

        **************
```

which shows that the period (.) separating the two Y's is not allowed. It should be an asterisk (*) to indicate "multiply". Two points should be noted when these errors are indicated. The first is that only the first error in the statement will be indicated. If this error is corrected, it may need a second (or third) run to uncover other problems further into the statement. When you make a correction, take a long hard look at the rest of the statement. The second is that line listed may not look like the input text if continuation cards are used. The error listing gives the complete string to be analyzed after the trailing blanks have been squeezed from the end of any continued cards.

Next, check for misspelling - variables that should have been the same get keypunched wrongly or names that should have been changed, overlooked. To check these, look at the symbol cross-reference tables listed at the end of the translator output. Any variables listed under 'VARIABLES NOT SPECIFIED IN ANY BLOCK' will be misspellings, constants you forgot to specify, or correct variables that had their name misspelled at the statement defining them. They should have been defined.

Next, take note of any unsatisfied external references from the load map. These will usually correspond to arrays you forgot to declare in an ARRAY statement. Without this, they look just like functions.

The first run-time command should set up a debug action with usually the first five or ten derivative evaluations sufficing. Include the following card at run-time:

```
SET  NDBUG = 10
```

Alternatively, an action can be scheduled that will ensure a debug printout after every START until CLEARed:

```
ACTION 'VAR' = 0.0, 'VAL' = 10, 'LOC' = NDBUG
```

NOTE: While the system variable NDBUG is greater than zero, the complete set of user variables is printed out and the the value of NDBUG is reduced by one.

This output is probably the most important data to help in debugging; the previous set of tools was merely to ensure that the mechanics were correct - commas in the right place, spellings consistent, etc. This debug output gives the actual numbers calculated for every one of the state derivatives and intermediate variables. The numbers should be examined carefully and checked for reasonableness using knowledge of the system being modelled. It is a good idea to start with initial conditions nonzero. If there are too many zero values, the arithmetic calculations can conceal errors. For preference, pick conditions so the derivatives all have a nonzero value which can be checked. Check the values that are listed for the constants. Any that have been preset in a CONSTANT statement and where the decimal point has been left off will be listed as having a value of 0.0. This problem is a very common error. Some arrays may be missing from this printout if they happen to be longer than the integer contained in the system variable MALPRN (Maximum Array Limit for Print Out). See system variable summary for the default value.

Now try the first full run. Plan what significant output variables will yield correct model operation. Specify these in an OUTPUT command, increase the termination time and START.

It is at this point that the modeler's skill comes in to rationalize the behavior of the simulation in terms of how the real word system is expected to behave. About the only help that can be offered is that once questionable areas have been uncovered, schedule debug printouts to cover the area of interest so that as much information is recorded as possible. Note that the debug output occurs after every derivative evaluation. For Runge-Kutta fourth order integration, four derivative evaluations are made for a time step (calculation interval): one at the beginning, two in the middle, and one at the end. The independent variable will appear to advance in half-steps with <u>two</u> derivative evaluations taking place each step. An <u>extra</u> evaluation will take place prior to each communication interval or <u>trip</u> through the DYNAMIC section.

MEANING OF DEBUG PRINT OUT

The debug output is generated by going through the user dictionary, which points to all variables in the user common block, and listing the values of each one by one. The first fifteen variables are ACSL control variables that are defined as follows:

(a) T - Real; Independent variable. May have been renamed in a VARIABLE statement

(b) ZZTICG - Real; Initial condition on the independent variable

(c) CINT - Real; Current communication interval. May have been renamed by CINTERVAL

(d) ZZIERR - Logical; Variable step error flag. May have been renamed by ERRTAG

(e) ZZNBLK - Integer; Number of DERIVATIVE and DISCRETE blocks in use

(f) ZZI - Integer; Distinguishes pre-initial (=0), START (=1) and CONTIN (=2)

(g) ZZST - Logical; Stop flag set by TERMT operator

(h) ZZFRFL - Logical, First flag set true at first derivative evaluation of every step

(i) ZZICFL - Logical; Initial condition flag set true at first derivative evaluation of every run - immediately after initial conditions have been transferred to states

(j) ZZRNFL - Logical; Reinitialize flag set true by REINIT. Used during initialization (ZZICFL = .TRUE.) and then turned false

(k) ZZNS - Integer array of length number of DERIVATIVE blocks giving number of state variables in each block

(l) MINT - Real array of length number of DERIVATIVE blocks giving minimum integration step size for each block. Name may be changed by global MINTERVAL statement

(m) MAXT - Real array of length number of DERIVATIVE blocks giving maximum integration step size for each block. Name may be changed by global MAXTERVAL statement

(n) NSTP - Integer array of length number of DERIVATIVE blocks giving communication interval divisor for each block. Name may be changed by global NSTEPS statement

(o) IALG - Integer array of length number of DERIVATIVE blocks giving integration algorithm number to be used for each block. Name may be changed by global ALGORITHM statement

Next, in the debug printout comes the list of state variables in DERIVATIVE block order and in alphabetical order within each block, with their corresponding derivatives and initial conditions on the same line. If line width (see TCWPRN and HVDPRN) is sufficient (126) the corresponding values of absolute error (XERR) and relative error (MERR) are also listed on the same line. In general, the derivatives will all be dummy variables (ZOnnnn form) except for those defined by the INTVC integration operator.

All the algebraic variables follow the states in alphabetical order. Any EQUIVALENCED variables are listed at the end. System variable ZZSEED contains the random number seed variable which will change (depends on machine type) with every call for a new random number. ZZTLXP is a logical variable present in some machine versions to request the reprieve/interrupt capability. If it is set false before the first START, normal system dumps can be obtained if desired.

DEBUG

A call to this routine will produce a debug list of all variables, excluding arrays greater than MALPRN (Maximum Array Limit for Print) on both PRN and DIS units. The technique of setting NDBUG to a positive integer; yields a debug list at the end of every derivative evaluation. While useful as a checkout tool with large programs, this action can produce an overwhelming amount of output. Selective output can now be obtained by:

```
IF (logical condition) CALL DEBUG
```

included in the DERIVATIVE section. Including the statement

```
CALL DEBUG
```

in the DYNAMIC section produces the entire list at each communication interval and is synonymous with asking for the OUTPUT of all variables.

Including

```
IF (DUMP) CALL DEBUG
```

in the TERMINAL section is a useful artifice since all final values are displayed as well as the initial conditions for that run.

AD-A166 617

REPORT DOCUMENTATION PAGE

1. Recipient's Reference	2. Originator's Reference	3. Further Reference	4. Security Classification of Document
	AGARD-AR-206	ISBN 92-835-1514-5	UNCLASSIFIED

5. Originator Advisory Group for Aerospace Research and Development
North Atlantic Treaty Organization
7 rue Ancelle, 92200 Neuilly sur Seine, France

6. Title VALIDATION OF MISSILE SYSTEM SIMULATION

7. Presented at

8. Author(s)/Editor(s)	9. Date
Various	November 1985

10. Author's/Editor's Address	11. Pages
Various	114

12. Distribution Statement This document is distributed in accordance with AGARD policies and regulations, which are outlined on the Outside Back Covers of all AGARD publications.

13. Keywords/Descriptors

Missile simulators	Missiles
Mathematical models	Computer systems programs

14. Abstract

A survey of missile simulation facilities ~~in~~ 1982 indicated that very little has been devoted to the validation of missile system simulations. Those validation techniques that are used are not standardized, are often ill defined and are generally undocumented. In an attempt to rectify this situation, the AGARD Flight Mechanics Panel established Working Group 12 to examine missile simulation procedures and recommend techniques for the validation and universalization of the results of such procedures. This report presents the output of this Working Group.

The report recommends a simulation terminology that, if adopted, should simplify the validation processs. A hierarchical model representation called, "Confidence Level in Model Behaviour," (CLIMB) is also recommended ~~that~~ organizes the simulation process and emphasizes a standardized documentation procedure for both simplifying the orderly use of simulation and insuring confidence in the final simulation results. Examples of using the CLIMB model are included in the Appendices.

A brief section on Computer Languages and how they can benefit the simulation process is included. Software verification, validation and assessment methods are examined and recommendations made to enhance the validation of the simulation.

This report represents the combined efforts of a group consisting of representatives from France, West Germany, Italy, United Kingdom and United States. The information presented, therefore, is truly international in nature.

This Advisory Report was prepared at the request of the AGARD Flight Mechanics Panel.

AGARD Advisory Report No.206
Advisory Group for Aerospace Research and Development, NATO
VALIDATION OF MISSILE SYSTEM SIMULATION
Published November 1985
114 pages

A survey of missile simulation facilities in 1982 indicated that very little has been devoted to the validation of missile system simulations. Those validation techniques that are used are not standardized, are often ill defined and are generally undocumented. In an attempt to rectify this situation, the AGARD Flight Mechanics Panel established Working Group 12 to examine missile simulation procedures and recommend techniques for the validation and universalization of the results of such procedures. This report presents the output of this Working Group.

P.T.O

AGARD-AR-206

Missile simulators
Mathematical models
Missiles
Computer systems programs

AGARD Advisory Report No.206
Advisory Group for Aerospace Research and Development, NATO
VALIDATION OF MISSILE SYSTEM SIMULATION
Published November 1985
114 pages

A survey of missile simulation facilities in 1982 indicated that very little has been devoted to the validation of missile system simulations. Those validation techniques that are used are not standardized, are often ill defined and are generally undocumented. In an attempt to rectify this situation, the AGARD Flight Mechanics Panel established Working Group 12 to examine missile simulation procedures and recommend techniques for the validation and universalization of the results of such procedures. This report presents the output of this Working Group.

P.T.O

AGARD-AR-206

Missile simulators
Mathematical models
Missiles
Computer systems programs

AGARD Advisory Report No.206
Advisory Group for Aerospace Research and Development, NATO
VALIDATION OF MISSILE SYSTEM SIMULATION
Published November 1985
114 pages

A survey of missile simulation facilities in 1982 indicated that very little has been devoted to the validation of missile system simulations. Those validation techniques that are used are not standardized, are often ill defined and are generally undocumented. In an attempt to rectify this situation, the AGARD Flight Mechanics Panel established Working Group 12 to examine missile simulation procedures and recommend techniques for the validation and universalization of the results of such procedures. This report presents the output of this Working Group.

P.T.O

AGARD-AR-206

Missile simulators
Mathematical models
Missiles
Computer systems programs

AGARD Advisory Report No.206
Advisory Group for Aerospace Research and Development, NATO
VALIDATION OF MISSILE SYSTEM SIMULATION
Published November 1985
114 pages

A survey of missile simulation facilities in 1982 indicated that very little has been devoted to the validation of missile system simulations. Those validation techniques that are used are not standardized, are often ill defined and are generally undocumented. In an attempt to rectify this situation, the AGARD Flight Mechanics Panel established Working Group 12 to examine missile simulation procedures and recommend techniques for the validation and universalization of the results of such procedures. This report presents the output of this Working Group.

P.T.O

AGARD-AR-206

Missile simulators
Mathematical models
Missiles
Computer systems programs

The report recommends a simulation terminology that, if adopted, should simplify the validation processs. A hierarchical model representation called, "Confidence Level in Model Behaviour," (CLIMB) is also recommended that organizes the simulation process and emphasizes a standardized documentation procedure for both simplifying the orderly use of simulation and insuring confidence in the final simulation results. Examples of using the CLIMB model are included in the Appendices.

A brief section on Computer Languages and how they can benefit the simulation process is included. Software verification, validation and assessment methods are examined and recommendations made to enhance the validation of the simulation.

This report represents the combined efforts of a group consisting of representatives from France, West Germany, Italy, United Kingdom and United States. The information presented, therefore, is truly international in nature.

This Advisory Report was prepared at the request of the AGARD Flight Mechanics Panel.

ISBN 92-835-1514-5

The report recommends a simulation terminology that, if adopted, should simplify the validation processs. A hierarchical model representation called, "Confidence Level in Model Behaviour," (CLIMB) is also recommended that organizes the simulation process and emphasizes a standardized documentation procedure for both simplifying the orderly use of simulation and insuring confidence in the final simulation results. Examples of using the CLIMB model are included in the Appendices.

A brief section on Computer Languages and how they can benefit the simulation process is included. Software verification, validation and assessment methods are examined and recommendations made to enhance the validation of the simulation.

This report represents the combined efforts of a group consisting of representatives from France, West Germany, Italy, United Kingdom and United States. The information presented, therefore, is truly international in nature.

This Advisory Report was prepared at the request of the AGARD Flight Mechanics Panel.

ISBN 92-835-1514-5

The report recommends a simulation terminology that, if adopted, should simplify the validation processs. A hierarchical model representation called, "Confidence Level in Model Behaviour," (CLIMB) is also recommended that organizes the simulation process and emphasizes a standardized documentation procedure for both simplifying the orderly use of simulation and insuring confidence in the final simulation results. Examples of using the CLIMB model are included in the Appendices.

A brief section on Computer Languages and how they can benefit the simulation process is included. Software verification, validation and assessment methods are examined and recommendations made to enhance the validation of the simulation.

This report represents the combined efforts of a group consisting of representatives from France, West Germany, Italy, United Kingdom and United States. The information presented, therefore, is truly international in nature.

This Advisory Report was prepared at the request of the AGARD Flight Mechanics Panel.

ISBN 92-835-1514-5

The report recommends a simulation terminology that, if adopted, should simplify the validation processs. A hierarchical model representation called, "Confidence Level in Model Behaviour," (CLIMB) is also recommended that organizes the simulation process and emphasizes a standardized documentation procedure for both simplifying the orderly use of simulation and insuring confidence in the final simulation results. Examples of using the CLIMB model are included in the Appendices.

A brief section on Computer Languages and how they can benefit the simulation process is included. Software verification, validation and assessment methods are examined and recommendations made to enhance the validation of the simulation.

This report represents the combined efforts of a group consisting of representatives from France, West Germany, Italy, United Kingdom and United States. The information presented, therefore, is truly international in nature.

This Advisory Report was prepared at the request of the AGARD Flight Mechanics Panel.

ISBN 92-835-1514-5