This is to certify that the

thesis entitled

DYNAMIC MODEL OPTIMIZATION
USING COMPUTATIONAL TECHNIQUES

presented by

Mark Lee Davis

has been accepted towards fulfillment
of the requirements for

Masters degree in Mechanical Engineering

R C Rosenberg
Major professor

Date 2/24/92

DYNAMIC MODEL OPTIMIZATION USING COMPUTATIONAL TECHNIQUES

By

Mark Lee Davis

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Mechanical Engineering

1992

ABSTRACT

DYNAMIC MODEL OPTIMIZATION USING COMPUTATIONAL TECHNIQUES

BY

Mark Lee Davis

Simulation modeling software can create and solve a set of differential equations and provide numerical results of the system state responses. Optimization software can explore a design space, searching for a feasible solution which best satisfies a given cost function. Our objective is to establish an interface between these two types of software packages, so the strengths of each are joined in an effort to reduce design time while ensuring accuracy. A specific interface between modeling and optimization software has been established using ENPORT for simulation and OPTDES.BYU for optimization. By using this interface dynamic responses can be optimized with respect to system design variables for any bond graph model. Design time can be reduced while maintaining confidence in the results.

DEDICATION

To my devoted and loving wife Diana
and
My sons Mark, Brian and Eric

ACKNOWLEDGMENTS

I would like to begin by expressing my appreciation to the College of Engineering, the Department of Mechanical Engineering and the CASE Center for the support that made this opportunity of a lifetime possible. My thanks go to the CASE Center staff consultants who, with great satisfaction, answered my many questions. I would like to thank Dr. Clark Radcliffe for both his service as a committee member and also his valuable input to the final thesis document. Finally I wish to express my thanks to professors Ronald Rosenberg and Alejandro Díaz my co-advisors. With the wisdom and guidance displayed by these men this accomplishment was possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

AF	=	**ANALYSIS FUNCTION**
AV	=	**ANALYSIS VARIABLE**
DOF	=	**DEGREES OF FREEDOM**
DV	=	**DESIGN VARIABLE**
E/F	=	**EFFORT/FLOW**
MFS	=	**MULTIPLE FORCING SYSTEM**
NFF	=	**NUMBER OF FORCING FUNCTIONS**
NFFCN	=	**FORCING FUNCTION NUMBER**
NP	=	**NAMED PARAMETER**
NPM	=	**NUMBER OF PERFORMANCE MEASUREMENTS**
NU	=	**NUMBER OF INPUT VARIABLES**
NX	=	**NUMBER OF STATE VARIABLES**
NY	=	**NUMBER OF OUTPUT VARIABLES**
OC	=	**OBJECTIVE CONSTRAINT**
OF	=	**OBJECTIVE FUNCTION**
OP	=	**OBJECTIVE PERFORMANCE**
PM	=	**PERFORMANCE MEASUREMENT**
PMM	=	**PERFORMANCE MEASUREMENT METHOD**
XIN	=	**STATE VARIABLE INITIAL CONDITIONS**
Y	=	**ENPORT MODEL OUTPUT RESPONSE**

CHAPTER 1

INTRODUCTION

1.1 Motivation

Simulation software can create and solve a set of differential equations and provide numerical results of the system state responses. Optimization software can explore the design space, searching for a feasible solution which best satisfies a given cost function. In general simulation software does not have the capability to systematically search a design space for an optimal solution. The user must do this by hand, guessing design variable changes. Similarly, optimization software is not capable of assembling and solving a set of differential equations; this task must be done by the user. Each of these software package types has limitations in providing a complete service to the design engineer. Our objective is to establish an interface between these two types of software packages. By doing so the strengths of each will be joined in an effort to reduce design time while ensuring accuracy.

1.2 Basic Problem

In dynamics design considerations often relate to controlling the transient or steady state response for an excited system. The desired response will be a function of the individual system components and their inter-relationships. Hence changing component values will alter the system response. Consider a mechanical position control system.

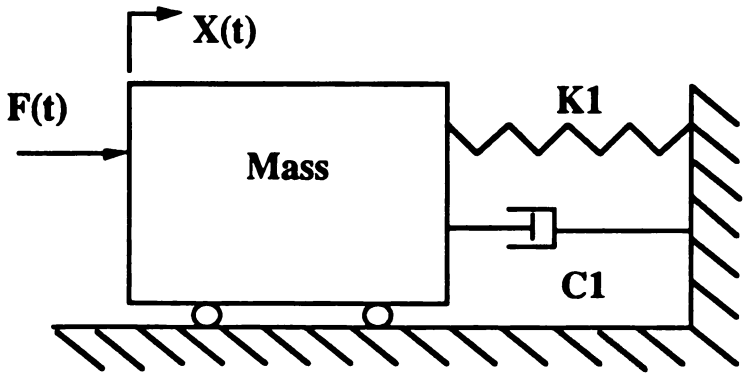


Figure 1.2.1
Mechanical Positioner

Typical design objectives include minimizing settling time or rise time, setting displacements, and so on. Each objective requires a particular design. The design variables of this system might be the spring, mass and damper parameters. These parameters will typically be bounded by design limitations. The question then becomes "Which combination of design variables best meets the objective?" Optimization is a systematic process of searching for the feasible design which best satisfies an objective function. In a 1-DOF problem like this analytical results may be obtainable. However, in multi-DOF problems analytical solutions are not easily found and the search space becomes very large. With our interface it is possible to optimize a multi-DOF model relative to a specific objective with proven computational techniques.

1.3 Organization of the Thesis

In chapter 2 an overview of modeling and optimization software is discussed. The advancement made by the interface software is described. An example illustrating use of the interface is presented in chapter 3. The problem is a 5 DOF suspension system posed and solved 3 different ways. Chapter 4 will conclude the report with a summary and recommendations. In the appendices the reader will find discussion detailing the interface. In appendix A design considerations made in ENPORT [1] and OPTDES [2] will be discussed. Appendix B will discuss a special class of problem where the system is driven by multiple inputs. To expand the interface capablilites the user can create their own program model which can be used with the ENPORT model. These user written subroutines are discussed in appendix C. Appendix D is the users guide.

CHAPTER 2

CURRENT STATUS

2.1 Simulation Using ENPORT

ENPORT is a modeling and simulation software package which aids in the performance assessment of dynamic systems. ENPORT is based on bond graph theory. It combines a set of standard block diagrams and bond graph elements to build system models. With ENPORT the user can create output files of the state equations describing the model behavior.

This software is typical of simulation packages in that it is capable of determining the state response of an N-DOF system but does not explore the design space in search of an optimal design solution. Given a specific set of design variables, ENPORT generates a set of output responses.

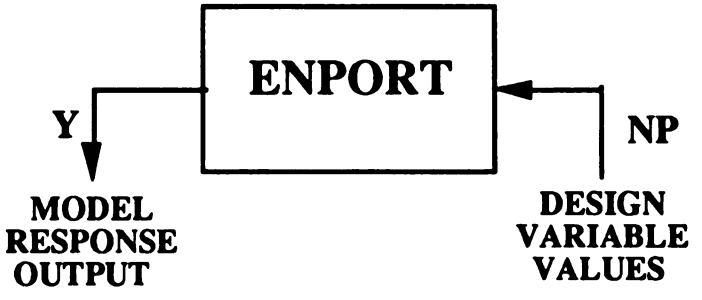


Figure 2.1.1
ENPORT Data Flow

With ENPORT the user must enter a set of design variables by hand, then examine the system response to determine design changes. To provide a convenient way to search the design space ENPORT can describe the set of design variables using named-parameters (NP). By changing the NP vector the user can study new

designs. By examining the output responses (Y) the user can determine if the new design is better, feasible, and which changes can improve the model performance. Optimizing with simulation software is obviously burdensome to the design engineer.

2.2 Optimization Using OPTDES

OPTDES is a design optimization software package. Within OPTDES many optimization algorithms for exploring the design space are offered. The user can control the design search by changing the step size, convergence tolerance, and other parameters. The design space can be scaled, which improves algorithm performance and identifies active constraints more simply. OPTDES provides many post-processing features. OPTDES is typical of most optimization software in that it passes a set of independent design variable values (AV) to the model and receives a set of analysis function values (AF) back from the model. To start the optimization process the user provides an initial design point (AV_0) and the final optimized design is returned (AV_f).

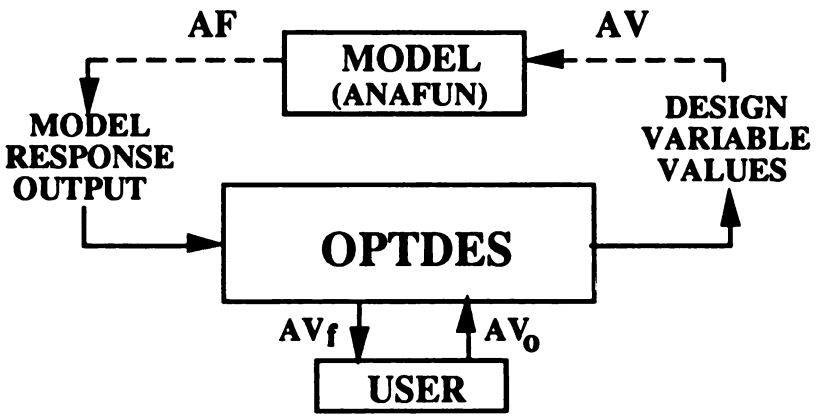


Figure 2.2.1
OPTDES Data Flow

The analysis functions are used to evaluate the performance of the model to determine the best feasible design. With OPTDES the user must develop and program the set of equations describing the model. The equations program name used with the OPTDES software is ANAFUN. In an N-DOF problem this process will require much time and allows for errors to creep into in the model. When

designing, an engineer requires accuracy and speed in building a model, which optimization software does not provide.

2.3 Interfacing ENPORT and OPTDES

The interface is intended to combine the equation solving capabilities of ENPORT with the design space exploration power of OPTDES. By doing this models built in ENPORT can be optimized.

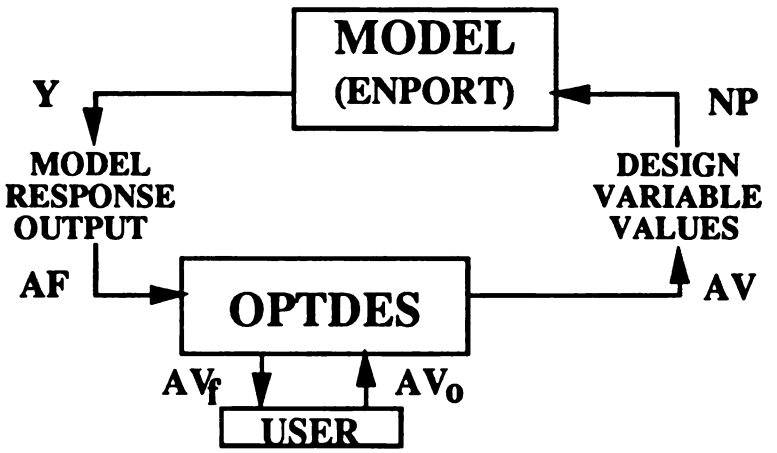


Figure 2.3.1
ENPORT - OPTDES Connection

FIGURE 2.3.1 illustrates the ideal interface between ENPORT and OPTDES. This figure shows the relationships between the AV and NP vectors and the Y and AF vectors. The ENPORT model receives a set of design variables (AV/NP) from OPTDES and returns the analysis functions (Y/AF) for evaluating the system performance. The purpose of the interface is to make the transition from the variable forms AV to NP and Y to AF.

As seen from FIGURE 2.3.2 the NP and AV vectors are the same. This allows the ENPORT model to receive design variables directly. Output from the

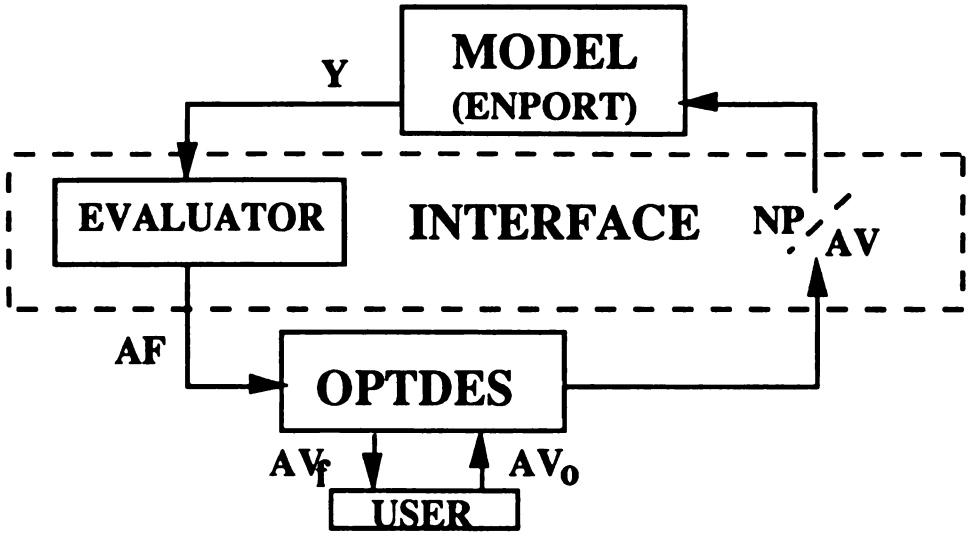


Figure 2.3.2
Interface Function

ENPORT model is a function of time. As a result the time response must be evaluated to provide OPTDES with the analysis functions in the proper format, a constant value (i.e. $\underline{AF} = \max |\underline{Y}(t)|$).

2.3.1 Describing the Interface Connection

Enport has the capability of generating a Fortran file defining a set of N coupled first-order differential equations from the system graph model. This ENPORT file can be used to calculate the state derivatives for a given set of initial conditions. Within OPTDES (in subroutine ANAFUN) is an integration routine (4^{th} order Runge-Kutta) which can call the ENPORT Fortran file and produce a time response of the system.

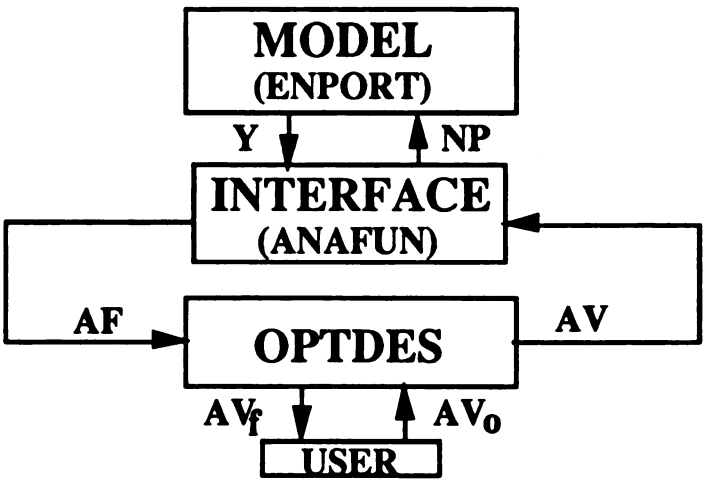


Figure 2.3.1.1
ENPORT - OPTDES - INTERFACE Connection

OPTDES passes the design variables to the interface through the AV vector. The interface calls the ENPORT Fortran file, passing the design variables through the NP vector. ENPORT returns a time history to the interface through the Y vector. The interface now evaluates the time history and returns the analysis function values to OPTDES through the AF vector. One consideration in designing this link was to establish a standardized call to the ENPORT Fortran file, which makes this connection possible for any bond-graph model. In addition this connection will work with any filename given to the ENPORT Fortran file. This allows for a library of models to be created from which the user can select one for optimization.

CHAPTER 3

AN APPLICATION EXAMPLE

3.1 Introduction

To illustrate the capabilities of the interface the design of a 5-DOF suspension system was used. The problem posed here is a re-creation of the one given by Haug and Arora [3]. In their solution they employed analytical methods to determine the state equations and the derivatives of the design functions with respect to the design variables. Using computational methods this model was optimized. With this interface the state equations will be generated by ENPORT. OPTDES will then determine the necessary gradients to explore the design space. With the interface the design engineer is responsible for building the bond-graph in ENPORT and posing the optimization problem in OPTDES. As we shall see the results obtained for the objective function using the design interface were within 5% of those obtained by Haug and Arora for each problem.

3.2 Design Problem Considerations

The following schematic illustrates the 5 DOF suspension system being designed.

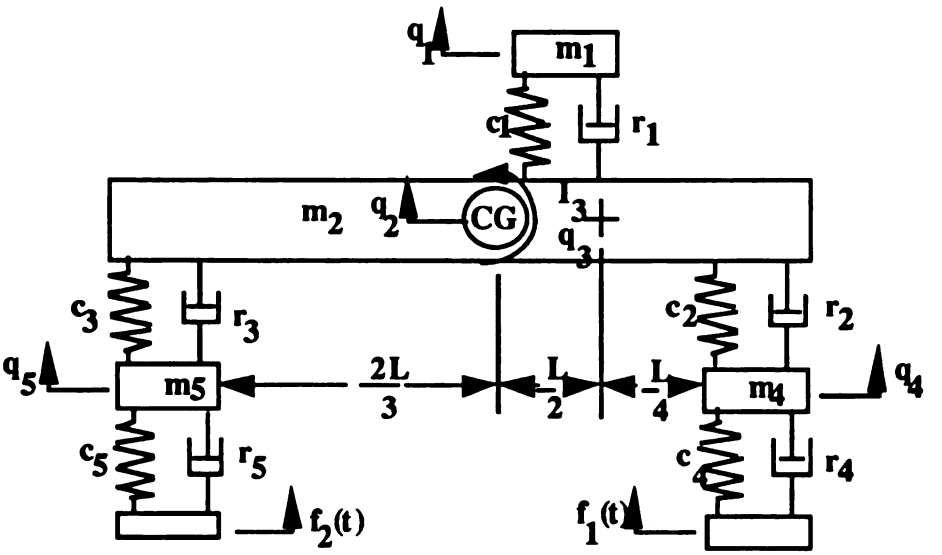


Figure 3.2.1
5 D.O.F. Suspension System

The system is excited by tire displacements due to interaction with the road surface. Three surfaces were considered representing a severe bump and two different highway conditions. For modeling purposes the tire displacement profiles were differentiated one time. The result is a velocity profile of the tire input. The velocity can be modeled as a source of flow into the ENPORT model. The following velocity profiles were used.

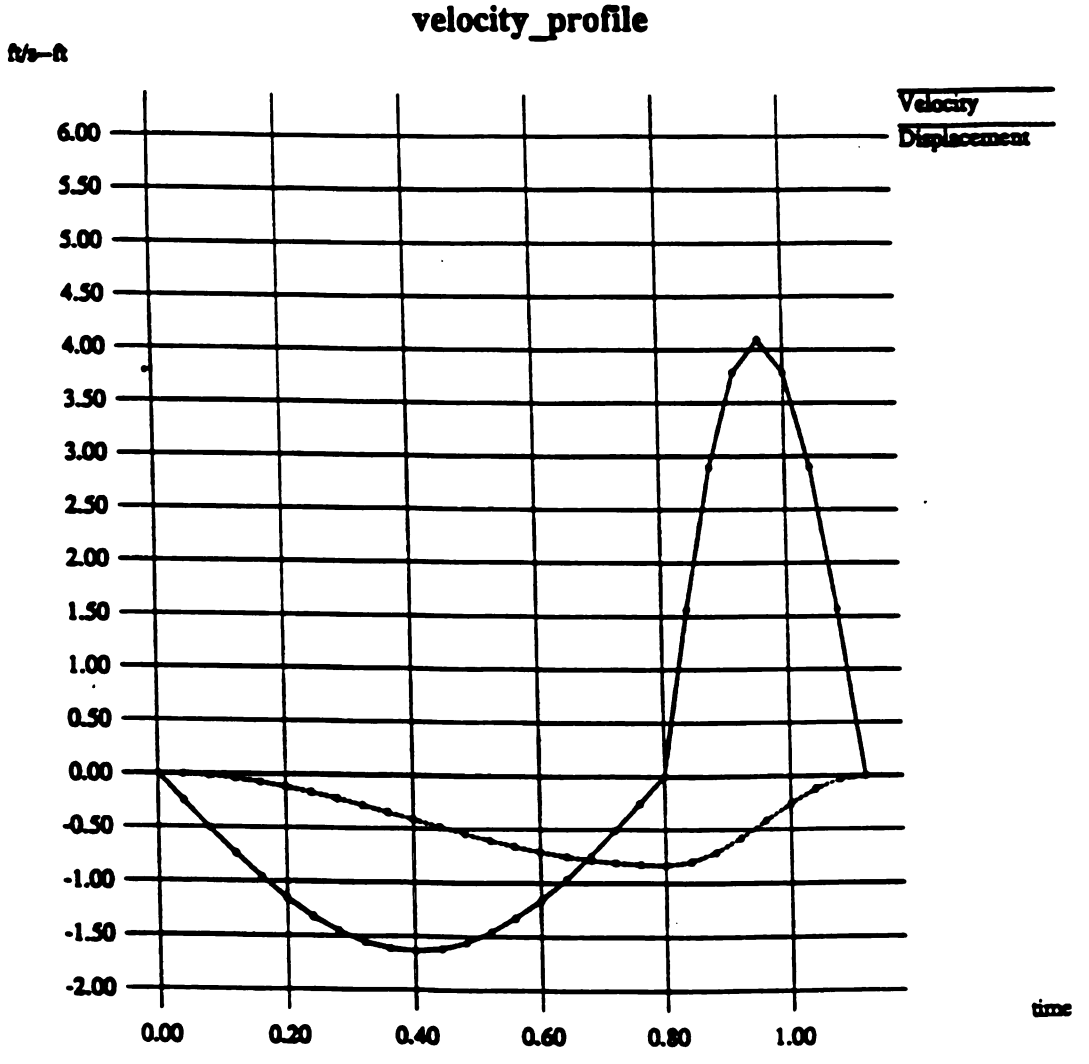


Figure 3.2.2
Profile 1 - Severe Bump

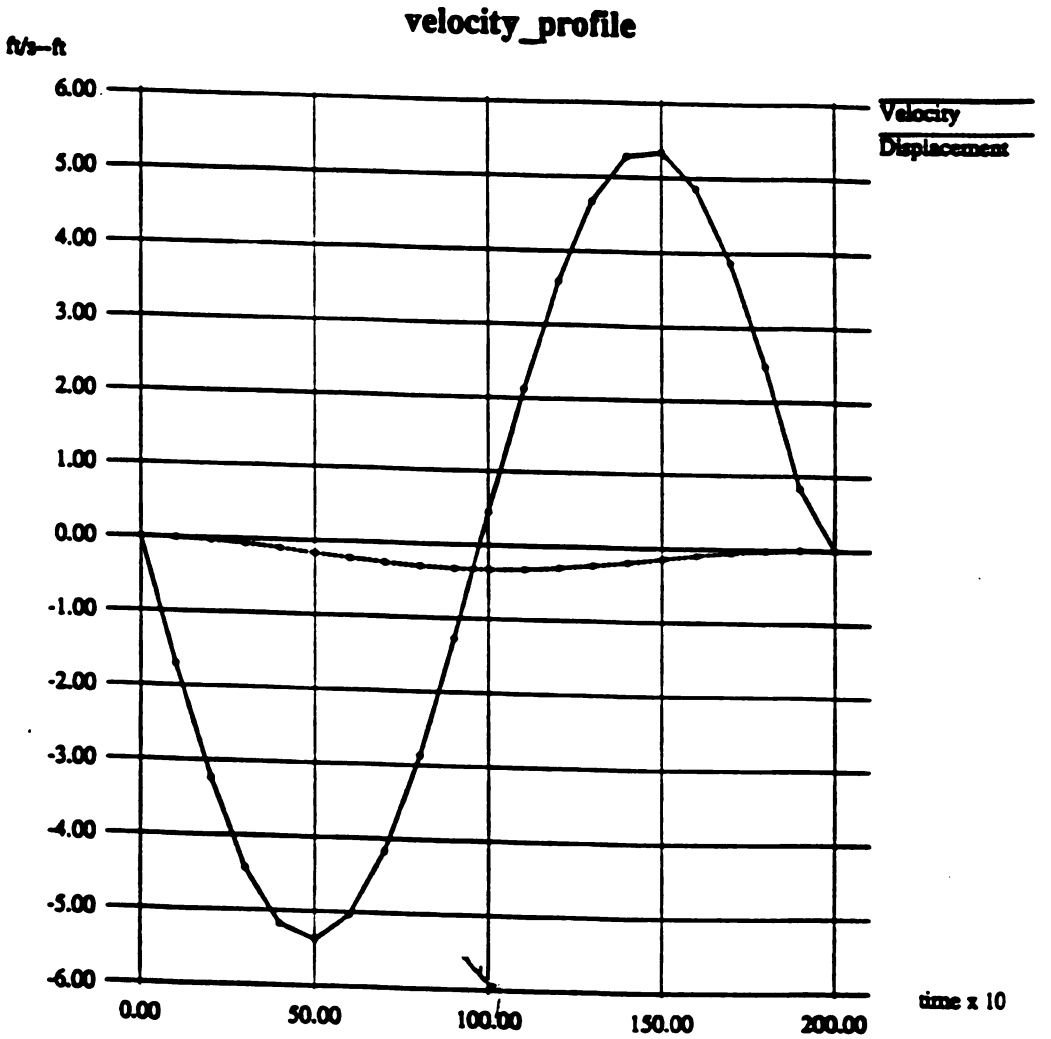


Figure 3.2.3
Profile 2 - Highway Condition No. 1

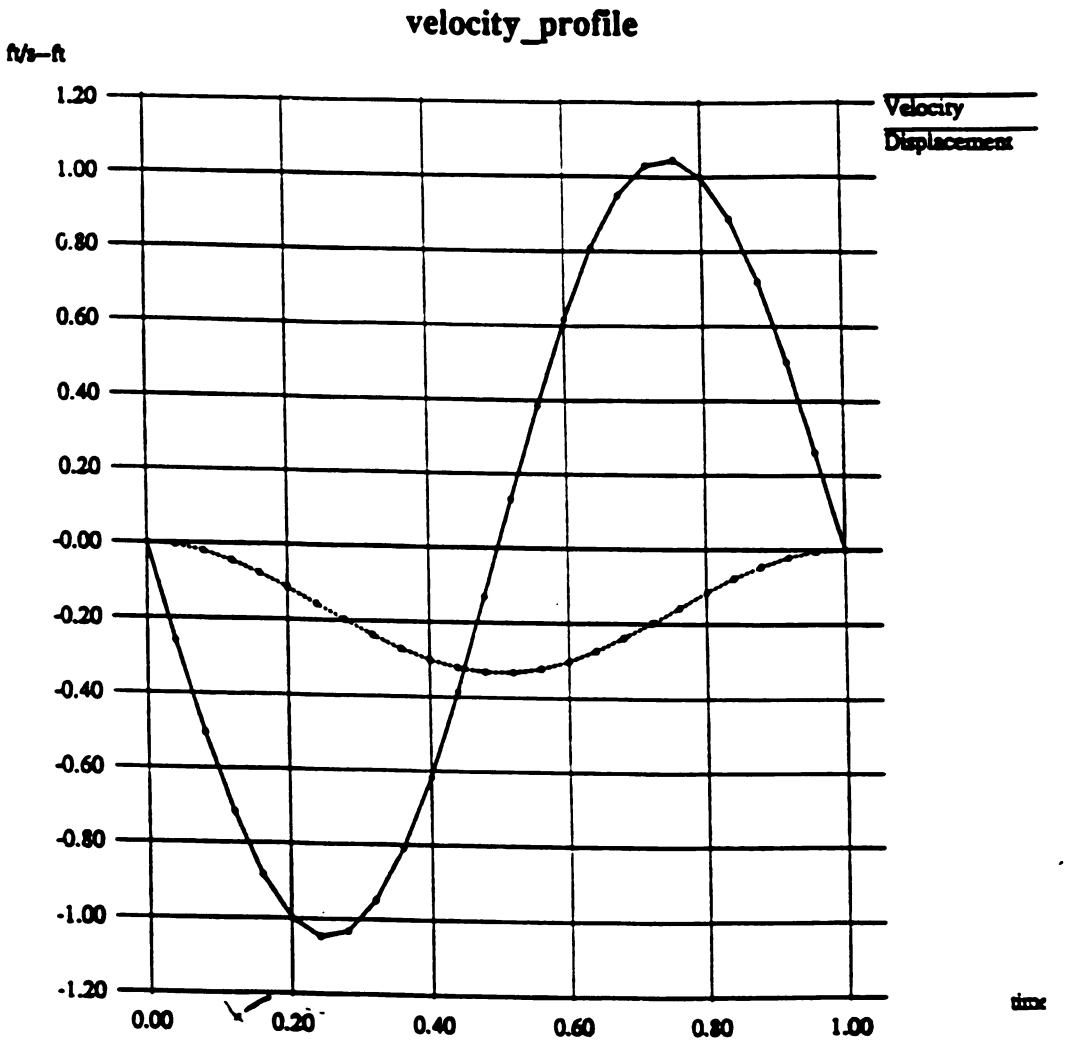


Figure 3.2.4
Profile 3 - Highway Condition No. 2

3.3 Bond Graph Design

The bond-graph model of Figure 3.3.1 was built in ENPORT. This model represents the suspension system. SF1 represents velocity input to the front tire and SF2 is velocity input to the rear tire.

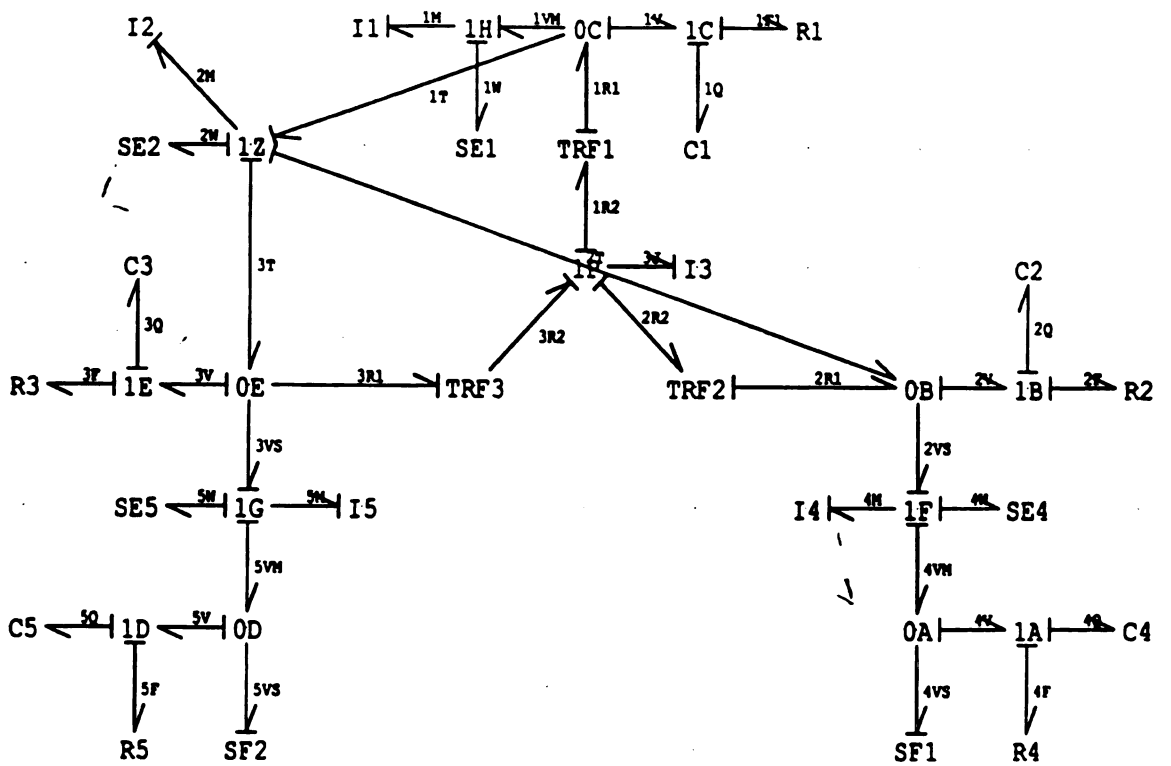


Figure 3.3.1
5 D.O.F. Bond Graph

3.4 Optimization Problem

Three different optimization problems will be posed. In each case the objective is to minimize the maximum seat force ensuring driver comfort. Each problem is characterized by road conditions.

Table 3.4.1
Problem Charachterization

Problem No.	Road Profile(s)	Vehicle Speed(s) (in/sec)
1	1	450
2	2, 3	960 / 960
3	1, 2	450 / 960

Problems 2 and 3 will illustrate a problem where the system is optimized to responses for more than one input.

3.4.1 Objective Functions

The objective function for each problem is posed in the following way:

Problem 1 $\quad \text{Min } \Psi = \text{Max} \mid E.1M_1(t,b) \mid$

Problem 2 $\quad \text{Min } \Psi = \text{Max} \mid E.1M_i(t,b) \mid \quad i = 2,3$

Problem 3 $\quad \text{Min } \Psi = \text{Max} \mid E.1M_2(t,b) \mid$

Note: Subscripts represent velocity profiles.
E.1M is the effort (i.e., force) at the seat.

3.4.2 Model and Optimiation Parameters

The analysis variables used are

Table 3.4.2.1
Analysis Variables

AV	DESCRIPTION	NODE	BOUNDS	
			LOWER	UPPER
1	Seat Spring	C1	600	6000
2	Front Spring	C2	2400	12000
3	Rear Spring	C3	2400	12000
4	Seat Damper	R1	24	600
5	Front Damper	R2	30	960
6	Rear Damper	R3	30	960

The performance measurements are

Table 3.4.2.2
Performance Measurements

PM	DESCRIPTION	VBL	MAXIMUM ALLOWABLE
1	Seat Spring Disp	Q.1Q	.167
2	Seat Force	E.1M	Obj. Fcn.
3	Front Spring Disp	Q.2Q	.417
4	Rear Spring Disp	Q.3Q	.417
5	Front Tire Compr	Q.4Q	.167
6	Rear Tire Compr	Q.5Q	.167

The AF vectors will have the following formats in SETUP:

Table 3.4.2.3
Analysis Functions

AF	PM	Constraint Value		
		PROBLEM 1	PROBLEM 2	PROBLEM 3
1	1	< .167	< .167	< .167
2	2	* 0.0	** < 0.0	< 500
3	3	< .417	< .417	< .417
4	4	< .417	< .417	< .417
5	5	< .167	< .167	< .167
6	6	< .167	< .167	< .167
7	1	----	< .167	< .167
8	2	----	** < 0.0	* 0.0
9	3	----	< .417	< .417
10	4	----	< .417	< .417
11	5	----	< .167	< .167
12	6	----	< .167	< .167
13	1	----	* 0.0	----

* Objective Function
** Objective Function Set Member

3.5 Re

tho

rep

des

of

3.5

3.5 Results

The results of each problem are presented in a set of tables and compared to those reported by Haug and Arora. Included with each problem is a graphical representation of the seat force and active constraint for both initial and optimized design variables. These graphs were produced using ENPORT. A graphical history of the objective function values is also included.

3.5.1 Problem 1 Summary

In problem one the most significant difference was in the seat damper values (see Table 3.5.1.1). The optimized value was 142% greater than the Haug and Arora solution. The optimized values for the objective functions were within 1% of each other. Figures 3.5.1.1 and 3.5.1.2 illustrate the seat force and the front spring displacement for initial and optimized design variables. These results were achieved in 7 iterations (see Figure 3.5.1.3).

Table 3.5.1.1
Problem 1 Results

AV	HAUG & ARORA	OPTDES RESULTS	PERCENT DIFFERENCE
C1	600.0	600.0	0.00
C2	2400.0	2400.0	0.00
C3	2902.8	2400.0	- 17.32
R1	154.7	375.0	142.04
R2	930.2	926.3	- 0.42
R3	960.0	960.0	0.00
OBJ FCN	193.1	191.5	-0.83

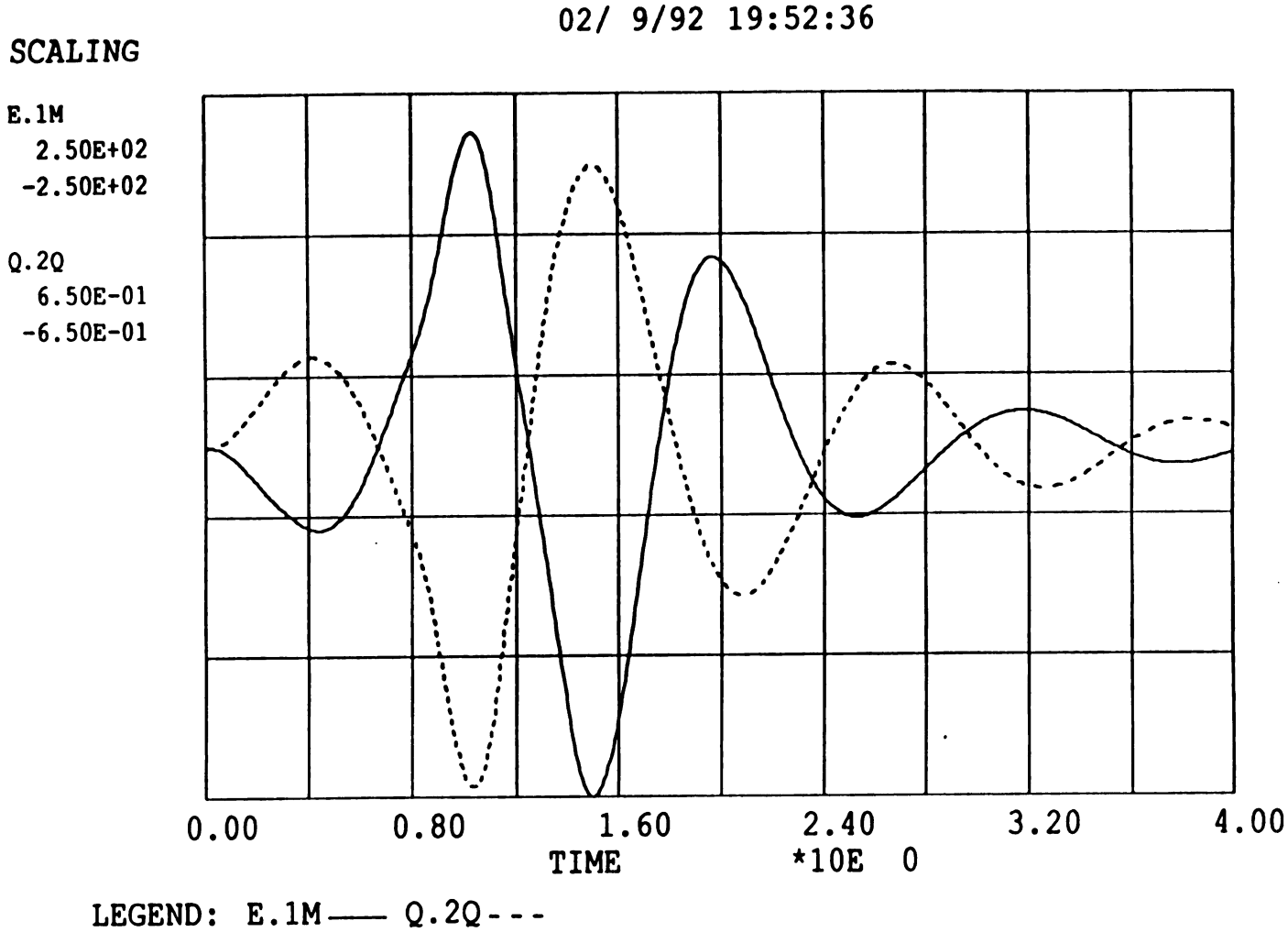


Figure 3.5.1.1
Objective Function and Active Constraint Responses to Initial Design Variables: Problem 1

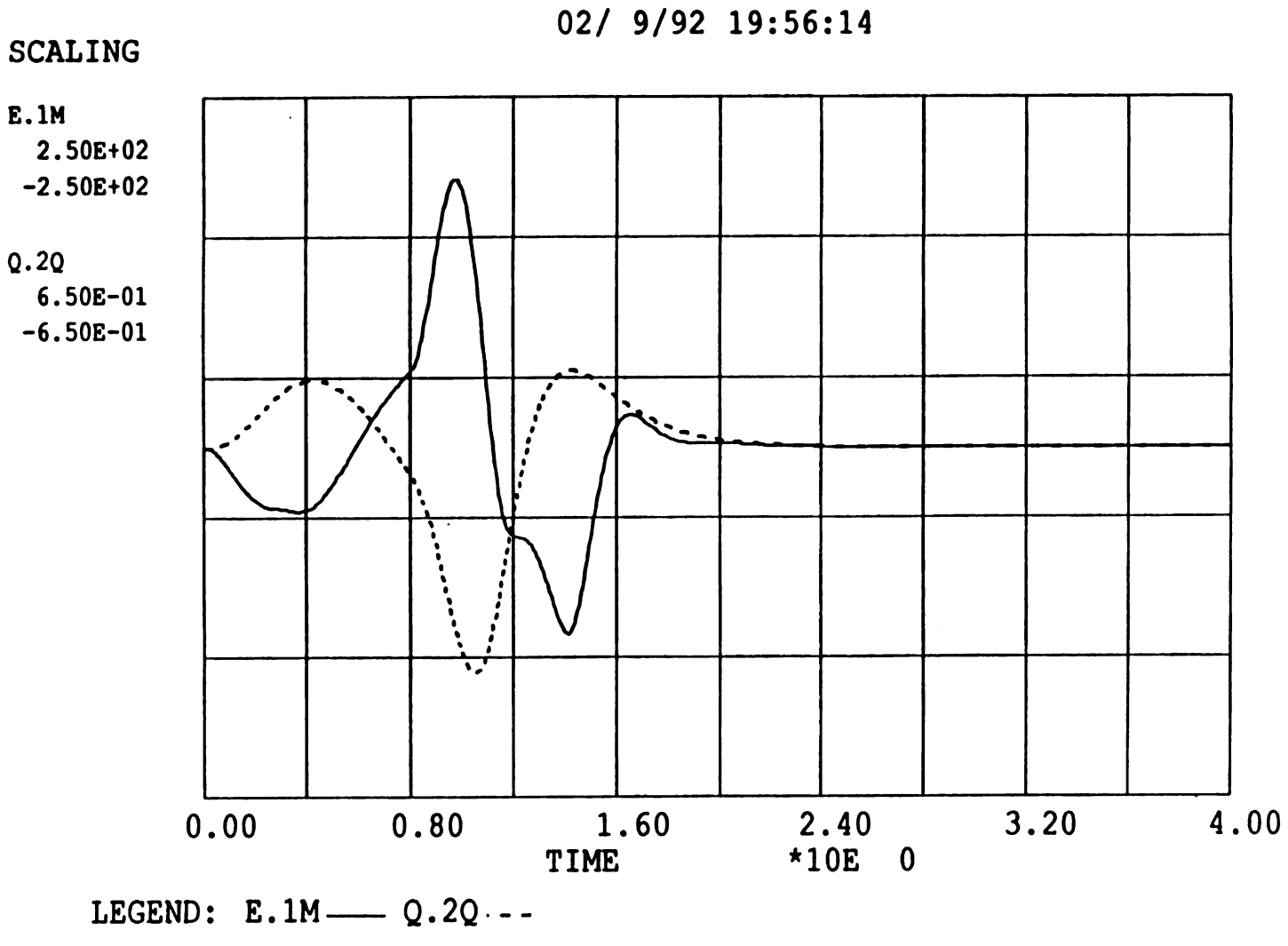


Figure 3.5.1.2
Objective Function and Active Constraint Responses to Optimal Design Variables: Problem 1

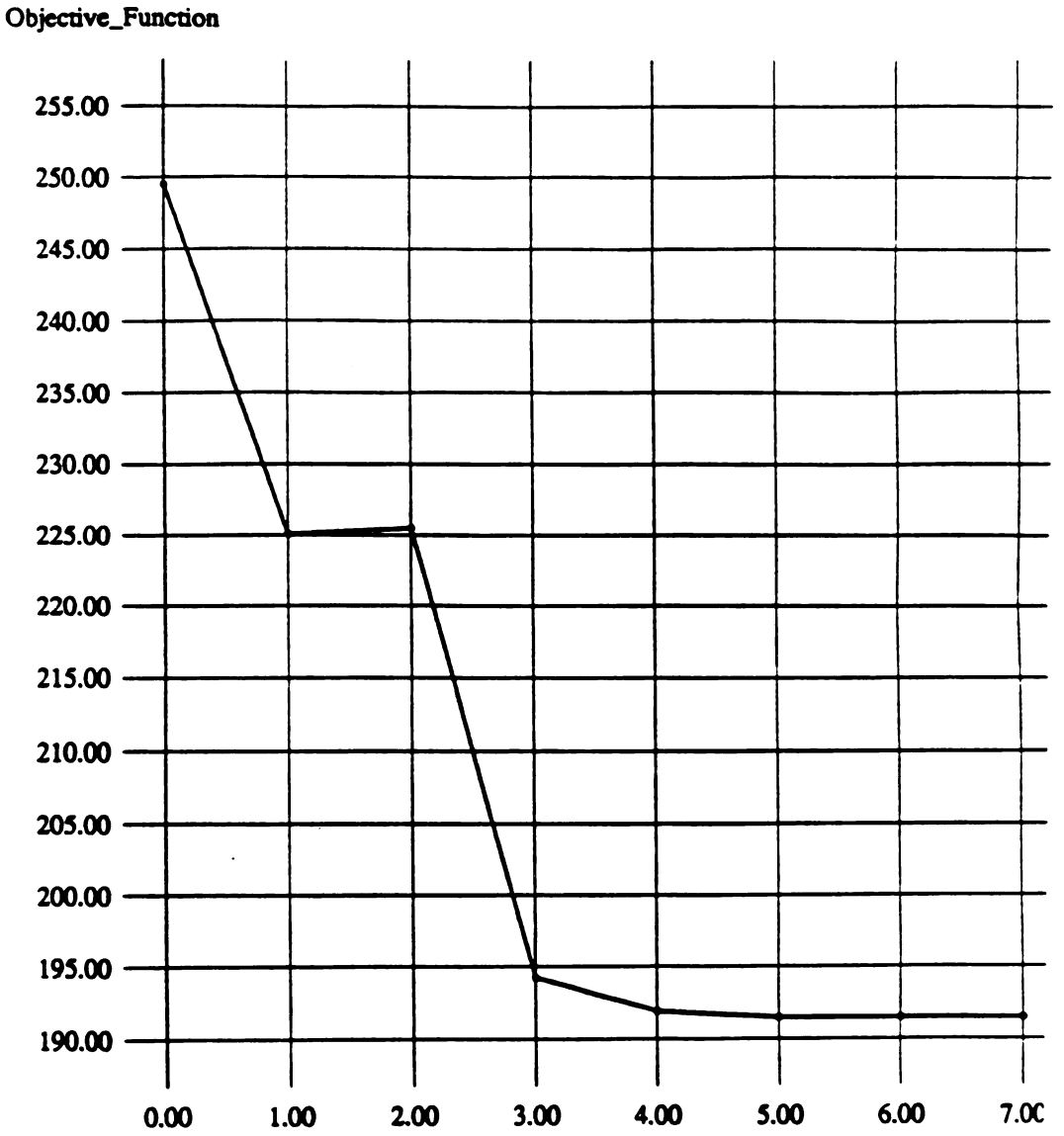


Figure 3.5.1.3
Objective Function Iteration History: Problem 1

3.5.2 Problem 2 Summary

The results (see Table 3.5.2.1) in problem 2 are very similar with a 3.4% difference in seat force values. The front damper values had the largest difference of 39.4%. These results were produced using the GRG algorithm in 7 iterations (see Figure 3.5.2.5). Road profile 2 produced the highest seat force at initial DV values. Figure 3.5.2.1 illustrates the initial seat force response. The active seat force for optimized DV values is created by road profile 3. Figure 3.5.2.3 shows the optimized seat force response. The active constraint is the rear tire compression for road profile 3. Figures 3.5.2.2 and 3.5.2.3 shows the tire compression for initial and optimized DV values respectively.

Table 3.5.2.1
Problem 2 Results

AV	HAUG & ARORA	OPTDES RESULTS	PERCENT DIFFERENCE
C1	600.0	600.0	0.00
C2	2400.0	2400.0	0.00
C3	2400.0	2400.0	0.00
R1	107.2	105.0	- 2.1
R2	551.0	333.9	- 39.4
R3	453.7	439.6	- 3.1
OBJ FCN	94.1	97.3	3.4

02/06/26 21:06:43

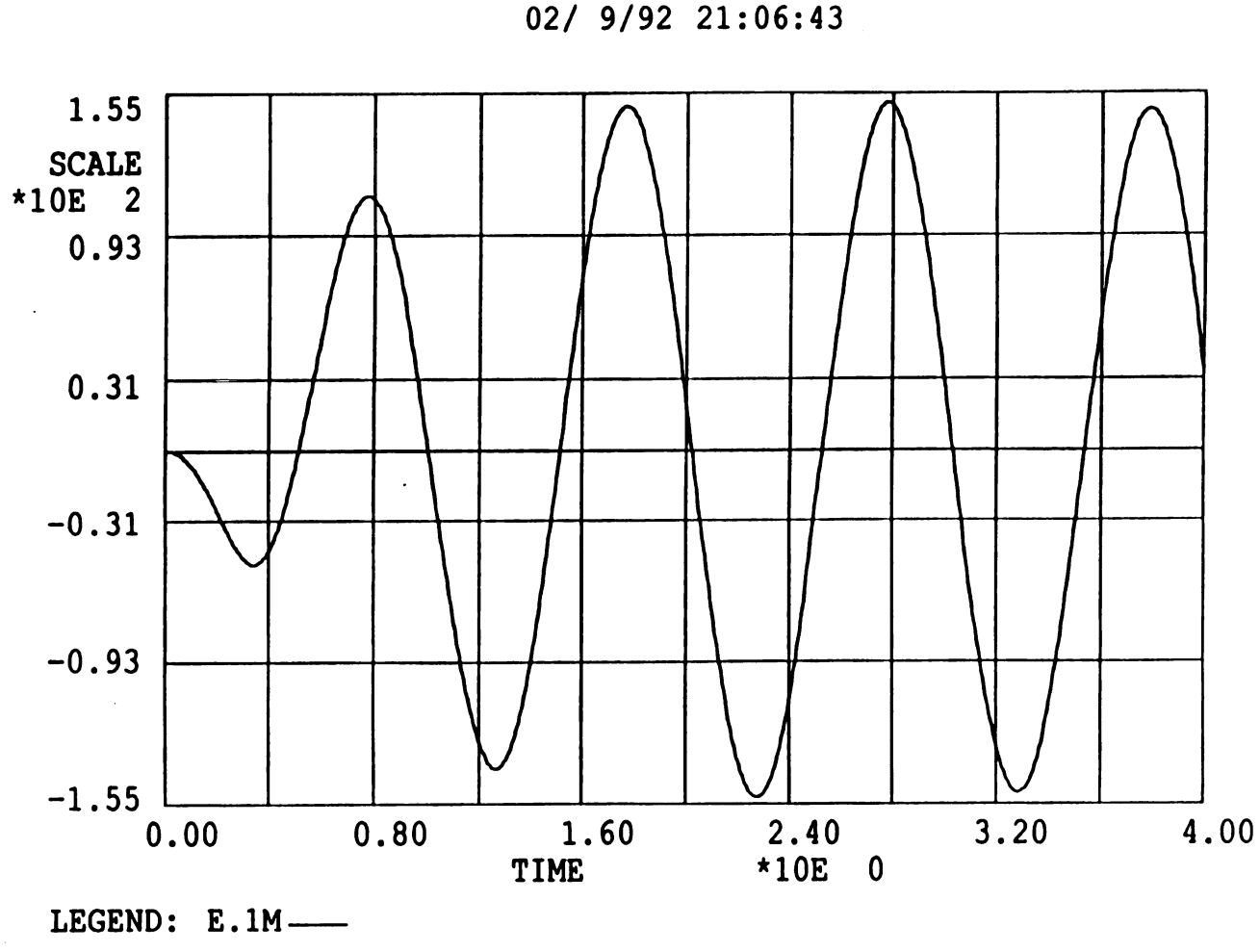


Figure 3.5.2.1
Objective Function Response to Initial Design Variables: Problem 2

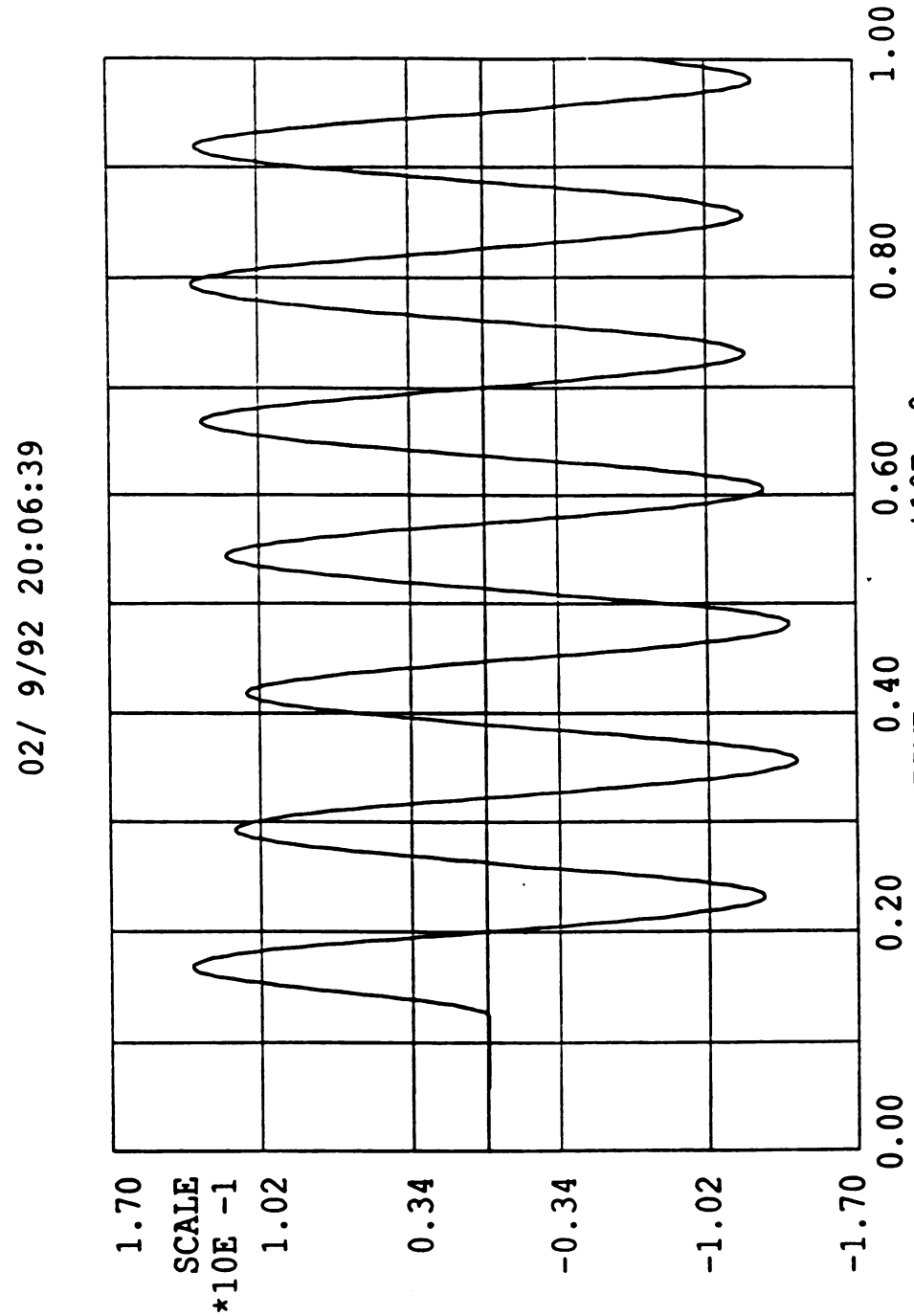


Figure 3.5.2.2
Active Constraint Response to Initial Design Variables: Problem 2

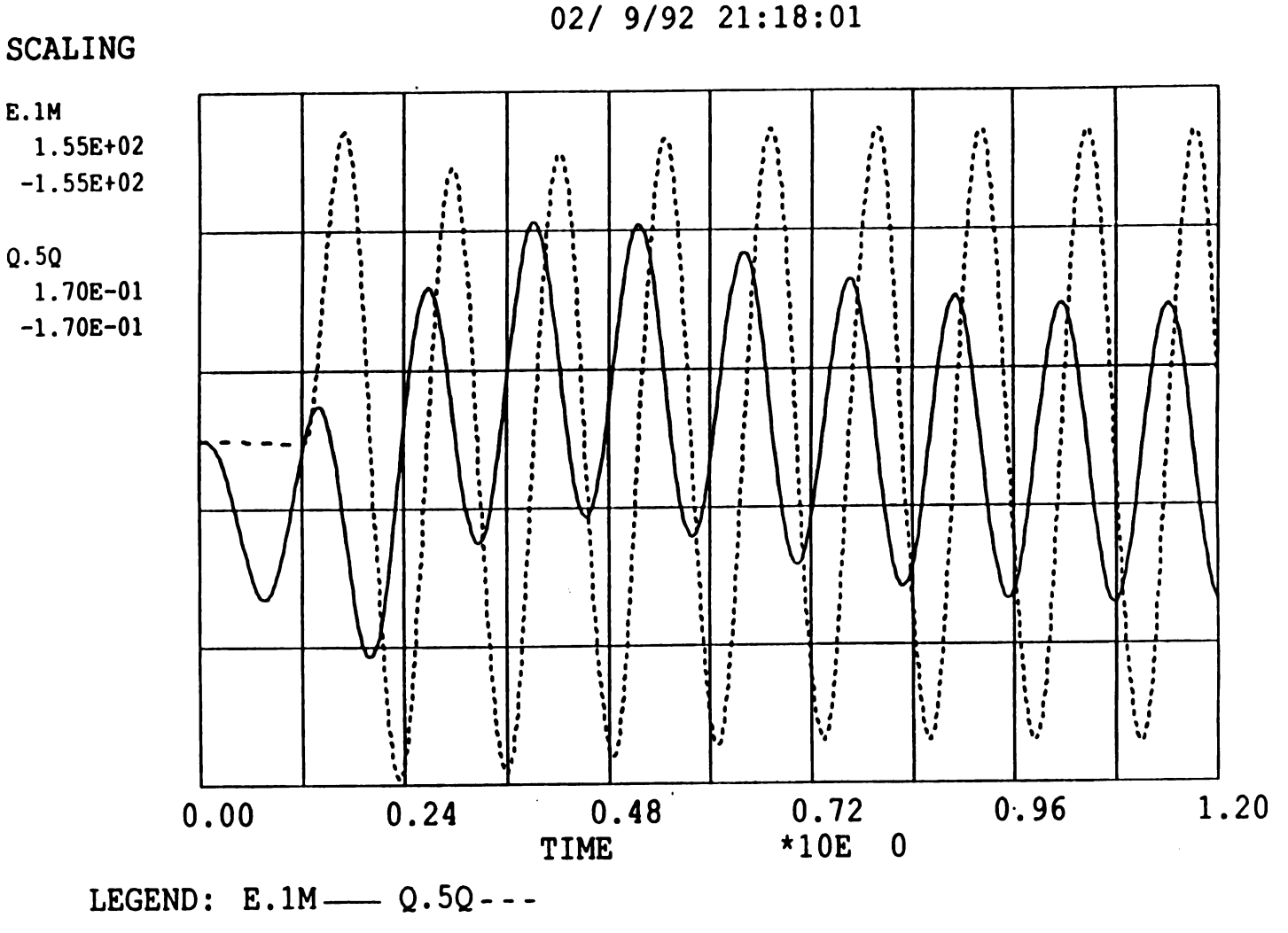


Figure 3.5.2.3
Objective Function and Active Constraint Responses to Optimal Design Variables: Problem 2

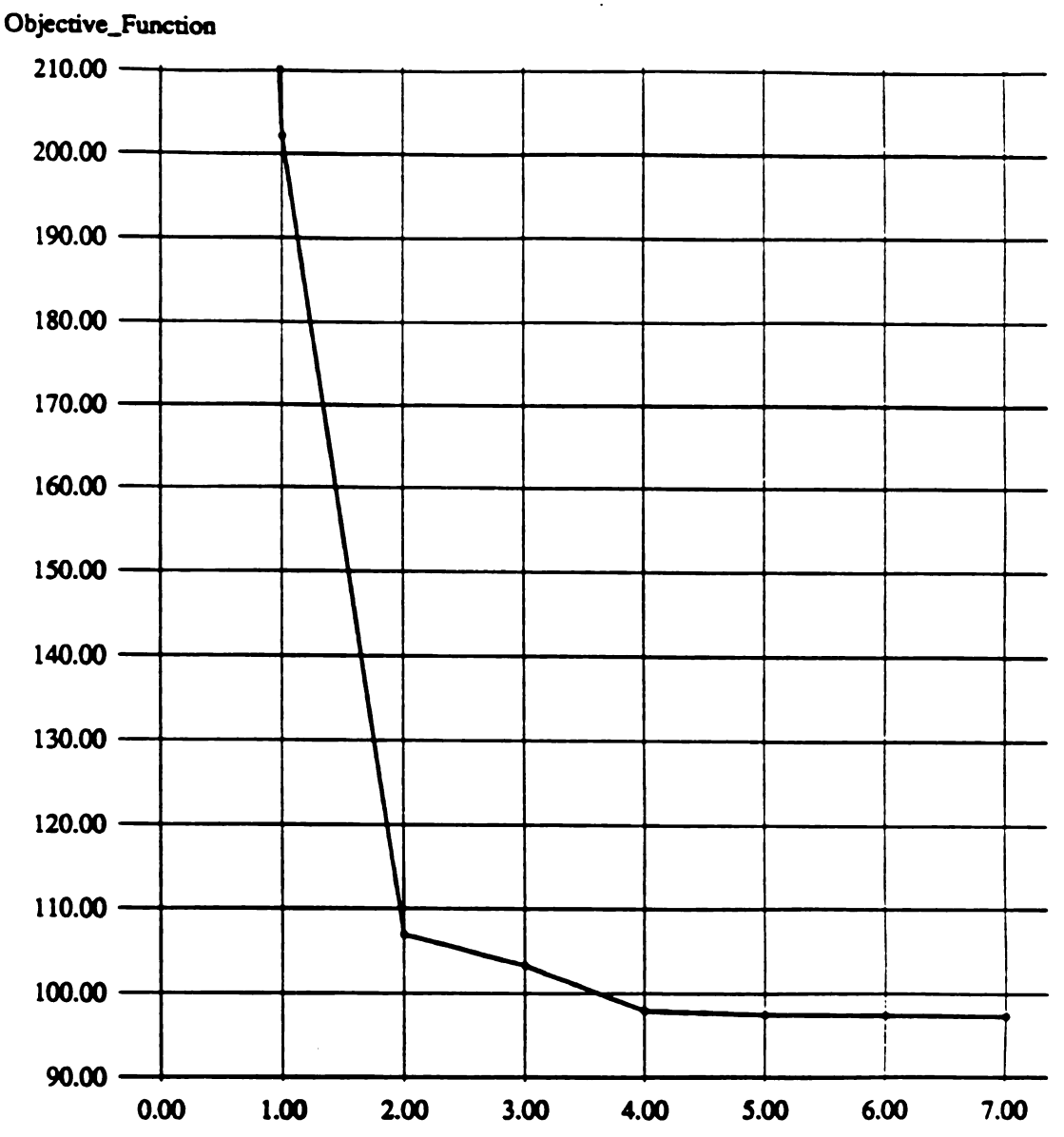


Figure 3.5.2.4
Objective Function Iteration History: Problem 2

3.5.3 Problem 3 Summary

The results for problem 3, see Table 3.5.3.1, show a difference in seat force values of approximately 4%. The AV values were very close except for the rear damper which had a 205% difference. In this problem the dependent constraints bounding the feasible set were not active at the optimal design point. The active constraints were the simple bounds for each DV at the optimal design point. Each spring was at its lower bound and each damper was at its upper bound. Figure 3.5.3.1 illustrates the seat force for initial DV values. Figure 3.5.3.1 shows the seat force for optimized DV values. These results were produced using the GRG algorithm after 8 iterations (see Figure 3.5.3.3). In this problem, as in problem 1, the front spring displacement constraint is violated (see Figure 3.5.1.1) for the initial DV values. Figure 3.5.3.2 shows the front spring displacement for the optimized DV values.

Table 3.5.3.1
Problem 3 Results

AV	HAUG & ARORA	OPTDES RESULTS	PERCENT DIFFERENCE
C1	600.0	600.0	0.00
C2	2400.0	2400.0	0.00
C3	2400.0	2400.0	0.00
R1	559.0	600.0	6.84
R2	941.3	960.0	1.99
R3	314.0	960.0	205.00
OBJ FCN	77.6	74.4	- 4.11

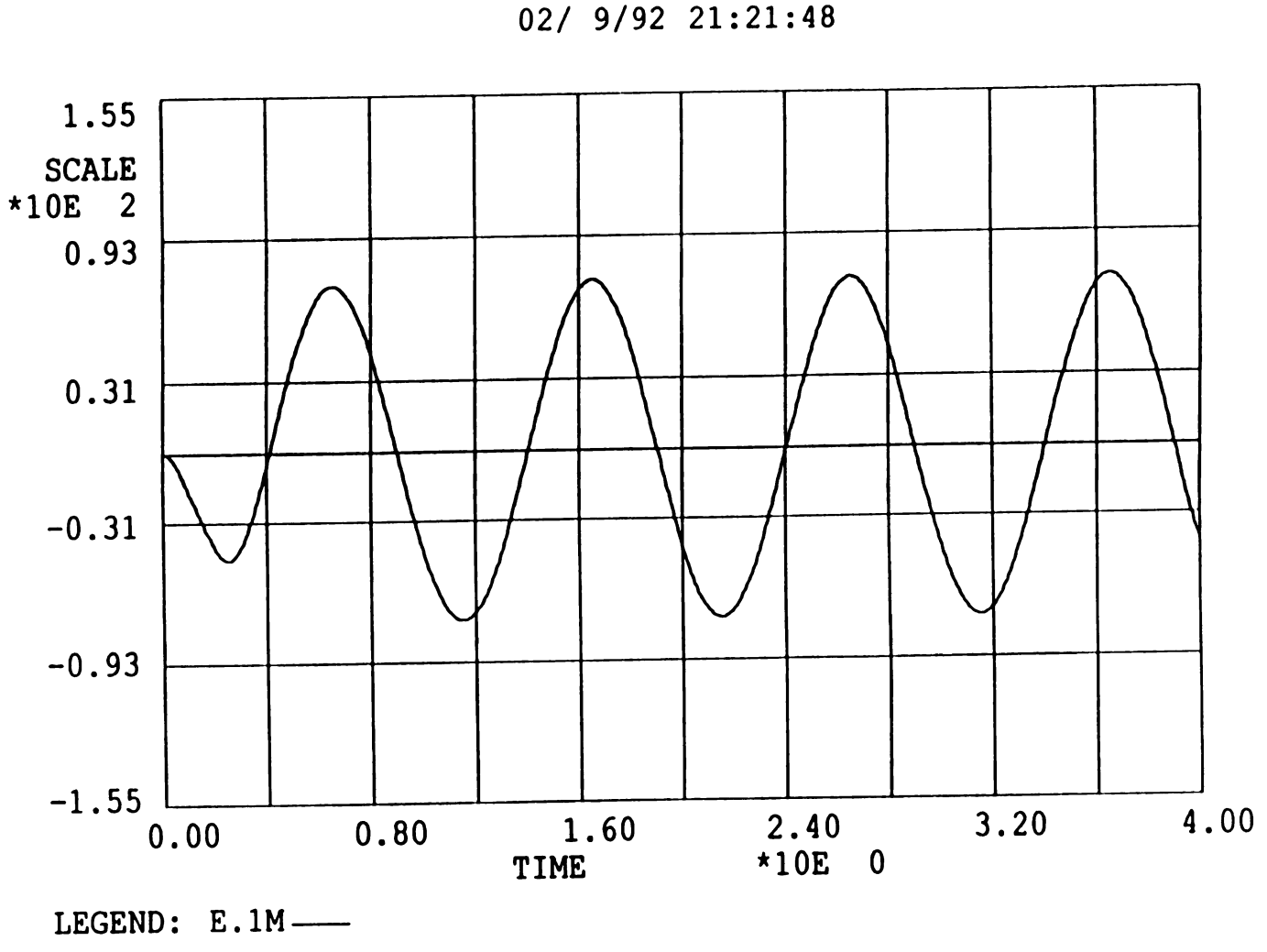


Figure 3.5.3.1
Objective Function Response to Optimal Design Variables: Problem 3

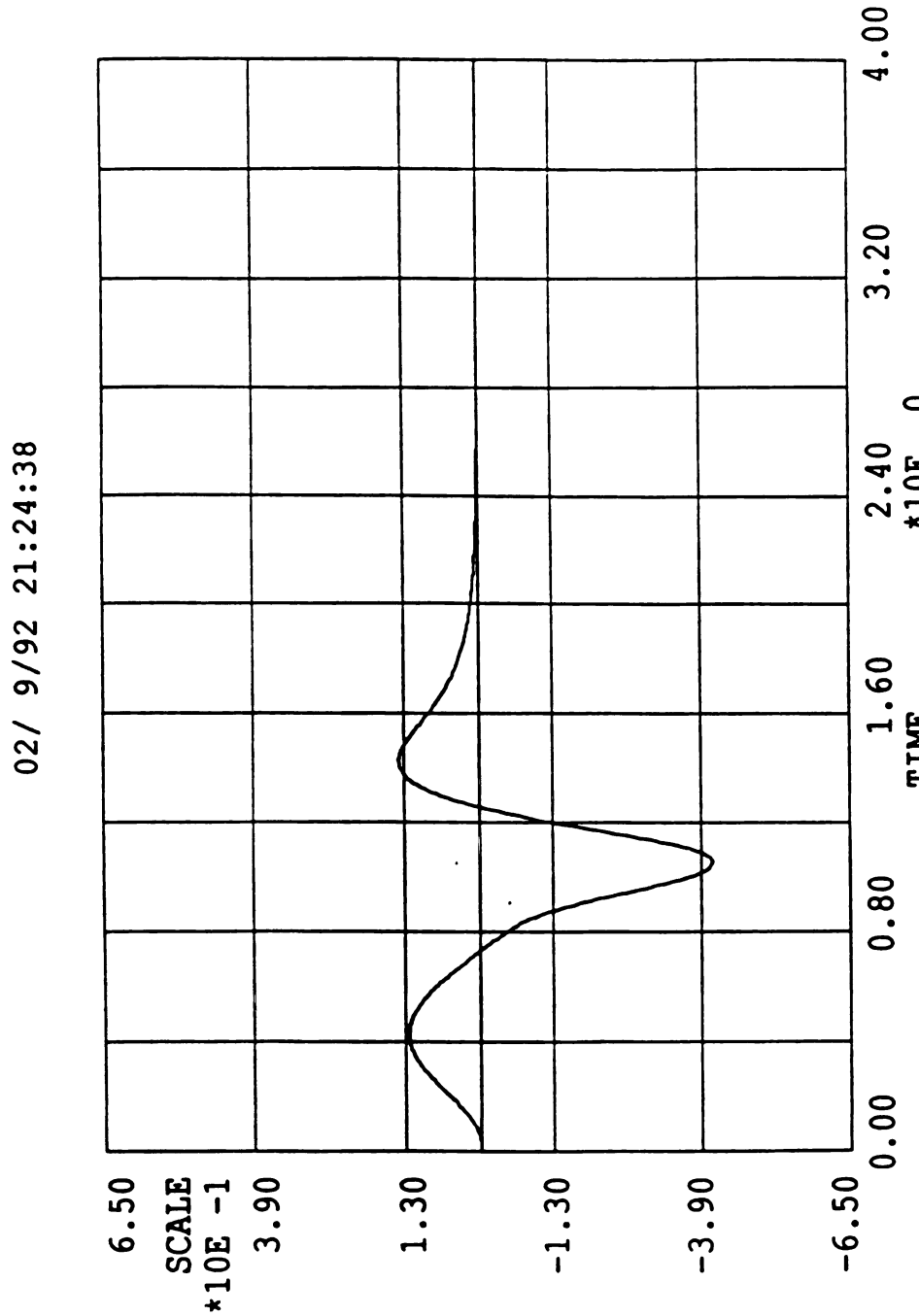


Figure 3.5.3.2
Active Constraint Responses to Optimal Design Variables: Problem 3

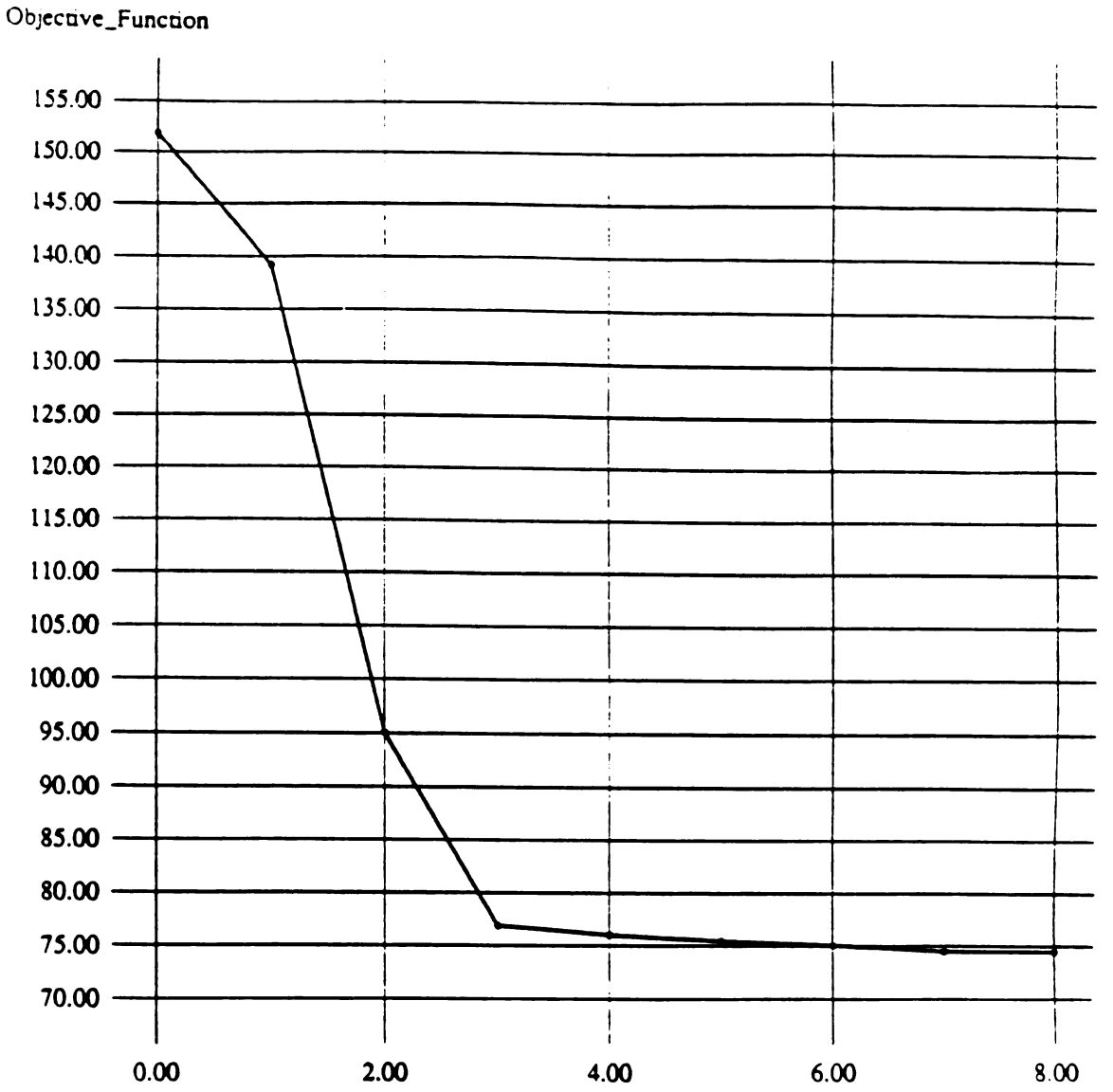


Figure 3.5.3.3
Objective Function Iteration History: Problem 3

3.6 Discussion

In each problem the optimal DV values were found using the GRG algorithm. The search parameters were not changed from their default settings. In a problem of this nature there are many local minimums. Different optimization algorithms will step through the design space differently. As a result different optimal solutions will be found. This is one explanation for the differences in results between OPTDES and the optimization method used by Haug and Arora. To illustrate this idea further problem 2 was solved using a combination of SQP and GRG algorithms. The problem was started using the SQP method. On the 15^{th} design the GRG method was used to complete the optimization problem. As seen from the results in TABLE 3.6.1 an entirely new solution was found.

Table 3.6.1
Problem 2 Results Comparison

AV	OPTDES RESULTS 1	OPTDES RESULTS 2	PERCENT DIFFERENCE
C1	600.0	2711.9	352.0
C2	2400.0	2400.0	0.00
C3	2400.0	2400.0	0.00
R1	105.0	35.0	-66.7
R2	333.9	278.9	-16.5
R3	439.6	418.9	-4.7
OBJ FCN	97.3	92.4	-5.0

The seat force was reduced by 5%. The most interesting differences were found in the seat spring and damper. This example shows how important it is for the designer to employ good optimization techniques to thoroughly research a problem before making any determinations.

CHAPTER 4

CONCLUSIONS

4.1 Summary

An interface between modeling software and optimization software has been established using ENPORT and OPTDES.BYU. With this interface dynamic responses can be optimized with respect to system components (e.g. springs, masses, and dampers). The interface uses the equation solving capabilities of ENPORT with the design space exploration facilities of OPTDES. The interface has been generalized to optimize any bond graph model with minimal user input. Also the designer can develop a library of designs to call upon. With this interface design time can be reduced while maintaining confidence in the results.

4.2 Recommendations

The interface established here is just the first step in fully integrating simulation and optimization software. Through use of this interface and advancements in the areas of simulation and optimization ideas for improvement will result. In developing this interface some of the following thoughts about improvements have come to mind:

- Use initial design conditions from ENPORT to create the datafile in CONVEN. This will help to reduce time and redundant effort.

- Create a standardized format in ANAPOST to write a file of optimized design variables. This file could be used to pass these values back to ENPORT.

- Expand the numerical integration capabilities of ANAFUN. This could be done by formatting ANAFUN to call the ENPORT integration library.

- Eliminate prompts for NX, NY, and NU if confidence is high for the accurracy of the values now being passed.

- Develop new methods of performance evaluation for the system response.

APPENDICES

APPENDIX A

DESIGN AND IMPLEMENTATION CONSIDERATIONS

APPENDIX A

DESIGN AND IMPLEMENTATION CONSIDERATIONS

A.1 Introduction

When creating the bond graph and posing the optimization problem the user must

- **Select elements for design variables**
- **Determine how system performance is measured**

When selecting design variables in ENPORT those elements are identified using the named-parameter option. These elements create the NP vector. The system performance is determined by selecting model outputs. These outputs build the Y vector (See Figure 2.3.1).

A.2 Design Variables

The named-parameter feature plays a key role in properly building the bond graph for optimization. The named-parameter option describes an element used in a node equation by a variable name. Usage of this option is discussed in chapter 6 of the ENPORT Reference Manual on pages 6-4a and 6-4b. When the ENPORT Fortran file is generated, elements described by a named-parameter will be coded with the variable name. These elements will be passed values through the AV vector when the interface software calls the ENPORT Fortran file (See Figure 2.3.1.1). The relationship between the AV and NP vector is one to one.

$$\begin{vmatrix} AV(1) \\ AV(2) \\ \vdots \\ AV(n) \end{vmatrix} = \begin{vmatrix} NP(1) \\ NP(2) \\ \vdots \\ NP(n) \end{vmatrix}$$

The user must know the size and read order of the NP vector created in the ENPORT Fortran file. This information is needed when running CONVEN.

A.3 Analysis Functions

The interface is responsible for interpreting the time response (Y vector) of the model. The following performance measurement methods (PMM) are used to evaluate the model output

M	=	**Maximum Value**
A	=	**Average Value**
R	=	**RMS Value**
S	=	**Minimum Value**
X	=	**Absolute Maximum Value**
N	=	**Absolute Minimum Value**

Any combination of these can be selected for a problem. The size and order of the AF vector is a function of the output vector,Y, and the number of PMMs selected. The subroutine ANAFUN will call subroutine perform to evaluate the model response (Y vector).

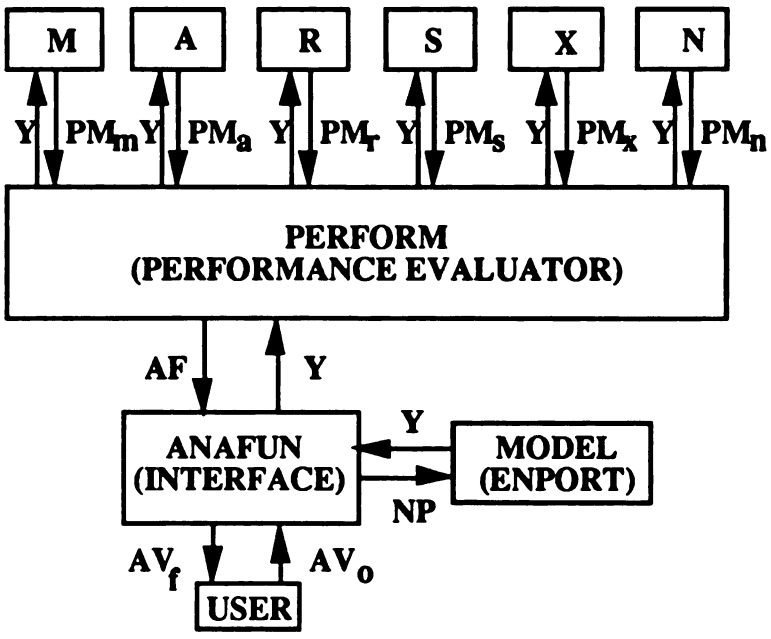


Figure A.3.1
Performance Evaluation Data Flow

Perform will call any of the PMM subroutines depending on which methods are selected. Perform will also call the PMM subroutines in the order they were selected. In each PMM subroutine the output response is analyzed and PM values are returned. The PMs are on a one to one basis with the Y vector.

$$\begin{vmatrix} PM(1) \\ PM(2) \\ \vdots \\ PM(n) \end{vmatrix} = \begin{vmatrix} Y(1) \\ Y(2) \\ \vdots \\ Y(n) \end{vmatrix}$$

As the PMM subroutines are called the PM vectors are assembled to form the AF vector

$$AF = \begin{vmatrix} PM_m \\ \vdots \\ PM_n \end{vmatrix}$$

Hence the size of the AF vector is

#AF members = # Y members * #PMMs selected

The AF vector is sorted first by PMM then by PM. Therefore, if

PMM = 2
PM = x 3
AF = 6

and AF will be sorted as

AF(1) = PM(1) |
AF(2) = PM(2) |-----**For PMM(1)**
AF(3) = PM(3) |

AF(4) = PM(1) |
AF(5) = PM(2) |-----**For PMM(2)**
AF(6) = PM(3) |

The size and order of AF must be known for setting constraints in SETUP. As a service to the user an option to view the AF vector is available when executing the interface software.

A.4 General Model Design

Up to this point the discussion has centered around building a specific model for a specific optimization problem. However, using the named-parameter option to its fullest extent the user can build generalized models to solve many different optimization problems. The named-parameter option can be used to define physical parameters besides design variables. These physical parameters are passed values using the system input vector U. These values are entered once before optimizing and remain fixed. They are not design variables. Referring to figure 2.3.1.1 and

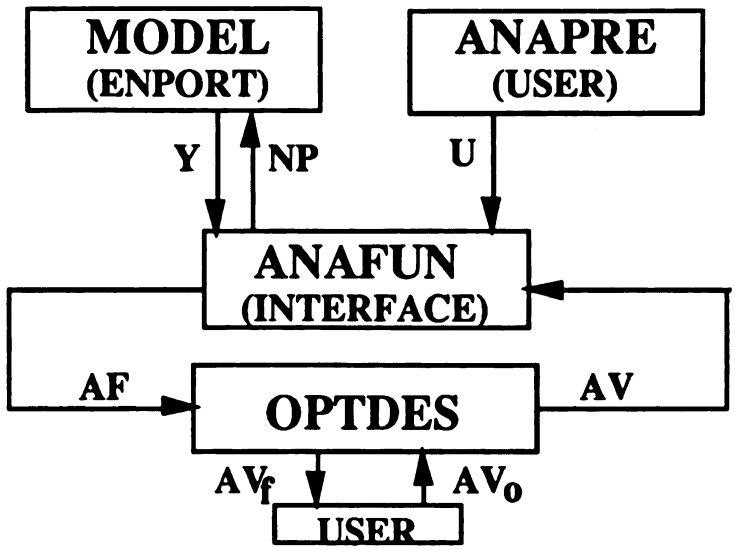


Figure A.4.1
User Entry of System Parameters

expanding this diagram to the one seen in Figure A.4.1. In this figure it can be seen that subroutine ANAFUN will call subroutine ANAPRE. This call is made one time to collect information necessary to execute the optimization process. At this point the user can enter information which profiles the bond graph model being optimized.

Consider the mechanical position control system in Figure 1.1.1. Suppose F(t) and m are fixed and K1 and C1 are the DVs. The ENPORT model could be built accordingly and optimized. Now suppose the mass was changed to a new fixed value. A new model would be generated using the new mass value. On the other hand a general bond graph can be created identifying the mass using the named-parameter option. The mass is identified as a member of the U vector (declared as external in ENPORT). Now the user can simply re-enter a new mass value, thru ANAPRE, to execut a new optimization problem in lieu of creating a new bond graph. The U vector values can be entered using a datafile or interactively. See appendix D.2 EXECUTING ANAPRE for details. Also see appendix E.2 for creating an interactive menu. It is recommended that the user begin by entering data using a datafile.

APPENDIX B

MULTIPLE FORCING SYSTEM PROBLEM

APPENDIX B

MULTIPLE FORCING SYSTEM PROBLEM (MFS)

B.1 Introduction

This is a special class of problem and requires the usage of a general model design. Suppose the mechanical positioner is designed to minimize settling time. In addition the postioner can cycle at two different frequencies; 60 and 30 hertz (assume a constant amplitude). Now the system must be optimized to satisy two different forcing functions. In this case the same force is applied at two different natural frequencies such as

$$F_1(t) = F_0 \sin \omega_1 t$$

$$F_2(t) = F_0 \sin \omega_2 t$$

The best design possible is when the system has the same settling time at each frequency.

B.2 Posing the MFS Optimization Problem

The problem would be posed as

$$\text{Min} \quad \Psi$$

where

$$\Psi = \text{Max}\{t_1, t_2\}$$

t_1 = Settle time at ω_1

t_2 = Settle time at ω_2

Subject to Constraints

$$\Phi^i_{jk_{min}} < \Phi^i_{jk} < \Phi^i_{jk_{max}}$$

and Simple Bounds

$$\Theta_{g_{min}} < \Theta_g < \Theta_{g_{max}}$$

where

$i = 1,2, \ldots l$; l = no. of forcing systems

$j = 1,2, \ldots m$; m = no. of performance meas. methods

k = 1,2, ... n; n = no. of performance measurements

Φ^i_{jk} = k^{th} constraint response for the j^{th} performance measurement method for the i^{th} forcing system.

Θ_g = g^{th} design variable.

B.3 AF-Vector Build for the MFS Problem

This type of problem adds another dimension to the AF vector. For each set of driving conditions subroutine ANAFUN will be repeated, hence the AF vector will grow in magnitude of order n. The user must be aware of this to pose the optimization problem in CONVEN and SETUP. Consider the example in appendix A.3. With two forcing systems the AF vector would now have 12 members

#AF = { 2(=l) x 2(=m) x 3(=n) }

	AF(1) = PM(1)			
	AF(2) = PM(2)	------	PMM(1)	
*	AF(3) = PM(3)			NFF(1)
	AF(4) = PM(1)			
	AF(5) = PM(2)	------	PMM(2)	
*	AF(6) = PM(3)			
	AF(7) = PM(1)			
	AF(8) = PM(2)	------	PMM(1)	
*	AF(9) = PM(3)			NFF(2)
	AF(10) = PM(1)			
	AF(11) = PM(2)	------	PMM(2)	
*	AF(12) = PM(3)			

Figure B.3.1
Example of MFS AF Vector

B.4 Setting up the MFS Problem in OPTDES

Now suppose the third PM is used to measure the objective performance then the objecitve function set, Ω, is

$$\Omega = \{AF(3), AF(6), AF(9), AF(12)\}$$

$$\Psi = \text{Max } \{\Omega\}$$

To solve this problem an additional AF member is required. This member is Ψ. Hence, from Figure B.3.1 the additional AF member is

$$AF(13) = \Psi$$

Also an additional AV member is required. This value will be referred to as Λ. Λ is used as the objective function. Now using the mechanical positioner where C1 and K1 are the AVs then the additional AV member is

$$AV(3) = \Lambda$$

Λ must have an arbitrary initial value (Λ_0) that is at least greater than all members of the objective function set. Then in this example

$$\Lambda_0 > \text{Max } \{\Omega\}$$

This initial value for Λ_0 is entered the same as all AV values when CONVEN is executed.

Now in SETUP Λ can have bounds of

$$0 < \Lambda < \Lambda_0$$

The objective function set members are set less than zero in SETUP where

$$\begin{aligned} AF(3) &< 0 \\ AF(6) &< 0 \\ AF(9) &< 0 \\ AF(12) &< 0 \end{aligned}$$

The additional AF member is declared the objective function in SETUP where

v AF(13)

B.5 Determining the Objective Function Value

Recall that in this example the third PM was used to measure the objective performance which is why AF(3), AF(6), AF(9), and AF(12) are the objective function set members (refer to Figure B.3.1).

$$\begin{aligned} AF(3) &= \Phi^i_{jk} - \Lambda \quad \text{where i=1, j=3, k=1} \\ AF(6) &= \Phi^i_{jk} - \Lambda \quad \text{where i=1, j=3, k=2} \\ AF(9) &= \Phi^i_{jk} - \Lambda \quad \text{where i=2, j=3, k=1} \\ AF(12) &= \Phi^i_{jk} - \Lambda \quad \text{where i=2, j=3, k=2} \\ AF(13) &= \Psi = \Lambda^2 \end{aligned}$$

Λ is an independent variable which must be lowered to minimized Ψ. Λ will have lower bounds set by the constraint set Ω. These OCs, Ω, will become active as Λ is lowered. The OCs are functions of the AVs. Hence the optimization algorithm will adjust the AVs to lower the OC values in order to lower Λ.

B.6 Summarizing the MFS Optimization Problem

These calculations are performed internally and are based on responses entered by the user when executing the optimization software. Therefore the user is not responsible for understanding or setting up this calculation process. The user must be aware that they are posing an MFS problem. They must know which PM is

measuring the OP and how to determine the proper AV and AF sizes. To summarize the user must do the following for

ANAPRE

- **Enter the number of forcing systems (NFF)**
- **Enter the size of the Objecive Function Set (generally equals NFF)**
- **Enter index No. for PM member measuring the OP**

CONVEN

- **Calculate and enter the size of the AV vector**

  #AV = #NP + 1

- **Calculate and enter the size of the AF vector**

  #AF = #PMM x #PM x #NFF + 1

SETUP

- **Declare the additional AF member as the Objective Function**
- **Set the additional AV member with the proper bounds in SETUP**

Internal calculations are based on entries to these prompts. See appendix G.5 to see how this information was entered for the application example in chapter 3. The multiple forcing system procedure described here was used in problem 2.

Note that in problem 3 of the application example the objective function set size was 1. In this problem the first forcing system was used to determine constraints only. The overall technique previously described to determine the objective function is not necessry. Now AV and AF are determined as follows:

#AV = #NP

#AF = #PMM x #PM x #NFF.

APPENDIX C

USER WRITTEN SUBROUTINES AND SAMPLES

APPENDIX C

USER WRITTEN SUBROUTINES

C.1 Introduction

These options are available to expand the capabilities of the interface as requirements become more sophisticated. With this option the user can write E/F models, create menus for an interactive session and build the AF vector to desired specifications. The checkfile option should be used when debugging these subroutines.

C.2 User Written Generator (USRGEN.FOR)

This subroutine is used to model Effort/flow systems. USRGEN.FOR must be written to return the instantaneous Effort/flow value for each source. When creating the ENPORT Fortran file each source of effort or flow is identified as a single external source. Hence if there are 2 sources of flow (as in the application example in chapter 3) driving the model USRGEN will generate and return a U(1) and U(2) value. USRGEN can also be used in MFS problems. The subroutine statement must be written as:

SUBROUTINE USRGEN (NFFCN,T,U)

Where

Nffcn = forcing system No.
t = time
U = U - vector

Nffcn and t are passed to USRGEN and U is returned. A sample program is included. This sample will generate the road profiles used in the application example in chapter 3.

C.2.1 Sample Program: USRGEN

FORTRAN FILE

USRGEN.FOR

WRITTEN BY

MARK DAVIS

```
c.************************************************************
c.----Subroutine USRGEN:
c.
c.---Purpose: This subroutine serves as an interface between the
c.            subroutine Anasub.for and the user.  This user may
c.            code in values or equations to describe external
c.            inputs to the system.
c.
c.---Variables used:
c.   Passed variables
c.   -------------------------------------
c.     T  = The instantaeous time
c.     U  = The external function vector
c.
c.   Local variables
c.   -------------------------------------
c.     tlag = Time lag between front and rear wheels
c.     tr1  = Initial response time for the rear wheel
c.     tr2  = Second phase response time for the rear wheel
c.     tf2  = Second phase response time for the front wheel
c.     tt   = Initial phase response time with time delay
c.
       subroutine USRGEN(nffcn,t,u)
c.
c. Define passed variables
c.
       double precision t,u(1)
c.
c. Define local variables
c.
       double precision tlag, tr1, tr2, tf2, tt, w1, w2,
     +                  MASS(20),TF(20)
c.
       REAL sp, wb, l1, l2, ya
       Integer iwave, nffcn

       Common /extmen/ sp(10), wb(10), l1(10), l2(10), ya(10),
       iwave(10)
c.
c. Determine velocity input for the front wheel.  The time .48s
c. represents the
c. phase shift necessary for the second cos function to begin
c. at 180 degrees.
c. This will match the asymptotes of the two functions. .48 is
c. of the second cos function.
c.
c. Natural frequencies
c.
       pi = 3.14159265d0
       w1 = pi*sp(nffcn)/l1(nffcn)
       w2 = pi*sp(nffcn)/l2(nffcn)
c.
c. Periodic times
c.
       t1 = l1(nffcn)/sp(nffcn)
       t2 = (l1(nffcn)+l2(nffcn))/sp(nffcn)
c.
c. Front wheel initial velocity
c.
       Ifin = Iwave(nffcn)
```

```
DO 10 ISW = 1,Ifin
          IPER = ISW
            IF(t .le. isw*t2)goto 20
 10     continue
 c.
 20     t1 = iper*t1+(iper-1)*t2
        t2 = iper*t2
 c.
        if (t .lt. t1) then
          u(1) = -Ya(nffcn)*w1*sin(w1*t)
        elseif (t .le. t2) then
          u(1) =  Ya(nffcn)*w2*sin(w2*(t-t1))
        else
          u(1) = 0.
        endif
 c.
 c. Determine velocity input for the rear wheel. The is a time
 c. lag between these inputs equal to:
 c.                      time lag = (wheel base)/(speed of car)
 c.
        tlag = wb(nffcn)/sp(nffcn)
        tr1  = t1 + tlag
        tr2  = t2 + tlag
 c.
        if (t .lt. tlag) then
          u(2) = 0
        elseif (t .lt. tr1) then
          tt = t - tlag
          u(2) = -Ya(nffcn)*w1*sin(w1*tt)
        elseif (t .le. tr2) then
          u(2) =  Ya(nffcn)*w2*sin(w2*(t-tr1))
        else
          u(2) = 0.
        endif
 c.
 C. SET REMAINING EXTERNAL INPUTS
 C.
        MASS(1)=9.01
        MASS(2)=139.75
        MASS(3)=3416.7
        MASS(4)=3.0
        MASS(5)=3.0
 C.
        TF(1)=.8333333
        TF(2)=3.333333
        TF(3)=6.666667
 C.
        U(3) =MASS(1)
        U(4) =MASS(2)
        U(5) =MASS(3)
        U(6) =MASS(4)
        U(7) =MASS(5)
        U(8) =TF(1)
        U(9) =TF(2)
        U(10)=TF(3)
        return
        end
```

C.3 User Written Menu (USRMEN.FOR)

This subroutine is used to read in values necessary to execute USRGEN. The option is available to the user to enter data via the keyboard or by data files. A common storage file must be used to pass data from USRMEN to USRGEN. The name and formatting of the common file is left to the programmer. This file does not receive or return any variables via the subroutine statement. This file must also include the integration variable common statement INTVAR if data is collected for multi-forcing system (NFF = # of forcing functions)

COMMON/INTVAR/T1,T2,h,XIN(20),NX,NY,NS,NU,NFF,IEND,UAF

A sample program written to operate in conjunction with USRGEN is included. This sample is written to be used with the generator described in C.2.

C.3.1 Sample Program: USRMEN

FORTRAN FILE

USRMEN.FOR

WRITTEN BY

MARK DAVIS

```
c.*************************************************************
c.----Subroutine USRMEN
c.
c.----Purpose: This subroutine allows the user to create a more
c.             interactive model of the external inputs.  This
c.             program can be called from anapre allowing for
c.             different conditions to be used without
c.             re-programming subroutine ENPGEN.
c.
c.----Variables used:
c.          Passed variables
c.          ---------------------------------------
c.              sp  = vehicle speed
c.              wb  = wheelbase of vehicle
c.              l1  = 1st length of road undulation
c.              l2  = 2nd length of road undulation
c.              Ya  = amplitude of road undulation
c.              isw = number operiods
c.
        Subroutine USRMEN
c.
c. Define passed variables
c.
        Dimension sp(10),wb(10),l1(10),l2(10),ya(10),iwave(10)
        Character iagn*1
        Real sp, wb, l1, l2, ya
        Integer iwave
c.
        Common /intvar/ T1,T2,h,xin(20),nx,ny,ns,nu,nff,iend,uaf
        Common /extmen/ sp, wb, l1, l2, ya, iwave
c.
        do 20 i = 1,nff
10      Write (*,102) i
          Write (*,100)
          Read  (*,*) sp(i), wb(i), l1(i), l2(i), ya(i), iwave(i)
          write (*,101) sp(i), wb(i), l1(i), l2(i), ya(i),
     +          iwave(i)
          Read  (*,200) iagn
          If (iagn .eq. 'y') goto 10
20      continue
c.
100     format (1x,'Enter the following simulation parameters',/,
     +          1x,'       1 - Vehicle Speed',/,
     +          1x,'       2 - Wheelbase',/,
     +          1x,'       3 - Half-wavelength # 1',/,
     +          1x,'       4 - Half-wavelength # 2',/,
     +          1x,'       5 - Amplitude',/,
     +          1x,'       6 - No. of periods',/,
     +          1x,'  ',$)
101format(1x,'You entered the following simulation parameters',/,
     +          1x,'      Vehicle Speed      =',f7.2,/,
     +          1x,'      Wheelbase          =',f7.2,/,
     +          1x,'      Half-wavelength #1 =',f7.2,/,
     +          1x,'      Half-wavelength #2 =',f7.2,/,
     +          1x,'      Amplitude          =',4x,f3.1,/,
     +          1x,'      No. of periods     =',4x,i3,//,
     +          1x,'Do you wish to re-enter these values',/,
     +          1x,'      y = yes        n = no',/,
     +          1x,'             ',$)
```

```
102     format (1x,'Your are entering loading conditions no.',i3)
c.
200     format (a1)
c.
        return
        end
```

C.4 User Written AF - Vector Build (USRAFN.FOR)

This subroutine allows the user to build the AF-vector as desired. This subroutine is used independently of the other user written subroutines. The subroutine statement must read as:

SUBROUTINE USRAFN(NFFCN,NPM,AV,AF)

Where

NFFCN = Forcing system Number
NPM = Number of Performance Measurements
AV = Analysis Variable Vector
AF = Analysis function Vector

Variables NFFCN, NPM, and AV are passed to USRAFN and AF is returned. A sample is included. This sample was written to use with an MFS problem. In this sample the AF vector is the size of the PM vector. Each PM value is compared for each forcing system and only the worst case value is returned. This is just one idea for how this subroutine could be used.

C.4.1 Sample Program: USRAFN

FORTRAN FILE

USRAFN.FOR

WRITTEN BY

MARK DAVIS

```
c*****************************************************************
c.----Subroutine USRAFN
c.
c.----Purpose: This subroutine determines the maximum value from
c.             the objective function set.  In addition all AF
c.             components which are elements of the obj fcn set
c.             will have their return values calculated.
c.             In addition the objective function value used by
c.             optdes will be calculated.
c.
c.----Variables used:
c.         Passed variables
c.         ---------------------------------
c.              af  = the analysis function array
c.              ny  = number of output variables
c.              nff = number of forcing functions
c.              npm = number of performance variables
c.              nobj= size of objective function
c.
        subroutine USRAFN(NFFCN,NPM,AV,AF)
c.
c. Define passed variables
c.
        Dimension zf(10),af(1),av(1),dumaf(10,10,100),obfcn(10)
        Double Precision dumaf, af, av
        Real zf, amax, acur
        Integer rfact, ncont, npm, naf, iavn, obfcn
c.
        Common /intvar/ T1,T2,h,xin(20),nx,ny,ns,nu,NFF,IEND,UAF
        Common /objfcn/ nobj,obfcn,icstr,iavn
c.
c. Determine repeating factor and number of calculated af
c. constraints
        rfact = npm*ny
c.
c. Set AF vector
c.
        If (obfcn(1) .lt. 0) then
c.
c. Sort the af vector
c.
c. i = num of forcing functions
c. j = num of performance criteria
c. k = num of output variables
c.
          do 20 i = 1,nff
           do 20 j = 1,npm
            do 20 k = 1,ny
              iaf =  (i-1)*ny*npm +(j-1)*ny + k
20        dumaf(i,j,k) = af(iaf)
c.
c. Re-Initialize af
c.
        do 22 i=1,nff*npm*ny
22        af(i)=0
c.
c. Determine max values for each constraint
c.
```

```
          do 35 j = 1,npm
             do 30 k = 1,ny
               naf  = (j-1)*npm + k
               amax = dumaf(1,j,k)
                 do 25 i = 1,nff
                   acur = dumaf(i,j,k)
                   if(acur .gt. amax)then
                     af(naf) = acur**2
                   else
                     af(naf) = amax**2
                   endif
25               continue
30           continue
35         continue
c.
c. Set the AF vector obj fcn constraints
c.
c. Set obj fcn
c.
          AF(rfact+1) = AV(iavn)**2
c.
        ELSE
c.
c. Determine obj fcn subset
c.
           Do 50 i=1,nobj
             iaf = obfcn(i)
50           zf(i) = af(iaf)
c.
c. Determine Max value
c.
           amax = zf(1)
           Do 55 i=1,nobj
             if(zf(i) .gt. amax)then
              amax = zf(i)
             else
              amax = amax
             endif
55         continue
c.
c. Set the AF vector obj fcn constraints
c.
           Do 60 i=1,nobj
             iaf = obfcn(i)
60           AF(iaf) = AF(iaf) - AV(iavn)
c.
c. Set the Obj Fcn value
c.
          AF(rfact+1) = AV(iavn)**2
c.
        ENDIF
c.
        return
        end
```

APPENDIX D

HOW TO USE THE OPTIMIZATION INTERFACE

APPENDIX D

HOW TO USE THE OPTIMIZATION INTERFACE

D.1 Introduction

The entire interface is menu driven. All aspects of the design problem can be executed from the top level command file DSMOPT. When either CONVEN or CDESIGN are invoked subroutine ANAPRE will be executed. The following directions will explain how to execute and use the Design Simulation and Model Optimization Interface in a DEC VAX/VMS environment.

D.2 Executing the DSMOPT Command File

At the command prompt enter

> **@dsmopt** (ret

and the following menu appears:

```
**************************************************
*                   MAIN MENU                    *
**************************************************
*  Please make the following selection           *
*        Enter  E  = execute ENPORT              *
*               D  = create/edit DATA FILES      *
*               O  = execute OPTDES              *
*               C  = execute CHECKFILE           *
*               X  = exit the session            *
**************************************************
```

At this point the user will begin the menu selection process. To leave DMSOPT enter X. The preceding options will be discussed separately.

D.2.1 Invoking ENPORT

```
************************************************
*                  MAIN MENU                   *
************************************************
*  Please make the following selection         *
*        Enter  E  = execute ENPORT            *
*               D  = create/edit DATA FILES    *
*               O  = execute OPTDES            *
*               C  = execute CHECKFILE         *
*               X  = exit the session          *
************************************************
```

From the main menu select option **E**

Following is a transcript of the ENPORT session.

6.1.1.2 ENPORT State Equations File

```
*****************************************
*****************************************
**                                     **
**              ENPORT-7               **
**                                     **
**      BOND GRAPH/BLOCK DIAGRAM       **
**            PROCESSOR                **
**               for                   **
**         NONLINEAR SYSTEMS           **
**                                     **
**            Version 3.3              **
**                                     **
**            Developed by             **
**         Rosencode Associates        **
**          Lansing,  Michigan         **
**                                     **
*****************************************
*****************************************
   (C)1989. Rosencode Associates Inc.
          All rights reserved.

   Date: 01/20/92      Time: 14:30:09
```

*Comment. The trapping mode has been turned off/on at this point.

```
Utilities options
--------------------
Menu, Command, Trap, Debug,  Return? (full): R


ENPORT-7 options
------------------------------------------------------
I: Initialize for a new model or load an existing model
T: Title input or modification
G: Graph description or modification
E: Equation description for nodes
S: Solve for the time response of the system
D: Display the results
F: File the model for later reference
R: Return to the operating system from ENPORT
-- - - - - - - - - - - - - - - - - - - - - - - - - - -
M: Manage the file directory
U: Utilities- mode_set, trap, debug, terminal
O: Optimize the design
H: Help
------------------------------------------------------
Enter option (R): F
```

```
Filer options
---------------------------------------
E: write Enport model file
S: write System equations to file
X: write matrix-x eXec file
U: write matrix-x User_code_block file
A: write ACSL input file
D: write Adams DIFF function file
R: Return to main menu
H: Help
---------------------------------------
Enter option (R): U

*** Please select some output variables
    or you will get no output from Matrix-x.


Output list (Y) options
---------------------------------------
A: Add variables to current list
D: Delete variables from current list
E: set current list to Empty
S: Show current list
H: Help
R: Return
---------------------------------------
Enter option (R): A

Ready to add output variables.
  (Enter blank line to quit.)


Enter output variable: F.1M
Enter output variable: E.1M
Enter output variable: Q.2Q
Enter output variable: Q.3Q
Enter output variable: Q.4Q
Enter output variable: Q.5Q
Enter output variable:


Output list
-------------------
Add, Delete, Empty, Show, Help, Return? (full): S

There are   7 output variables (Y).

  Y(  1):  Q.1Q
  Y(  2):  F.1M
  Y(  3):  E.1M
  Y(  4):  Q.2Q
  Y(  5):  Q.3Q
  Y(  6):  Q.4Q
  Y(  7):  Q.5Q
```

```
Output list
 ------------------
 Add, Delete, Empty, Show, Help, Return? (full): R


 Please define the INPUT variable types for the Matrix-x file.

 Types are  "E": external input (to the block),
            "I": internal input (within block),
            "R": RP vector input.

 Enter type for 46S          : I
 Enter type for 2S           : I
 Enter type for 48S          : I
 Enter type for 49S          : I
 Enter type for 6S           : I
 Enter type for 51S          : I
 Enter type for 8S           : I
 Enter type for 9S           : I
 Enter type for 21S          : I
 Enter type for 22S          : I
 Enter type for 23S          : I
 Enter type for 29S          : I
 Enter type for 25S          : I
 Enter type for 28S          : I
 Enter type for 24S          : I
 Enter type for 54S          : I
 Enter type for 55S          : I
 Enter type for 57S          : I
 Enter type for 58S          : I
 Enter type for E.5W         : I
 Enter type for E.2W         : I
 Enter type for E.1W         : I
 Enter type for E.4W         : I


 Please define the Named-Parameter types for the Matrix-x file.

 Types are  "E": external (to block),
            "I": internal (within block),
            "R": RP vector.

 Enter type for W1           : E
 Enter type for W2           : E
 Enter type for PHI1         : E
 Enter type for PHI2         : E
 Enter type for PHI3         : E
 Enter type for PHI4         : E
 Enter type for T1           : E
 Enter type for T2           : E
 Enter type for T3           : E
 Enter type for T4           : E
 Enter type for T5           : E
 Enter type for AMP1         : E
 Enter type for AMP2         : E
 Enter type for AMP3         : E
 Enter type for PHI5         : E
```

```
Enter type for PHI6          : E
Enter type for C1            : RP
Enter type for C2            : RP
Enter type for C3            : RP
Enter type for C4            : I
Enter type for C5            : I
Enter type for R1            : RP
Enter type for R2            : RP
Enter type for R3            : RP
Enter type for R4            : I
Enter type for R5            : I
Enter type for I1            : E
Enter type for I2            : E
Enter type for I3            : E
Enter type for I4            : E
Enter type for I5            : E
Enter type for SE1           : E
Enter type for SE2           : E
Enter type for SE3           : E
Enter type for SE4           : E
Enter type for SE5           : E
Enter type for TF1           : E
Enter type for TF2           : E
Enter type for TF3           : E


Please enter a model file name for Matrix-x (MODEL0): EXMPL1

The title to be written to the file is:
5 Dof Suspension System W/Variable Flow Source - WO/Grav. Effects

Is that okay? (Y):

User_Code_Block file written: EXMPL1.USR

MATSETUP.DAT    file written: EXMPL1.DAT

RP definition   file written: EXMPL1.RPD


Filer options
-------------
adams_Diff, Return, Help? (full): R
```

ENC

I:

T:

G:

E:

S:

D:

F:

R:

M:

D:

O:

H

E:

Do

```
ENPORT-7 options
 ---------------------------------------------------------
 I: Initialize for a new model or load an existing model
 T: Title input or modification
 G: Graph description or modification
 E: Equation description for nodes
 S: Solve for the time response of the system
 D: Display the results
 F: File the model for later reference
 R: Return to the operating system from ENPORT
 -- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 M: Manage the file directory
 U: Utilities- mode_set, trap, debug, terminal
 O: Optimize the design
 H: Help
 ---------------------------------------------------------
 Enter option (R): R


 Do you want to leave ENPORT? (N): Y
```

D.2.2 Creating/Editing Data Files

The main menu appears

```
************************************************
*                  MAIN MENU                   *
************************************************
*  Please make the following selection         *
*       Enter  E  = execute ENPORT             *
*              D  = create/edit DATA FILES     *
*              O  = execute OPTDES             *
*              C  = execute CHECKFILE          *
*              X  = exit the session           *
************************************************
```

From the main menu select D.

The next menu appears

```
************************************************
*  How many Data files are you creating        *
*                                              *
*    Note:   Each set of system driving        *
*            conditions must be described      *
*            by a separate data file.          *
************************************************
```

Enter number

Now suppose the model is optimized analyzing its behavior for two different flow models. This will require two data files. Each data file has a set of members which describe one flow model. If beginning this problem then 2 would be an appropriate response. To edit an existing file then 1 is the correct response.

The next menu appears

```
************************************************
*  Create/Edit datafile No: X                  *
*                           ---                *
*  Please enter Datafile Name                  *
*       Example:  DRSYS1 (limit 6 char)        *
*                                              *
*    Note: Do not enter filetype .dat          *
************************************************
```

Enter filename

If you are creating a new file then enter a unique name. If you wish to edit an existing file then enter the name of that file.

Note: ï Do not enter filetype (.dat will be assigned)
ï Limit names to six (6) characters

The following verification menu appears

```
*******************************************
*  You Entered:     FFFFFF                *
*                   ------                *
*  Is this correct (Y or N)               *
*******************************************
```

When Y is selected an active editing mode is invoked. Enter the data and save the files. Use edit commands for the DEC/VAX operating system.

If creating U-Vector data files then enter data as a column vector with 1 entry per line.

If you are creating an experimental data file enter the data as an array. The first column must be time (T) and the following columns represent a time history for each input variable. If an input is zero for a particular analysis then zero's must be entered. This process will be repeated based on the entry for the number of data files being created.

D.2.3 Executing OPTDES

The following menu appears

```
**************************************************
*                  MAIN MENU                     *
**************************************************
*  Please make the following selection           *
*        Enter  E  = execute ENPORT              *
*               D  = create/edit DATA FILES      *
*               O  = execute OPTDES              *
*               C  = execute CHECKFILE           *
*               X  = exit the session            *
**************************************************
```

From the main menu select option **O**.

The following menu appears

```
**************************************************
*  Please enter model name from ENPORT           *
*                                                *
*       Example:  MODEL0                         *
*                                                *
*       Note: Do not include filetype            *
**************************************************
```

Enter the system state equations filename created in ENPORT.

Note: Do not enter filetype (handled internally)

The following verification menu appears

```
**************************************************
*  You entered:                                  *
*                                                *
*                      FFFFFF                    *
*                      ------                    *
*  Is this correct (y or n)                      *
**************************************************
```

Upon continuation the file management process occurs. The following menu appears.

```
**************************************************
*              OPTIMIZATION MENU                 *
**************************************************
*  Select one of the following options           *
*            C  = compile all programs           *
*            I  = compile individual programs    *
*            L  = link to optimization files     *
*            S  = create setup executable        *
*           RC  = run CONVEN.EXE                 *
*           RS  = run SETUP.EXE                  *
*           RD  = run CDESIGN.EXE                *
*            M  = return to main menu            *
*            X  = exit the session               *
**************************************************
```

At this point any user written subroutines must be compiled. Two choices are available to do this C and I. If C is entered all subroutines are compiled and this menu will re-appear. If I is entered then the following menu appears:

```
**************************************************
* The number of programs to be compiled          *
**************************************************
```

Then the following menu appears

```
**************************************************
*              Enter the program name            *
**************************************************
```

This prompt will be repeated based on your previous response.

At this point all files are compiled and the optimization menu re-appears:

```
************************************************
*              OPTIMIZATION MENU               *
************************************************
*  Select one of the following options         *
*            C  = compile all programs         *
*            I  = compile individual programs  *
*            L  = link to optimization files   *
*            S  = create setup executable      *
*            RC = run CONVEN.EXE               *
*            RS = run SETUP.EXE                *
*            RD = run CDESIGN.EXE              *
*             M = return to main menu          *
*             X = exit the session             *
************************************************
```

Option L

At this point option L must be entered even if no programs were compiled. This is necessary to link the model file to the optimization software. This is only necessary once if the model file has not changed. Two messages will appear to verify that the CONVEN and CDESIGN executable files have been created. The menu will re-appear.

Option S

This option is only necessary once to create the SETUP executable file.

Option RC

This option executes CONVEN. This will create the initial design point.

Option RS

This option executes SETUP. This will define the design space.

Option RD

This option executes CDESIGN. This will perform the optimization process.

Option M

Returns the user to the main menu.

Option X

Exits the user from DMSOPT.

CONVEN, SETUP and CDESIGN all execute OPTDES software. When CONVEN and CDESIGN are selected the ANAPRE subroutine will be invoked. This is a pre-processor used to set up the optimization problem (see Executing ANAPRE). After each program has ran the optimization menu will re-appear.

D.2.4 Executing the Checkfile

From the main menu select option C

```
************************************************
*                  MAIN MENU                   *
************************************************
*  Please make the following selection         *
*        Enter  E  = execute ENPORT            *
*               D  = create/edit DATA FILES     *
*               O  = execute OPTDES             *
*               C  = execute CHECKFILE          *
*               X  = exit the session           *
************************************************
```

Checkfile executes ANAMAIN which assists in constructing the user-written subroutines. ANAMAIN is a menu driven program which will determines a set of performance measurements for a given set of analysis variables. Once the subroutines are returning the correct values for a given input then the optimization software can be executed. ANAMAIN is much like executing CONVEN except results are obtained quicker. The user will prompted for the no. of AV's and their values. The subroutine ANAFUN will be executed and a set of AF values will be returned. No changes in executing ANAPRE are required.

D.3 Executing ANAPRE

ANAPRE is a pre-processing subroutine used to collect information necessary for executing the interface software. This program will be called when CONVEN and CDESIGN are executed. An overview of how ANAPRE is executed is as follows.

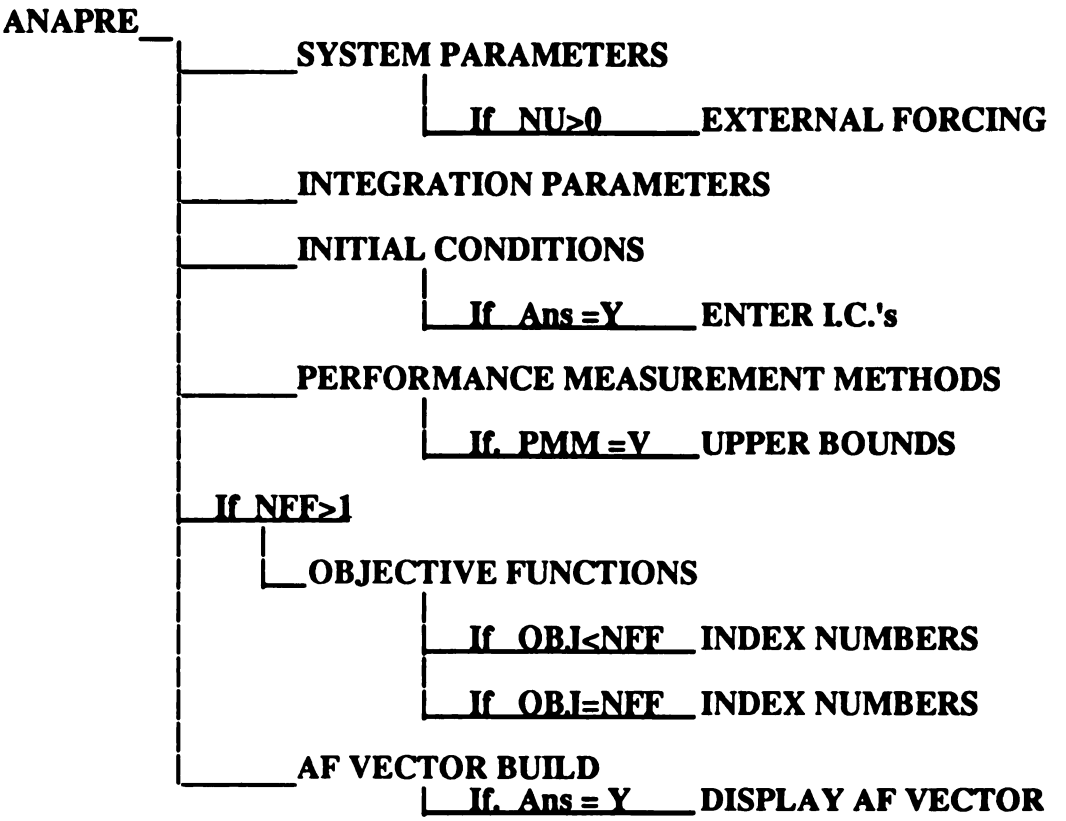


Figure D.3.1
Branching Tree for ANAPRE Execution

The following instructions are for entering data using ANAPRE. These prompts will be seen following OPTDES messages when either CONVEN or CDESIGN have been executed.

D.3.1 System Parameters

The following menu will appear:

Enter System Parameter Values		
Number of State Variables	(NX)	X
Number of Output Variables	(NY)	Y
Number of External Inputs	(NU)	U
Number of Forcing Conditions	(NFF)	F

Default values for X, Y, and U will be obtained from the summary xxx.rpd file. If this file is unavailable then the default values will be zero. F will have a default value of 1. Values for X,Y and U can be obtained from the system state equations file if they are unknown (See ENPORT user's manual appendix C).

D.3.2 External Forcing

If NU is greater than 0 then the model is being driven by a set of external inputs. The following sub-menu will appear:

Enter External Parameters
How is the forcing Data Being Modeled
E = Enport Model Generator
U = User Written Generator
D = Experimental Data

The following menus will appear for each response.

Enter **E**

Enter Datafile Information
Name of Datafile No (x)

Enter **U**

Do you wish to use your own menu (Y or N)

If **Y** then subroutine USRMEN is invoked
If **N** then

Enter Datafile Information
Name of Datafile No (x)

Enter **D**

Enter Datafile Information
Name of Datafile No (x)
Number of Data Pts for File No: (x)

In each case the user is entering the name of the datafile which is housing the U-Vector values. These prompts will be repeated based on the entry for NFF. When using experimental data these files must be edited to accommodate when (T_0 .ne. 0). If option D was entered the program will continue. If options E or U were entered the following menu will appear:

Please select one of the following

C =	**Continue**
V =	**View the data entered**
R =	**Re-enter the data files**

C will continue the program

V will display the data entered as it is stored in the **DS** matrix. The above menu will re-appear.

R will re-initialize **DS** and all data files must be re-entered. The above menu will re-appear.

D.3.3 Integration Parameters

The following prompt will appear:

Enter Integration Parameter Values
Initial Time Value (T_0)
Ending Evaluation Time (FT)
Number of Integration Steps

T_0 can be greater 0

FT must be greater than T_0

The number of integration steps must be less than 1024

After this information is entered the following message will appear:

The Integration Step Size (H) is: X

D.3.4 Initial Conditions

The following menu will appear:

Enter Initial Conditions (Y or N)
N

The default is N and will appear. To continue press enter. To enter initial conditions enter Y. When initial conditions are entered the following prompt will appear.

Initial Conds for State Variable i:

Where i is a counter displaying index number for the corresponding state variable. It is important to know how ENPORT has ordered the state variables to enter the IC's correctly. The order can be viewed in ENPORT where IC's are set or the state equation file can be edited to view how the X-Vector is ordered. This prompt will be repeated until i=NX.

D.3.5 Performance Measurement Methods

The following menu will appear:

Enter the Performance Measurement Methods to be used

M = Maximum Displacement
A = Ave Displacement
R = RMS Value
S = Minimum Displacement
X = Absolute Max Disp
N = Absolute Min Disp

Enter the appropriate letters (no spaces or delimiters required). Selections will be verified.

For each **PMM** selected a value for each **PM** will be calculated. Hence, more constraints could be created than will be used. Therefore, understanding how the **AF**-Vector is built is important when selecting constraints in SETUP.

D.3.6 Objective Functions

If NFF is greater than 1 then menus gathering information about the objective function will appear. If NFF is equal to one then proceed to the **AF** build message.

The following prompt will appear if NFF > 1:

Enter Objective Function & Constraint Information
Objective Function Set Size i

The default value i is equal to NFF. If the set size entered is greater than 1 then additional prompts will appear. One of two cases will occur:

Case 1:
If set size i: $1 < i < NFF$

Enter Index Numbers for the Following
AV Comparator Value
AF Objective Function Set Member

Case 2:
If set size i: $1 < i = NFF$

Enter Index Numbers for the Following
AV Comparator Value

In both cases the user is asked for which analysis variable (AV) is being used as Λ.

In case 1 the user must determine which analysis functions will be used in the objective function set. This prompt will be repeated based on the number entered for the set size.

In case 2 the user must determine which performance measurement is being used as the objective function. The index number entered corresponds to the Y-Vector index number.

D.3.7 AF Vector Build

The following prompt will appear:

Do You Wish to View the AF-Vector Format (Y or N)
No of members X **N**

The size of the AF-vector is calculated and displayed in the prompt statement (X). The default reply is N and is displayed. To continue press enter. To view how the AF-vector will be built in PERFORM enter Y.

APPENDIX E

MANAGING THE INTERFACE SYSTEM

APPENDIX E

MANAGING THE INTERFACE SYSTEM

E.1 Interface Directory

This directory should be set up to contain the following files:

ANASUB.FOR	**- Used for OPTDES**
ANAMAIN.FOR	**- Used for checking files**
USRGEN.FOR	**- Dummy file**
USRMEN.FOR	**- Dummy file**
USRAFN.FOR	**- Dummy file**
SYSPRM.EDT	**- Creates Screen display of System Parms**
SYSDAT.EDT	**- Passes System Parms from ENPORT to OPTDES**
CDES2/OPTIONS	**- Used to create CDESIGN.EXE**

The three Dummy files are necessary for creating the executables if user written subroutines are not used.

The EDT files are edit command files executed form DSMOPT. They are used to manage the model library and to pass information from ENPORT to OPTDES. These files reduce user input.

The options file contains the OPTDES subroutines used for creating the executable file CDESIGN.

E.2 OPTDES Directory

This directory is used for storing the compiled optimization programs. Managing these files is discussed in the OPTDES.BYU users manual.

At this time DSMOPT uses object files SCRHP and PENHP when creating CDESIGN.EXE. These files declare the type of CRT and plotter being used. Versions also exist for Textronix and SUN. DSMOPT should be modified to incorporate the CRT and plotter flexibility available.

E.3 Command File DSMOPT Maintenance

When the directories for the Interface and optimization files have been created it is necessary to set two symbols used in DSMOPT.COM. The symbols are:

OPTFI - Path name to the optimization software
USEFI - Path name to the Interface software

DSMOPT.COM can be edited using the DEC/VAX EDIT command. These symbols are at the beginning of the executable code.

APPENDIX F

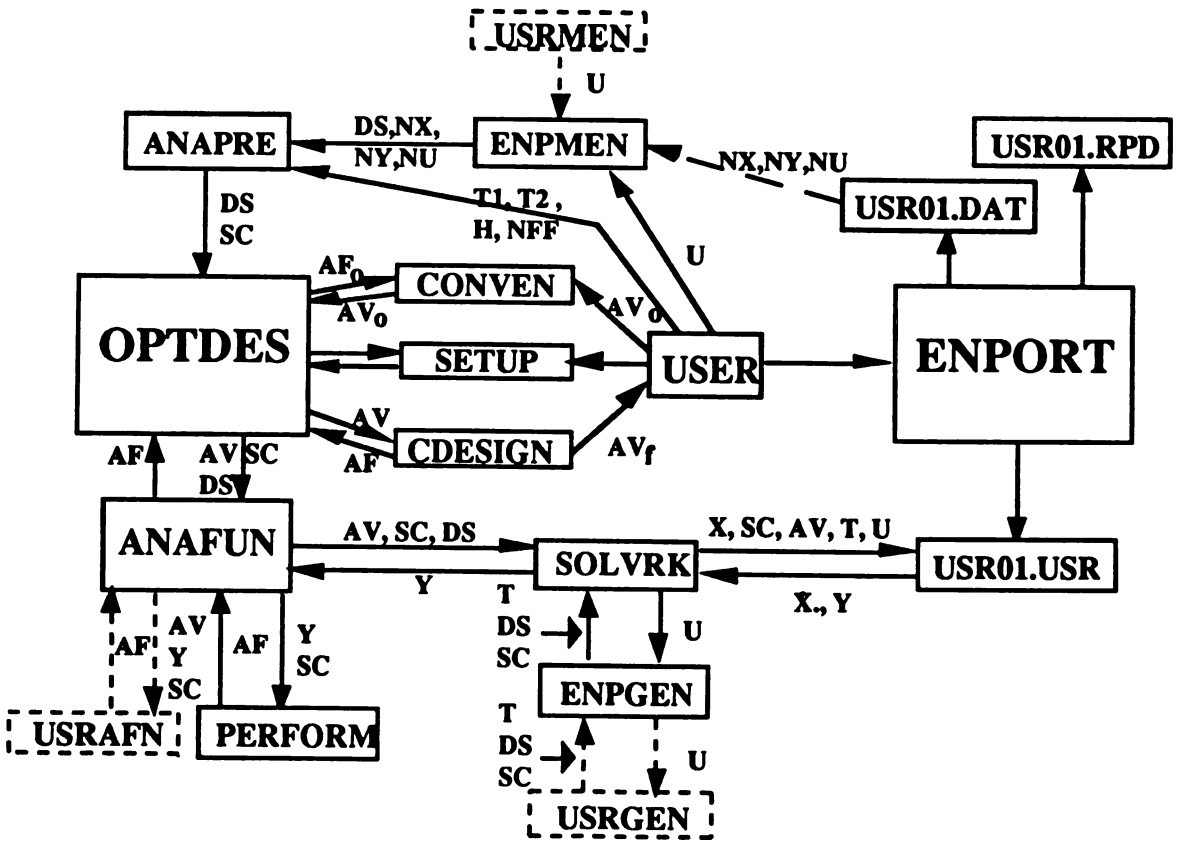
INTERFACE COMPUTER PROGRAMS

APPENDIX F

INTERFACE COMPUTER PROGRAMS

F.1 Introduction

The following figure illustrates the flow of data between ENPORT, OPTDES and the INTERFACE programs.



SC = SYSTEM CONSTANTS
{NX,NY,NU,NFF,T1,T2,H}

- - - - = OPTIONAL

— — — = DATA FILE TRANSFER

Figure F.1.1
Dynamic Modeling and Optimization Data Flow

F.2 ANASUB.FOR

This is the primary subroutine which is called by the main program in OPTDES. ANASUB will call SOLVRK and PERFORM which in turn call subsequent subroutines. ANASUB is responsible for receiving the AV vector from OPTDES and returning the AF vector.

COMPUTER PROGRAM

ANASUB.FOR

WRITTEN
BY

MARK DAVIS

```
C*****************************************************************
C.----SUBROUTINE ANAFUN        WRITTEN BY: MARK DAVIS  9/22/91
C.
C.----PURPOSE: THIS SUBROUTINE WILL PRODUCE TIME HISTORY DATA
C.             USING THE ENPORT CREATED MODEL.  THIS INFORMATION
C.             CAN BE USED TO OPTIMIZE SYSTEM RESPONSES OF A
C.             MULTI-DEG OF FREEDOM DYNAMIC SYSTEM DESIGN.
C.
C...GIVEN ARGUMENTS
C     AV()               = ANALYSIS VARIABLES
C.    AF()               = ANALYSIS FUNCTIONS

C...RETURNED ARGUMENTS
C.
C*****************************************************************
C.
      SUBROUTINE  ANAFUN (AV,AF)
C...ARGUMENTS
C.
        INTEGER DBUG, NYY, NYCNT, NFF, NPM
C.
        PARAMETER (NYY=20, NYCNT=4096)
C.
        DOUBLE PRECISION AV(1), AF(1), TV(NYCNT), yout(NYY,NYCNT)
C.
        INTEGER NOBJ,OBFCN,ICSTR,IAVN
        COMMON /INTVAR/ T1,T2,H,XIN(20),NX,NY,NS,NU,NFF,IEND,UAF
        COMMON /RESOUT/ TV,YOUT
        COMMON /TRACER/ IDBGR(20)
        COMMON /OBJFCN/ NOBJ,OBFCN(10),ICSTR,IAVN
C.
C. SET DEBUG TRACER 1=YES, 2=NO
C.
        DBUG = 2
C.
C. NFFCN set the number of different forcing functions for
C. analysis
        DO 10 NFFCN=1,NFF
C.
          CALL SOLVRK(NFFCN,AV)
C.
C. Dbugging aid
                          IF(DBUG.EQ.2)GOTO 1
                          CALL YRESLT(NYY)
1                         CONTINUE
C.
          CALL PERFORM(NFFCN,AV,AF,NPM)
C.
10      CONTINUE
C.
C. Set the AF vector values determined by the objective function
C. set.
        IF (NOBJ .GT. 1) CALL ENPAFN(NPM,AV,AF)
        IF (UAF .EQ. 'y' .or. UAF .EQ. 'Y')
       + Call usrafn(nffcn,npm,av,af)
C. Optional print operation
C.
```

```
                    if(idbgr(1) .eq. 1) then
                    do 12 i=1,ny*npm*nff
 12                   write(*,*)'this is af out',af(i)
                    else
                      continue
                    endif
C -
            return
            end
C ** ***********************************************************
C
            SUBROUTINE SOLVRK(NFFCN,AV)
C
            INTEGER ITIME, NU, INFO(4), IP(1), DBUG, NYY, NYCNT
C
            PARAMETER (NYY=20, NYCNT=4096)
C
            DOUBLE PRECISION AV(1)
C
            DOUBLE PRECISION  T,X(NYY,NYCNT),XDOT(NYY,NYCNT),U(40),
           +                  XT(NYY), VT(NYY),XI(NYY),X0(NYY),
           +                  XD(NYY),YRR(NYY),DR
C -
            COMMON /INTVAR/ T1,T2,H,XIN(20),NX,NY,NS,NU,NFF,IEND,UAF
            COMMON /RESOUT/ TV(NYCNT),YOUT(NYY,NYCNT)
            COMMON /TRACER/ IDBGR(20)
C
              nxt = 0
              dr  = 0.D0
C -
              do 5 icond=1,nx
                      xi(icond) = xin(icond)
  5           continue
C -
              info(1) = 0
              info(2) = 0
              info(3) = 1
              info(4) = 1
              IST     = (T1/H) + 1
C -
C
C  THIS IS USED FOR DBUG PURPOSES
C
C             WRITE(*,*)'MASS=',AV(1),' K4=',AV(2),' DAMP3=',AV(3)
C             WRITE(*,*)'ENTER 1 TO DISPLAY '
C             READ (*,*) IDIS
              IDIS = 2
              IEXT = 2
C -
C - - BEGIN RUNGE-KUTTA INTEGRATION
C -
              do 60 Intg=1,NS
C -
```

```
            IF(INTG .LT. IST)GOTO 7
            ICNT = INTG - ((T1)/H)
            TV(ICNT) = DR
7           T0       = DR
            T        = T0
C.
                 do 10 jl1=1,nx
                    x0(jl1) = xi(jl1)
                    xt(jl1) = xi(jl1)
10               continue
C.
                         IF(IDIS.EQ.1)
     +                   write(*,*)'1st disp',xt(1)
C.
            IF(NU .gt. 0)
     +      CALL ENPGEN(nffcn,t,u)
            CALL zzzzzz(INFO,T,U,NU,XT,VT,NX,YRR,NY,AV,IP)
c.
c. Save the specified output variables (Y-vector)
C.
               IF (INTG .LT. IST) GOTO 19
                 DO 18 IY=1,NY
                    YOUT(IY,ICNT) = YRR(IY)
                         if(idbgr(2) .eq. 1)
     +                     write(*,*)'this is yrr out',yrr(iy)
18               CONTINUE
C.
19               T = T0 + h*0.5
                 DO 20 JL2=1,NX
                    xt(jl2) = x0(jl2) + vt(jl2)*h*0.5
                    xd(jl2) = vt(jl2)
20               continue
C.
                         ww1 = vt(1)*h*.5
                         IF(IDIS.EQ.1)
     +                   write(*,*)'2nd disp',xt(1)
                         IF(Iext.eq.1)
     +           write(*,*)'u(1)=',u(1),'T=',T,'yout=',yout(1,icnt)
C.
            IF(NU .gt. 0)
     +      CALL ENPGEN(nffcn,t,u)
            CALL zzzzzz(INFO,T,U,NU,XT,VT,NX,YRR,NY,AV,IP)
C.
                 T = T0 + H*0.5
                 do 30 jl3=1,nx
                    xt(jl3) = x0(jl3) + vt(jl3)*h*0.5
                    xd(jl3) = xd(jl3) + 2.*vt(jl3)
30              continue
C.
                         ww2 = vt(1)*h*.5
                         IF(IDIS.EQ.1)
     +                   write(*,*)'3rd disp',xt(1)
                         IF(Iext.eq.1)
     +                   write(*,*)'  U(1)=',U(1),'    T=',T
C.
            IF(NU .gt. 0)
     +      CALL ENPGEN(nffcn,t,u)
            CALL zzzzzz(INFO,T,U,NU,XT,VT,NX,YRR,NY,AV,IP)
```

```
C.
                T = T0 + H
                do 40 jl4=1,nx
                   xt(jl4) = x0(jl4) + vt(jl4)*h
                   xd(jl4) = xd(jl4) + 2.*vt(jl4)
40              continue
C.
                        ww3 = vt(1)*h
                        IF(IDIS.EQ.1)
     +                  write(*,*)'4th disp',xt(1)
                        IF(Iext.eq.1)
     +                  write(*,*) '   U(1)=',U(1),'   T=',T
C.
           IF(NU .gt. 0)
     +     CALL ENPGEN(nffcn,t,u)
           CALL zzzzzz(INFO,T,U,NU,XT,VT,NX,YRR,NY,AV,IP)
C.
                        IF(IDIS.EQ.1)
     +                  write(*,*)'init disp',x0(1)
                        IF(Iext.eq.1)
     +                  write(*,*)'   U(1)=',U(1),'    T=',T
c.
                do 50 jl5=1,nx
                    xi(jl5) =  x0(jl5) + (vt(jl5) + xd(jl5))*h/6.
50              continue
c.
                        IF(IDIS.EQ.1)
     +                  write(*,*)'new init disp',xi(1)
                        IF(Iext.eq.1)
     +                  write(*,*)'  end 1 itteration '
c                       write(*,*)'******enter any num******'
c                       read(*,*) icont
C.
        dr         = dr + h
C.
60      CONTINUE
C
        DBUG = 2
        IF(DBUG.EQ.2)GOTO 1
           CALL YRESLT
 1      continue
C
        return
        end
C
C*****************************************************************
       SUBROUTINE ANAPRE
C.
C...  This Subroutine will be used for entering integration
C     conditions.
C ---- PURPOSE:   DEFINE VARIABLES USED
C                 FOR INTEGRATION
C
C ---- VARIABLES: T1  = INITIAL TIME
C                 T2  = FINAL TIME
C                 H   = INTEGRATION STEP SIZE
C                 XIN = STATE VARIABLE ITIAL CONDITION
```

```
C                NX  = NUMBER OF STATE VARIABLES
C                NY  = NUMBER OF OUTPUT VARIABLES
C                NS  = NUMBER OF INTEGRATION STEPS
C                NU  = NUMBER OF EXTERNAL SOURCES
C                NFF = NUMBER OF FORCING FUNCTIONS
C                OBJ = OBJECTIVE FUNCTION NUMBER
C                CMAX = UPPER BOUND LIMITS ON THE Y-VECTOR
C                       OUTPUT VARIABLES
C
      INTEGER NX, NY, NS, IEND, NYY, NFF, INF, NOBJ, ICSTR,
     +          IAVN, OBFCN, IDBGR, NSTEP, DOBJ
      INTEGER NNX,NNY,NNU,NMEM
      PARAMETER(NYY=20,NOB=10)
      DIMENSION XIN(nyy), CMAX(nyy)
      REAL T1, T2, H, XIN
      REAL CMAX
      CHARACTER IMAX*1, IPAR*7, ACOND*1, ISW*1, UAF*1, AFBLD*1
      COMMON /INTVAR/ T1,T2,H,XIN,NX,NY,NS,NU,NFF,IEND,UAF
      COMMON /TRACER/ IDBGR(20)
      COMMON /MAXVAL/ CMAX
      COMMON /PAROPT/ IPAR
      COMMON /OBJFCN/ NOBJ,OBFCN(10),ICSTR,IAVN
C.
C. Set default values
C.
      icmx = 2
      NX = 0
      NY = 0
      NU = 0
      NOBJ = 1
C.
C. Open datafile with nu,ny and nx values
C.
      open(unit=12,file='zzzzzz.dat',form='formatted',
     +     status='unknown')
      read (12,*,end=10)NNU,NNY,NNX
      close(unit=12)
C.
C. Prompt the user for the number of state variables
C.
10    Write(*,118)
      Write(*,100) NNX
      Read (*,200) NX
        if(NX .eq. 0)then
          NX = NNX
        else
          NX = NX
        endif
C.
C. Prompt the user for the number of output variables
C.
      Write(*,101) NNY
      Read (*,201) NY
        if(NY .eq. 0)then
          NY = NNY
        else
          NY = NY
        endif
```

```
C.
C. Prompt the user for the Number of external source parameters.
C.   If NU>0 then call enpmen.  Enpmen will prompt for the data
C.   files describing each set of external forcing systems (nff).
C.
         Write(*,109) NNU
         Read (*,201) nu
          if(NU .eq. 0)then
            NU = NNU
          else
            NU = NU
          endif
C.
C. Prompt the user for the number of forcing functions
C.
        inf = 1
        Write(*,112) INF
        Read (*,201) NFF
         if(nff .eq. 0)then
            nff = inf
         else
            nff = nff
         endif
C.
C. Call ENPMEN if NU>0
C.
           If (nu .gt. 0) call enpmen
C.
C. Prompt the user for the intial start time
C.
35      Write(*,119)
        Write(*,102)
        Read (*,202) t1
C.
C. Prompt the user for the end evaluation time
C.
        Write(*,103)
        Read (*,203) t2
C.
C. Prompt the user for the integration step size
C.
        Write(*,104)
        Read (*,200) nstep
C.
        H = (t2 - t1)/nstep
C.
        Write(*,121) H
C.
C. Calculate the number of integration steps to use
C.
        ZS   = T2/H
        NS   = ZS
        IEND = (T2 - T1)/H
C.
C. Prompt the user for initial conditions on the state variables
C.
```

```
        acond = 'N'
48      Write(*,114) acond
        Read (*,207) acond
        if(acond .eq. ' ')then
          acond = 'n'
          goto 60
        elseif(acond .eq. 'y' .or. acond .eq. 'Y')then
          goto 49
        elseif(acond .eq. 'n' .or. acond .eq. 'N')then
          goto 60
        else
          goto 48
        endif
C.
49      ivar = 0
        do 50 icond=1,nx
         ivar = icond
           Write(*,105) ivar
           Read (*,205) xin(icond)
 50     continue

C.
C. Select Performance Measurement Methods
C.
60      write (*,110)
        read  (*,208) Ipar
C.
        loc1 = index(ipar,'m')
        loc2 = index(ipar,'a')
        loc3 = index(ipar,'r')
        loc4 = index(ipar,'v')
        loc5 = index(ipar,'s')
        loc6 = index(ipar,'x')
        loc7 = index(ipar,'n')
C.
        NPM = 0
        IF (LOC1 .GT. NPM) NPM = LOC1
        IF (LOC2 .GT. NPM) NPM = LOC2
        IF (LOC3 .GT. NPM) NPM = LOC3
        IF (LOC4 .GT. NPM) NPM = LOC4
        IF (LOC5 .GT. NPM) NPM = LOC5
        IF (LOC6 .GT. NPM) NPM = LOC6
        IF (LOC7 .GT. NPM) NPM = LOC7
C.
        LOC8 = LOC1 + LOC2 + LOC3 + LOC4 + LOC5 + Loc6 + Loc7
C.
        IF (LOC8 .EQ. 0) THEN
          WRITE(*,111) ipar
          GOTO 60
        ELSE
          Continue
        ENDIF
C.
        If (Loc4 .gt. 0)then
          Do 62 I=1,ny
62          Cmax(i) = 0
C.
```

```
C. Enter the upper bounds necessary for using the maximum
C. violation calculation.
C.
          Write(*,107)
          Do 65 Inx=1,ny
            Write(*,108) Inx
65        Read (*,*) Cmax(inx)
        Else
          Continue
        Endif
C.
C. Call for objective function size only if multiple external
C. forcing systems or more than 1 perf meas method is being used.
C.
         If (nff .gt. 1) Goto 66
         If (nff .le. 1) Goto 67
C.
C. Prompt the user for the  objective function number
C.  Note: This is only necessary if the objective function
C.        is the max/min of a set of constraints.  This is
C.        typical for problems with multiple external driving
C.        systems.
66      Write(*,120)
        Write(*,113) nff*npm
        Read (*,201) nobj
          if(nobj .eq. 0)then
            nobj = nff
          else
            nobj = nobj
          endif
C.
C. Call subroutine enpobj to determine the objective function set
C.
        If (nobj .gt. 1) Call Enpobj(npm)
c.
c. View the Analysis function build
c.
67      afbld = 'N'
        If (Nobj .gt. 1)then
         nmem = nff*npm*ny + 1
        Else
         nmem = nff*npm*ny
        Endif
c.
        Write(*,122) nmem, afbld
        Read (*,207) afbld
        if(afbld .eq. ' ')then
         afbld = 'n'
        elseif(afbld .eq. 'y' .or. afbld .eq. 'Y')then
         call afbild(AV,AF,NMEM)
        elseif(afbld .eq. 'n' .or. afbld .eq. 'N')then
         continue
        else
         goto 67
        endif
c.
c. Prompt the user to see if they wish to re-edit the af vector
```

```
c.
68      Uaf = 'N'
c       Write(*,106) uaf
c       Read (*,207) UAF
c       if(UAF .eq. ' ')then
c        UAF = 'n'
c       elseif(UAF .eq. 'y' .or. UAF .eq. 'Y')then
c        continue
c       elseif(UAF .eq. 'n' .or. UAF .eq. 'N')then
c        continue
c       else
c        goto 68
c       endif
c.********************Print Switches are Off*********************
c. Set print switches
c.*****************The Default Setting is Left on***************
        do 70 i=1,20
70        idbgr(i) = 2
c.
c.
c       Isw='N'
c72     Write(*,117)isw
c       Read (*,207)isw
c       If (isw .eq. ' ')then
c         isw = 'n'
c         goto 78
c       Elseif(isw .eq. 'n' .or. isw .eq. 'N')then
c         goto 78
c       ElseIf (isw .eq. 'y')then
c       Write(*,115)
c       Read (*,207) ISW
c         Do 74 i=1,4
c           Write(*,116) i
c74         read(*,*) idbgr(i)
c        else
c           goto 72
c        endif
c.
c78     continue
c.********************End Print Switches************************
C.
 118    format(' ',///////,
     +         ' Enter System Driving Parameter Values',/,
     +         ' ------------------------------------------
     +-----------------')
 100    format(11x,'Number of State Variables    (NX)  ',I2,' ',$)
 101    format(11x,'Number of Output Variables   (NY)  ',I2,' ',$)
 112    format(11x,'Number of Forcing Conditions (NFF) ',I2,' ',$)
 109    format(11x,'Number of External Inputs    (NU)  ',I2,' ',$)
C.*************************************************************
 119    format('   ',/,
     +         ' Enter Integration Parameter Values',/,
     +         ' ------------------------------------------
     +-----------------')
 102    format(11x,'Initial Time Value     (T0)          ',$)
 103    format(11x,'Ending Evaluation Time (TF)          ',$)
 104    format(11x,'Number of Intergration Steps         ',$)
 121   format('',//,5x,'The Integration Step Size (H) is:',e12.6)
```

```
C.************************************************************
 106    format('  ',/,
     +         ' Are You Using Subroutine USRAFN (Y or N)',/,
     +         ' ------------------------------------------
     +----------',/,
     +         '                                          ',a2,' ',$)
C.************************************************************
 110    format('  ',/,
     +' Enter the Performance Measurement Methods to be used',/,
     +' ------------------------------------------------------
     +----------',/,
     +         11x,'  M = Maximum Displacement    ',/,
     +         11x,'  A = Ave Displacement       ',/,
     +         11x,'  R = RMS Value              ',/,
     +         11x,'  V = Max Violation Integral ',/,
     +         11x,'  S = Minimum Displacement   ',/,
     +         11x,'  X = Absolute Max Disp      ',/,
     +         11x,'  N = Absolute Min Disp      ',/,
     +         11x,'  ',$)
 111  format(11x,'You Entered ',a7,' None of',/,
     +         11x,'These are Valid Try Again')
 107  format(11x,'You must enter the upper bound limits to use',/,
     +       11x,'  the maximum violation performance option.',/,
     +       11x,'-------------------------------------------')
 108    format(13x,'Upper Bound for Perf. Meas. #:',i3,':    ',$)
C.************************************************************
 120    format('   ',/,
     +' Enter Objective Function & Constraint Information',/,
     +' ------------------------------------------------
     +----------')
 113  format(11x,'Objective Function Set Size        ',i2,' ',$)
C.************************************************************
 114    format('   ',/,
     +         ' Enter Initial Conditions (Y or N) ',/,
     +       ' ------------------------------------------
     +----------',/,
     +         '                                          ',a2,' ',$)
 105    format(11x,'Initial Conds for State Variable',i3,':   ',$)
C.************************************************************
 115    format(11x,' 1st switch prints AF vector',/,
     +         11x,' 2nd switch prints YRR history in solvrk',/,
     +         11x,' 3rd switch prints perform call',/,
     +         11x,' 4th switch prints yperf build of af')
 116    format(13x,' Enter setting for switch number',i3,/,
     +         13x,'     1 = on   2 = off              ',$)
 117    format('   ',/,
     +       ' Set print switches on to view builds (Y or N) ',/,
     +         ' ------------------------------------------
     +----------',/,
     +         '                                          ',a2,' ',$)
 122    format('   ',/,
     +   ' Do You Wish to View the AF-Vector Format (Y or N) ',/,
     +         ' ------------------------------------------
     +----------',/,
     +         ' No of members:',i3,'                     ',a2,' ',$)
C.
```

```
 200    format (i4)
 201    format (i4)
 202    format (e6.0)
 203    format (e6.0)
 204    format (e4.2)
 205    format (e8.2)
 207    format (a1)
 208    format (a7)

        return
        end
C.
C. ****************************************************************
C.
        SUBROUTINE ANAPOS
        INTEGER NYY, NYCNT
        PARAMETER (NYY=20, NYCNT=4096)
        COMMON /RESOUT/ TV(NYCNT),yout(NYY,NYCNT)
        COMMON /INTVAR/ T1,T2,H,XIN(20),NX,NY,NS,NU,NFF,IEND,UAF
C



C.
        RETURN
        END
C.
C. ****************************************************************
C.
        SUBROUTINE ANAGRA
        COMMON /INTVAR/ T1,T2,H,XIN(20),NX,NY,NS,NU,NFF,IEND,UAF
C.
        RETURN
        END
C.
C. ****************************************************************
C. ****************************************************************
C.
C.      OPTIMIZATION SUBROUTINES
C.
C. ****************************************************************
C. ****************************************************************
C
C  ---- SUBROUTINE PERFORM
C
C  ---- PURPOSE:   DEFINE THE DYNAMIC PERFORMANCE OF THE SYSTEM
C
C  ---- OUTPUT
C       VARIABLE:  YPERF(I,J) = DATA ARRAY OF EACH DESIGN VARIABLE
C                               PERFORMANCE
C
C                          I = PERFORMANCE VARIABLE
C                          J = DESIGN VARIABLE
C
```

```
C ---- SYSTEM
C      PERFORMANCE
C      VARIABLES  YMAX(I)  = SYSTEM RESPONSE MAXIMUM DEFLECTION
C
C                 YMIN(I)  = SYSTEM RESPONSE MINIMUM DEFLECTION
C
C                 YABS(I)  = ABSOLUTE MAXIMUM DEFLECTION
C
C                 YINT(I)  = INTEGRAL FOR MAXIMUM DEFLECTION
C
C                 TMAX(I)  = TIME AT WHICH MAX VALUE OCCURS
C
C                 TMIN(I)  = TIME AT WHICH THE MIN VALUE OCCURS
C
C                 TABS(I)  = TIME AT WHICH ABS MAX VALUE OCCURS
C
C                 YMEAN(I) = SYSTEM RESPONSE MEAN VALUES
C
C                 YRMS(I)  = SYSTEM RESPONSE ROOT MEAN SQUARE VALUES
C
C                 YOVR(I)  = PERCENT OVERSHOOT OF YMAX VS. YMEAN
C
      SUBROUTINE PERFORM(NFFCN,AV,AF,NPM)
C
      CHARACTER Ipar*7
C
      INTEGER DBUG, NYPERF, NYY, NYCNT, NFFCN, NPM
C
      PARAMETER (NYPERF=5, NYY=20, NYCNT=4096)
C
      DOUBLE PRECISION YPERF(NYY,NYPERF), AV(1),
     +                 AF(1), ZOUT(2,NYY,NYCNT)
C
      DOUBLE PRECISION YMAX(NYY),  TMAX(NYY), YMEAN(NYY),
     +                 YRMS(NYY),  YSST(NYY), YOVR(NYY),
     +                 YLIN(NYY),  YOFF(NYY), TOFF(NYY),
     +                 RTIM(NYY),  YERR(NYY), YEXP(NYY),
     +                 YMIN(NYY),  TMIN(NYY), TSMAX(NYY)
C
      DOUBLE PRECISION TSMIN(NYY), YPP(NYY,2), TPP(NYY,2),
     +                 YINT(NYY),  YABS(NYY),  TABS(NYY),
     +                 YNAB(NYY),  TNAB(NYY)
C
      COMMON /INTVAR/ T1,T2,H,XIN(20),NX,NY,NS,NU,NFF,IEND,UAF
      COMMON /RESOUT/ TV(NYCNT), YOUT(NYY,NYCNT)
      COMMON /OBJFCN/ NOBJ,OBFCN(10),ICSTR,IAVN
      COMMON /EXTOUT/ ZOUT
      COMMON /TRACER/ IDBGR(20)
      COMMON /PAROPT/ IPAR
C
      PERF = IDBGR(3)
      PRFO = IDBGR(4)
C
            IF(PERF .EQ. 1)PRINT *,'CALLING PMAX'
       CALL PMAX(YMAX,TMAX,TSMAX,YPP,TPP,YABS,TABS,YINT)
C
```

```
              IF(PERF .EQ. 1)PRINT *,'CALLING PMEAN'
         CALL PMEAN(YMEAN)
C
              IF(PERF .EQ. 1)PRINT *,'CALLING PRMS'
         CALL PRMS(YRMS)
C
              IF(PERF .EQ. 1)PRINT *,'CALLING POVR'
         CALL POVR(YMAX,YMEAN,YOVR)
C
              IF(PERF .EQ. 1)PRINT *,'CALLING PMIN'
         CALL PMIN(TMAX,YMIN,TMIN,TSMIN,YNAB,TNAB)
C
        loc1 = index(ipar,'m')
        loc2 = index(ipar,'a')
        loc3 = index(ipar,'r')
        loc4 = index(ipar,'v')
        loc5 = index(ipar,'s')
        loc6 = index(ipar,'x')
        loc7 = index(ipar,'n')
C.
        DO 25 IVAR = 1,NY
          IF (loc1 .ne. 0) goto 10
17        IF (Loc2 .ne. 0) goto 11
18        IF (loc3 .ne. 0) goto 12
19        IF (loc4 .ne. 0) goto 13
20        IF (loc5 .ne. 0) goto 14
21        IF (loc6 .ne. 0) goto 15
22        IF (loc7 .ne. 0) goto 16
        goto 25
C.
10        YPERF(IVAR,LOC1) =  Ymax(IVAR)
        GOTO 17
11        YPERF(IVAR,LOC2) = YMEAN(IVAR)
        GOTO 18
12        YPERF(IVAR,LOC3) =  YRMS(IVAR)
        GOTO 19
13        YPERF(IVAR,LOC4) =  Yint(IVAR)
        GOTO 20
14        YPERF(IVAR,LOC5) =  YMIN(IVAR)
        GOTO 21
15        YPERF(IVAR,LOC6) =  Yabs(IVAR)
        GOTO 22
16        YPERF(IVAR,LOC7) =  Ynab(IVAR)
C
25     CONTINUE
C
C  BUILD THE AF VECTOR
C
        NPM = 0
        IF (LOC1 .GT. NPM) NPM = LOC1
        IF (LOC2 .GT. NPM) NPM = LOC2
        IF (LOC3 .GT. NPM) NPM = LOC3
        IF (LOC4 .GT. NPM) NPM = LOC4
        IF (LOC5 .GT. NPM) NPM = LOC5
        IF (LOC6 .GT. NPM) NPM = LOC6
        IF (LOC7 .GT. NPM) NPM = LOC7
```

```
C
        Ibeg = (nffcn-1)*npm + 1
        Ifin = nffcn*npm
        IPY  = 0
C
        DO 40 IY = Ibeg,Ifin
         ipy = ipy + 1
          DO 40, IP=1,ny
            I = (IY - 1)*ny + IP
C
C Dbugging aid
                          If(Prfo .eq. 1)
     +                    write(*,*)'this is yperf',yperf(ip,ipy)
c
40      AF(I) = YPERF(Ip,Ipy)
C Dbugging aid
                          DBUG = 2
                          IF(DBUG.EQ.2)GOTO 1
                          CALL YPRFRM(YPERF,NYPERF,NYY)
1                         CONTINUE
C.
c.
c.
        RETURN
        END
C.
C. NOTES
C.   NEED TO DETERMINE confidence interval for THE CALCULATED
C.   RESULTS BASED ON ENOUGH DATA POINTS TAKEN.
C.*************************************************************
C.----SUBROUTINE PMAX
c.
C.----PURPOSE
C.      THIS SUBROUTINE WILL DETERMINE THE MAXIMUM VALUES
C.      FOR THE SYSTEM RESPONSE VARIABLES
C.
        SUBROUTINE PMAX(YMAX,TMAX,TSMAX,YPP,TPP,YABS,TABS,YINT)
C.
        INTEGER dbug2, NYY, NYCNT
        PARAMETER(NYY=20, NYCNT=4096)
C.
        DOUBLE PRECISION YMAX(1), TMAX(1), TSMAX(1), YPP(NYY,1),
     +                   TPP(NYY,1), YABS(1), TABS(1), YINT(1)
C.
        COMMON /INTVAR/ T1,T2,H,XIN(NYY),NX,NY,NS,NU,NFF,IEND,UAF
        COMMON /RESOUT/ TV(NYCNT), YOUT(NYY,NYCNT)
        COMMON /MAXVAL/ CMAX(NYY)
C.
        dbug2 = 2
C.
```

```
DO 10 IVAR=1,NY
         ZMAX = -10000
         ZTMX = 0
                DO 20 IMAX=1,IEND
                  IF(ZMAX.LT.YOUT(IVAR,IMAX))THEN
                    ZTMX = TV(IMAX)
                    ZMAX = YOUT(IVAR,IMAX)
                  ELSE
                    ZTMX = ZTMX
                    ZMAX = ZMAX
                  ENDIF
 20             CONTINUE
          TMAX(IVAR) = ZTMX
          YMAX(IVAR) = ZMAX
 10      CONTINUE
C.
         DO 40 IVAR=1,NY
          Zabs = -10000
          ZTab = 0
                DO 30 IMAX=1,IEND
                  IF(Zabs .LT. abs(YOUT(IVAR,IMAX)))THEN
                    ZTab = TV(IMAX)
                    Zabs = abs(YOUT(IVAR,IMAX))
                  ELSE
                    ZTab = ZTab
                    Zabs = Zabs
                  ENDIF
 30             CONTINUE
          Tabs(IVAR) = ZTab
          Yabs(IVAR) = Zabs
 40      CONTINUE
C.
         IMXV = 2
         ITIM = (t2/H) + 1
C.
         Do 45 Ivar=1,ny
          sumodd = 0.
          sumeve = 0.
           Do 50 Isum=2,Itim-1,2
             Cout   = abs(Yout(ivar,Isum))
             Sumodd = Sumodd +
     +              Cout - Cmax(Ivar) + abs(Cout - Cmax(Ivar))
         if(IMXV .eq. 1)
     +   print *,'summodd=',sumodd,'cout=',cout,'cmax=',cmax(ivar)

50       continue
C.
           Do 60 Isum=3,Itim-2,2
             Cout   = abs(Yout(Ivar,Isum))
             Sumeve = Sumeve +
     +              Cout - Cmax(ivar) + abs(Cout - Cmax(Ivar))
         if(IMXV .eq. 1)
     +   print *,'sumeve=',sumeve,'cout=',cout,'cmax=',cmax(ivar)
60       continue
C.
             C0 = abs(Yout(Ivar,1))
             Ct = abs(Yout(Ivar,Itim))
```

```
C.
            F0 = C0 - Cmax(Ivar) + abs(C0  - Cmax(Ivar))
C.
            FT = Ct - Cmax(Ivar) + abs(Ct  - Cmax(Ivar))
C.
         if(IMXV .eq. 1)print *,'F0=',f0,'    FT=',ft
45       Yint(IVAR) = (F0 + FT + 4*Sumodd + 2*Sumeve) * (H/3)
C.
          return
          end
C.
C.*****************************************************************
C.----SUBROUTINE PMIN
c.
C.----PURPOSE
C.       THIS SUBROUTINE WILL DETERMINE THE MINIMUM VALUES
C.       FOR THE SYSTEM RESPONSE VARIABLES
C.
         SUBROUTINE PMIN(TMAX,YMIN,TMIN,TSMIN,YNAB,TNAB)
C.
         DOUBLE PRECISION YMIN(1), TMIN(1), TMAX(1), TSMIN(1),
     +                    YNAB(1), TNAB(1)
C.
         INTEGER dbug2, NYY, NYCNT
         PARAMETER(NYY=20, NYCNT=4096)
C.
         COMMON /INTVAR/ T1,T2,H,XIN(20),NX,NY,NS,NU,NFF,IEND,UAF
         COMMON /RESOUT/ TV(NYCNT), YOUT(NYY,NYCNT)
C.
         dbug2 = 2
C.
         DO 10 IVAR=1,NY
           ZMIN = 10000
           ZTMN = 0
           IST  = (TMAX(IVAR)/H)+1
             DO 20 IMIN=1,IEND
                IF (ZMIN .GT. YOUT(IVAR,IMIN)) THEN
                  ZTMN = TV(IMIN)
                  ZMIN = YOUT(IVAR,IMIN)
                ELSE
                  ZTMN = ZTMN
                  ZMIN = ZMIN
                ENDIF
 20          CONTINUE
           TMIN(IVAR) = ZTMN
           YMIN(IVAR) = ZMIN
 10      CONTINUE
C.
C.
         DO 40 IVAR=1,NY
          Zabs = 10000
          ZTab = 0
```

```
                DO 30 IMIN=1,IEND
                  IF(Zabs .GT. abs(YOUT(IVAR,IMIN)))THEN
                    ZTab = TV(IMIN)
                    Zabs = abs(YOUT(IVAR,IMIN))
                  ELSE
                    ZTab = ZTab
                    Zabs = Zabs
                  ENDIF
 30             CONTINUE
          TNAB(IVAR) = ZTab
          YNAB(IVAR) = Zabs
 40     CONTINUE
C.
         return
         end
C.
C.*****************************************************************
C.----SUBROUTINE PMEAN
C.
C.----PURPOSE
C.       THIS SUBROUTINE WILL DETERMINE THE MEAN VALUE OF THE
C.       OUTPUT VARIALBLES SPECIFIED IN THE YOUT VECTOR.
C.
         SUBROUTINE PMEAN(YMEAN)
C.
         DOUBLE PRECISION YMEAN(1), YSUM
C.
         INTEGER NS0, NYY, NYCNT
         PARAMETER(NYY=20, NYCNT=4096)
         DOUBLE PRECISION YMEAN1(NYY)
C.
         COMMON /INTVAR/ T1,T2,H,XIN(20),NX,NY,NS,NU,NFF,IEND,UAF
         COMMON /RESOUT/ TV(NYCNT), YOUT(NYY,NYCNT)
C.
         imn = 2
C.
         ITIME = (T2)/(H)
         IBEG  = (T1/H)+1.
         ISTO  = IBEG + 2
         ISTE  = ISTO + 1.
         AVET  = (T2 - T1)
C.
         DO 40 INY=1,NY
C.
          SUMODD = 0.0
          SUMEVE = 0.0
C.
           DO 30 ISUM=ISTO,ITIME-1,2
 30          SUMODD = SUMODD + (YOUT(INY,ISUM))
C.
           DO 35 ISUM=ISTE,ITIME-2,2
 35          SUMEVE = SUMEVE + (YOUT(INY,ISUM))
C.
           F0 = (YOUT(INY,IBEG))
           FT = (YOUT(INY,ITIME))
```

```
C.
        IF(IMN .EQ.1)
     +  PRINT *,'THIS IS THE FINAL TIME',TV(ITIME),TV(ITIME-1)
40     YMEAN(INY) = (1/AVET)*(F0 + FT + 4*SUMODD + 2*SUMEVE)*(H/3)
C.
         return
         end
C.
C.*************************************************************
C.----SUBROUTINE PRMS
C.
C.----PURPOSE: TO DETERMINE THE ROOT MEAN SQUARE VALUE OF
C.              THE OUTPUT DATA.
C.
        SUBROUTINE PRMS(YRMS)
C.
        INTEGER NYY, NYCNT
        PARAMETER(NYY=20, NYCNT=4096)
        DOUBLE PRECISION YRMS1(NYY)
C.
        DOUBLE PRECISION YSST(1), YRMS(1), YNUM
C.
        COMMON /INTVAR/ T1,T2,H,XIN(20),NX,NY,NS,NU,NFF,IEND,UAF
        COMMON /RESOUT/ TV(NYCNT), YOUT(NYY,NYCNT)
C.
        IRMS = 3
C.
        ITIME = (T2/H)
        IBEG  = (T1/H)+1.
        ISTO  = IBEG + 2
        ISTE  = ISTO + 1
        AVET  = t2 - t1
C.
        DO 20 INY=1,NY
C.
         SUMODD = 0.0
         SUMEVE = 0.0
C.
          DO 10 ISUM=ISTO,ITIME-1,2
        IF(IRMS.EQ.1)WRITE(*,*)'SUMODD=',SUMODD
10          SUMODD = SUMODD + abs(YOUT(INY,ISUM))
C.
          DO 15 ISUM=ISTE,ITIME-2,2
        IF(IRMS.EQ.1)WRITE(*,*)'SUMEVE=',SUMEVE
15          SUMEVE = SUMEVE + abs(YOUT(INY,ISUM))
C.
          F0 = abs(YOUT(INY,IBEG))
          FT = abs(YOUT(INY,ITIME))
C.
      IF(IRMS.EQ.2)WRITE(*,*)'FO=',F0,' FT=',FT,' SUMODD=',SUMODD,
     +  '  SUMEVE=',SUMEVE,'  ITIME=',TV(ITIME)
20     YRMS(INY) =(1/AVET)*(F0 + FT + 4*SUMODD + 2*SUMEVE)*(H/3)
C.
        IF(IRMS.LE.2)WRITE(*,*)'YRMS1=',YRMS1(1),'  YRMS=',YRMS(1)
C.
        RETURN
        END
```

```
C.*****************************************************************
C.----SUBROUTINE POVR
C.
C.----PURPOSE: TO DETERMINE THE PERCENT OVERSHOOT
C.
      SUBROUTINE POVR(YMAX,YMEAN,YOVR)
C.
      DOUBLE PRECISION YMAX(1), YMEAN(1), YOVR(1)
C.
      INTEGER NYY, NYCNT
      PARAMETER(NYY=20, NYCNT=4096)
C.
      COMMON /INTVAR/ T1,T2,H,XIN(20),NX,NY,NS,NU,NFF,IEND,UAF
      COMMON /RESOUT/ TV(NYCNT), YOUT(NYY,NYCNT)
C.
      DO 10 IVAR = 1,NY
       IF(YMEAN(IVAR).ne.0)THEN
        YOVR(IVAR)=((YMAX(IVAR)-YMEAN(IVAR))/(YMEAN(IVAR)))*100.
       ELSE
        YOVR(IVAR) = 100.
       ENDIF
 10    CONTINUE
C.
      RETURN
      END
C.
C.*****************************************************************
C.*****************************************************************
C.
C.    DEBUGGING PROGRAMS
C.
C.*****************************************************************
C.*****************************************************************
c
      SUBROUTINE YRESLT
c
      INTEGER NYY, NYCNT
      PARAMETER(NYY=20, NYCNT=4096)
c
      COMMON/INTVAR/t1,t2,h,xin(20),NX,NY,NS,NU,NFF,IEND,UAF
      COMMON/RESOUT/TV(NYCNT), YOUT(NYY,NYCNT)
c
      write(*,*) '  TIME       position     THIS IS YRESLT '
      write(*,*)NX,NY,NS,NU,IEND
c
      do 120 idv=1,IEND
      write(*,*)TV(IDV),'  ',yout(1,idv),'  ',YOUT(2,IDV)
 120  continue
c
      return
      end
C
```

```
C ***************************************************************
C
        subroutine yprfrm(YPERF,NYPERF,NYY)
C
        DOUBLE PRECISION YPERF(NYPERF,1)
C
        COMMON /INTVAR/ T1,T2,H,XIN(20),NX,NY,NS,NU,NFF,IEND,UAF
C
        DO 40 IY = 1,NY
          DO 40 IP = 1,NYPERF
 40             write(*,*) YPERF(IP,IY)
C
         return
         end
C.
c
c***************************************************************
c
        subroutine maxchk(ayout,ymax)
        double precision ayout,ymax(1)
c
        write (*,*)'ayout =',ayout,'      ymax =',ymax(1)
c
        return
        end
c***************************************************************
c
        subroutine MINCHK(XMIN,YMIN)
        double precision XMIN,YMIN(1)
c
        write (*,*)'XMIN =',XMIN,'      YMIN =',YMIN(1)
c
        return
        end
C***************************************************************
c.----Subroutine ENPGEN: GENERATOR = -U- vector
c.
c.----Purpose: This subroutine is the standard interface which
c.              extracts the appropriate U-vector from DS when
c.              multiple forcing systems are used.

        Subroutine ENPGEN(nffcn,t,u)
c.
c. Define passed variables
c.
        Double Precision t,U(1),dtim,ctim,tratio,uratio
        Integer nffcn,icnt,inxt
        Character Icls*1
c.
        Common /intvar/ T1,T2,h,xin(20),nx,ny,ns,nu,NFF,IEND,UAF
        Common /drvsys/ DS(1500,10), ICLS
c.
c       write(*,'(a)')'icls in enpgen',icls
        If (Icls .eq. 'e' .or. Icls .eq. 'E') Goto 10
        If (Icls .eq. 'u' .or. Icls .eq. 'U') Goto 30
        If (Icls .eq. 'd' .or. Icls .eq. 'D') Goto 50
```

```
c.
10      Do 12 j=1,nu
12        U(j) = DS(j,nffcn)
        Goto 90
c.
30      Call Usrgen(nffcn,t,u)
        Goto 90
c.
50      Continue
        Icnt = (nffcn-1)*(nu+1) + 1
        nend = (nffcn-1)*(nu+1) + nu
        Ctim = 0
        Dtim = 0
        i = 1
        do while (t .gt. dtim)
          dtim = DS(i,icnt)
          i = i + 1
c       write(*,*)'t=',t,'  dtim=',dtim,'  i=',i
        enddo
c.
        inxt = i + 1
        ctim = DS(inxt,icnt)
        tratio = (t - dtim)/(ctim - dtim)
c       write(*,*)'ctim=',ctim,'   tratio=',tratio
c.
        if(t .gt. 0) then
          uratio = ((dtim - t)/t)
        else
          uratio = 0
        endif
c.
          If(uratio .lt. .02)then
            m = 0
            do 54 j=icnt,nend
              k    = j+1
              m    = m+1
54            u(m) = DS(i,k)
          else
            m = 0
            do 56 j=icnt,nend
              k    = j+1
              m    = m+1
              u3   = ds(inxt,k) - ds(i,k)
56            u(m) = tratio*u3 + ds(i,k)
          endif
c.
90      Continue
c.
c       do 91 j=1,nu
c91     write(*,*)u(1),u(2)
        return
        end
```

```
c.*************************************************************
c.----Subroutine ENPMEN
c.
c.----Purpose: This subroutine is the standard interface used for
c.              entering the parameters for multiple driving
c.              systems.  To use this subroutine the user must set
c.              up a datafile for each driving system.  Each
c.              datafile is 1 U vector.  These datafiles are then
c.              read into an array (DS) which is nothing more than
c.              a set of column U vectors.  Each column is then
c.              passed to ENPGEN as required for analysis.
c.
        subroutine ENPMEN
c.
c. Define passed variables
c.
        Dimension fname(10), ndpt(10)
        Character  fname*30, isel*1
        Real DS
        Integer ndpt
c.
        Common /intvar/ T1,T2,h,xin(20),nx,ny,ns,nu,NFF,IEND,UAF
        Common /drvsys/ DS(1500,10), ICLS
c.
        Write(*,108)
5       Write(*,104)
        Read (*,201) Icls
        If (Icls .eq. 'e' .or. Icls .eq. 'E')Then
          goto 10
        Elseif(Icls .eq. 'u' .or. Icls .eq. 'U')then
          goto 30
        Elseif(Icls .eq. 'd' .or. Icls .eq. 'D')then
          goto 50
        Else
          goto 5
        Endif
c.
10      do 12 i=1,10
          do 12 j=1,1500
12          ds(j,i) = 0
c.
        Write(*,109)
        Iend=1
        If(ICLS .eq. 'd' .or. ICLS .eq. 'D')Iend = 2
        do 16 i=1,nff
          Write(*,106) i
          Read (*,200) fname(i)
          Open (12,file=fname(i),status='unknown')
            do 15 j=1,nu
             Icnt = i*Iend - (Iend - 1)
              do 14 k=1,Iend
                Read (12,204,end=16) DS(j,icnt)
                Icnt = Icnt + 1
14            Continue
15          Continue
          Close(12)
16      Continue
```

```
c.
18      Write (*,101)
        Read  (*,201) isel
          If (isel .eq. 'c' .or. isel .eq. 'C') goto 90
          If (isel .eq. 'v' .or. isel .eq. 'V') goto 20
          If (isel .eq. 'r' .or. isel .eq. 'R') goto 10
          goto 18
c.
cccccc20        Do 22 i=1,nff
20        Write (*,102) (fname(i), i=1,nff)
            Do 24 j=1,nu
24             Write (*,103) j,(DS(j,k), k=1,nff)
cccccc    Write (*,*) '      '
cccccc22        continue
        goto 18
c.
30      Write(*,105)
        Read (*,201) isel
        If(Isel .eq. 'y' .or. Isel .eq. 'Y')Then
         goto 32
        Elseif(Isel .eq. 'n' .or. Isel .eq. 'N')Then
         goto 32
c.
        Else
         goto 10
        Endif
c.
32      Call Usrmen
        Goto 90
c.
50      do 52 i=1,10
          do 52 j=1,1500
52          ds(j,i) = 0.
c.
        Write(*,109)
        do 53 i=1,nff
        write(*,106)i
        read (*,200)fname(i)
        Write(*,107)i
53      read (*,205)ndpt(i)
c.
     do 58 k=1,nff
     open(unit=12,file=fname(k),form='formatted',status='unknown')
          m = (k-1)*(nu+1) + 1
          l = m + nu
            do 56 j = 1,ndpt(k)
56              Read (12,*,end=57) (DS(j,i),i=m,l)
57        Close (unit=12)
58      Continue
c.

c       do 60 j=1,ndpt(1)
c60     write(*,*)ds(j,1),ds(j,2),ds(j,3),ds(j,4),ds(j,5),ds(j,6)
c
90      continue
c.
```

```
101     format (17x,'Please select one of the following',/,
     +          17x,'      C = Continue',/,
     +          17x,'      V = View the data entered',/,
     +          17x,'      R = Re-enter the datafiles',/,
     +          17x,'      ',$)
102     format (17x,' ',//,20x,'DS Matrix Build',/,
     +          1x,'File Names:   ',a12,' ',a12,' ',a12,' ',a12,
     +  ' ',a12,' ',a12,//,
     +          5x,'----------------------------------------')
103     format (3x,'U comp #',i3,'=',e11.5,'  ',e11.5,'  ',e11.5,
     +  '  ',e11.5,'  ',e11.5,'  ',e11.5)
108     format (11x,'    ',/,
     +          13x,'Enter System Driver Parameters',/,
     +          13x,'----------------------------------------')
104     format (17x,'How is The Forcing System Being Modeled',/,
     +          17x,'----------------------------------',/,
     +          17x,'   E = Enport model generator',/,
     +          17x,'   U = User written generator',/,
     +          17x,'   D = Experimental data',/,
     +          17x,'   ',$)
105     format (13x,'Do you wish to use your own menu',/,
     +          13x,'      Y = Yes      N = No      ',/,
     +          13x,'      ',$)
109     format (17x,'Enter Datafile Information ')
106     format (17x,' Name of File NO:',i2,'                    ',$)
107     format (17x,' Number of Data Pts for File NO:',i2,'  ',$)
c.
200     format (a30)
201     format (a1)
204     format (d12.5)
205     format (i4)
c.
        return
        end
c.*****************************************************************
c.*****************************************************************
c.----Subroutine ENPAFN
c.
c.----Purpose: This subroutine determines the maximum value from
c.             the objective function set.  In addition all AF
c.             components which are elements of the obj fcn set
c.             will have their return values calculated.
c.             In addition the objective function value used by
c.             optdes will be calculated.
c.
c.----Variables used:
c.         Passed variables
c.         ------------------------------------
c.              af  = the analysis function array
c.              ny  = number of output variables
c.              nff = number of forcing functions
c.              npm = number of performance variables
c.              nobj= size of objective function
c.
        Subroutine ENPAFN(NPM,AV,AF)
```

```
c.
c. Define passed variables
c.
      Dimension zf(10), af(1), av(1), dumaf(10,10,100)
      Double Precision dumaf, af, av
      Real zf, amax, acur
      Integer rfact, ncont, npm, naf, nobj, obfcn, icstr, iavn
c.
      Common /intvar/ T1,T2,h,xin(20),nx,ny,ns,nu,NFF,IEND,UAF
      Common /objfcn/ nobj,obfcn(10),icstr,iavn
c.
c. Determine repeating factor and number of calculated af
c. constraints.
      rfact = npm*ny
      ncont = npm*ny*nff
c.
c. Set AF vector
c.
      If (obfcn(1) .lt. 0) then
c.
c. Sort the af vector
c.
c. i = num of forcing functions
c. j = num of performance criteria
c. k = num of output variables
c.
        do 20 i = 1,nff
         do 20 j = 1,npm
          do 20 k = 1,ny
            iaf =  (i-1)*ny*npm +(j-1)*ny + k
20      dumaf(i,j,k) = af(iaf)
c.
c. Determine max values for each constraint
c.
        do 35 j = 1,npm
          do 30 k = 1,ny
            naf  = (j-1)*npm + k
            amax = dumaf(1,j,k)
              do 25 i = 1,nff
                acur = dumaf(i,j,k)
                if(acur .gt. amax)then
                  zf(naf) = acur
                else
                  zf(naf) = amax
                endif
25            continue
30        continue
35      continue
c.
c. Set the AF vector obj fcn constraints
c.
        do 40 i=icstr,ncont,rfact
40            AF(i) = AF(i) - AV(iavn)
c.
```

```
c. Set obj fcn
c.
        AF(ncont+1) = AV(iavn)**2
c.
       ELSE
c.
c. Determine obj fcn subset
c.
         Do 50 i=1,nobj
           iaf = obfcn(i)
50         zf(i) = af(iaf)
c.
c. Determine Max value
c.
         amax = zf(1)
         Do 55 i=1,nobj
           if(zf(i) .gt. amax)then
            amax = zf(i)
           else
            amax = amax
           endif
55       continue
c.
c. Set the AF vector obj fcn constraints
c.
         Do 60 i=1,nobj
           iaf = obfcn(i)
60         AF(iaf) = AF(iaf) - AV(iavn)
c.
c. Set the Obj Fcn value
c.
        AF(ncont+1) = AV(iavn)**2
c.
       ENDIF
c.
       return
       end
c.*****************************************************************
c.----Subroutine enpobj
c.
c.----Purpose: This subroutine sets the objective function value
c.             to be the maximum of the ith constraint for a set
c.             of j' forcing functions. (where j' = 1,2...j
c.             forcing functions).  This subroutine should be
c.             used when multiple forcing functions are used for
c.             analysis.
c.
c.----Variables used:
c.         Passed variables
c.         ------------------------------------
c.              NOBJ  = determines the size of the
c.                      objective function set.
c.              obfcn = the objective function set.
c.              icstr = the ith performance measurement.
c.              iavn  = analysis variable number
c.                      used for the obj fcn.
```

```
C.
      subroutine enpobj(npm)
c.
c. Define passed variables
c.
      Integer npm, nobj, obfcn, icstr, iavn
c.
      Common /intvar/ T1,T2,h,xin(20),nx,ny,ns,nu,NFF,IEND,UAF
      Common /objfcn/ nobj,obfcn(10),icstr,iavn
c.
      do 5 i=1,10
5       obfcn(i)=0
c.
c.
      If (nobj .eq. nff*npm) then
         obfcn(1) = -1
      Else
        Write(*,104)
        Write(*,102)
        Read (*,200) iavn
         Do 10 i=1,nobj
           Write(*,101) i
10         Read (*,200) obfcn(i)
      Endif
c.
c.
c.
      If(obfcn(1) .lt. 0)then
       Write(*,104)
       Write(*,102)
       Read (*,200) iavn
       Write(*,103)
       Read (*,200) icstr
      Else
       Continue
      Endif
c.
104   format ('  ',/,
     +         11x,'Enter Index Numbers for the Following',/,
     +         11x,'----------------------------------------')
101   format (11x,'  ',/,
     +         11x,'AF Objective Function Set Member',i3,' ',$)
102   format (11x,'AV Objective Function Comparator      ',$)
103   format (11x,'Perf Meas used as the Obj Fcn         ',$)
c.
200   format (i3)
c.
      return
      end
C.*****************************************************************
C.----Subroutine Matwr
C.
      Subroutine MATWR(string)
C.
      Character String*72
C.
      Write (*,100) string
```

```
C.
100     Format (a72)
C.
        return
        end
C*************************************************************
C
C.----Subroutine afbild
C.
        Subroutine afbild(AV,AF,NMEM)
c.
      DOUBLE PRECISION AV(10), AF(100)
C.
C... Call data dialogue subroutine
C.
        INTEGER NYY, NYCNT, NFF, NDX, INDX, obfcn, nobj, iavn,
     +  NPM, NMEM, icstr
        PARAMETER(NYY=20, NYCNT=4096)
        DIMENSION FCN(nyy), CSTR(100)
        CHARACTER FCN*32, IPAR*5, AGN*1, CSTR*2
        Common /objfcn/ nobj,obfcn(10),icstr,iavn
        COMMON /INTVAR/ T1,T2,H,XIN(20),NX,NY,NS,NU,NFF,OS,IEND
        COMMON /PAROPT/ IPAR
c.
        loc1 = index(ipar,'m')
        loc2 = index(ipar,'a')
        loc3 = index(ipar,'r')
        loc4 = index(ipar,'v')
        loc5 = index(ipar,'s')
        loc6 = index(ipar,'x')
        loc7 = index(ipar,'n')
C.
        if (loc1 .gt. 0) fcn(loc1) = 'maximum value'
        if (loc2 .gt. 0) fcn(loc2) = 'mean value'
        if (loc3 .gt. 0) fcn(loc3) = 'root mean square'
        if (loc4 .gt. 0) fcn(loc4) = 'max violation'
        if (loc5 .gt. 0) fcn(loc5) = 'minimum value '
        if (loc6 .gt. 0) fcn(loc6) = 'absolute max '
        if (loc7 .gt. 0) fcn(loc7) = 'absolute min '
c.
        NPM = 0
        if(loc1 .gt. npm) npm = loc1
        if(loc2 .gt. npm) npm = loc2
        if(loc3 .gt. npm) npm = loc3
        if(loc4 .gt. npm) npm = loc4
        if(loc5 .gt. npm) npm = loc5
        if(loc6 .gt. npm) npm = loc6
        if(loc7 .gt. npm) npm = loc7
c.
        Ifin = Nff*Ny
c.
        DO 5 I=1,Nff*Npm*Ny
5         Cstr(I) = ' '
c.
```

```
        If (Nobj .eq. 1) Then
          Write(*,105)
          Read (*,200) icstr
          cstr(icstr) = '*'
        Elseif (obfcn(1) .lt. 0) Then
          Do 10 i=Icstr,Nff*Npm*Ny,Ny
10          Cstr(I) = '**'
        Else
          Do 15 I=1,10
            Indx = Obfcn(I)
            If(Indx .ne. 0)
     +      Cstr(Indx)='**'
15        Continue
        Endif
c.
        Ip = 1
        Do 25 j=1,Nff
        Do 25 i=1,Npm,1
c         IP = (j-1)*Ny*Npm + (I-1)*NY + 1
c         IEP = (j-1)*Ny*Npm + i*ny
          Write(*,104)fcn(i)
            Do 20 K=1,NY
              Write(*,106) Cstr(Ip), Ip, K
20            Ip = Ip + 1
25      Continue
c.
        If (Nobj .eq. 1) Then
          Write(*,108)
        Else
          Write(*,103)nmem
          Write(*,107)
        Endif
c.
103     format(10x,' *AF(',i2,') Objective Function')
104     format(5x,a32)
105     format(11x,'Enter AF Index # of the Objective Function')
106     format(10x,a2,'AF(',i2,')  Y-Output(',i2,')')
107     format(' ',/,10x,'** = Objective Function Set Member')
108     format(' ',/,10x,'*  = Objective Function')
C.
200     format(i4)
C.
      RETURN
      END
C.*************************************************************
```

F.3 DSMOPT.COM

This is the DEC command file that controls the interface environment. From this file the user can execute any phase of the model building or optimization process desired.

COMPUTER PROGRAM

DSMOPT.COM

WRITTEN
BY

```
$!*************************************************************
$! ----Filename: DSMOPT.COM
$!*************************************************************
$! ----Purpose:  Provide user with menu driven interface between
$!               the ENPORT simulation/modeling software and
$!               OPTDES.BYU optimization software.
$!*************************************************************
$! ----Written BY: Mark Davis
$!*************************************************************
$! ----Date Commissioned: September 20, 1991
$!*************************************************************
$! ----External Files: ENPMOD.EDT - SYSPRM.EDT - SYSDAT.EDT
$!                     CDES2.OPT  - ENPLIN.USR
$!*************************************************************
$! ----Setup: When installed this command file must have two
$!            symbol-names changed:
$!              OPTFI = Directory name where OPTDES Subroutines
$!                      reside.
$!              USEFI = Directory name where INTERFACE
$!                      Subroutines reside.
$!              ENPFI = Directory name where ENPORT model files
$!                      reside
$!*************************************************************
$! ----Modification History:
$!
$!   Date     Programmer               Description of Changes
$!  ------   ------------   ------------------------------------
$!
$!*************************************************************
$ set noverify
$ optfi = "PROJECT6:[optdes]"
$ usefi = "user3:[davism.caltech]"
$
$ dir_nam = f$directory()
$ enpfi = dir_nam
$
$ fi_exist = f$file_attributes("usrgen.for","uic")
$ the_message == f$message($status)
$ good_gen = f$extract(10,10,the_message)
$ if good_gen .eqs. "NOSUCHFILE" then goto usegen
$ genfi = enpfi
$ goto chkmen
$
$ usegen:
$ genfi = usefi
$
$ chkmen:
$ fi_exist = f$file_attributes("usrmen.for","uic")
$ the_message == f$message($status)
$ good_men = f$extract(10,10,the_message)
$ if good_men .eqs. "NOSUCHFILE" then goto usemen
$ menfi = enpfi
$ goto chkafn
$
$ usemen:
$ menfi = usefi
```

```
$
$ chkafn:
$ fi_exist = f$file_attributes("usrafn.for","uic")
$ the_message == f$message($status)
$ good_afn = f$extract(10,10,the_message)
$ if good_afn .eqs. "NOSUCHFILE" then goto useafn
$ afnfi = enpfi
$ goto continueon
$
$ useafn:
$ afnfi = usefi
$
$ continueon:
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output "     This is the MSU Simulation/Optimization"
$ write sys$output "                  Design System              "
$ write sys$output "
$ again:
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output "     ****************************************"
$ write sys$output "     *                MAIN  MENU            *"
$ write sys$output "     ****************************************"
$ write sys$output "     * Please make the following selection *"
$ write sys$output "     *     Enter E = execute ENPORT         *"
$ write sys$output "     *           D = create/edit DATAFILES  *"
$ write sys$output "     *             O  = execute OPTDES      *"
$ write sys$output "     *             C  = execute CHECKFILE   *"
$ write sys$output "     *             X  = exit the session    *"
$ write sys$output "     ****************************************"
$ inquire choice   ""
$ if choice .eqs. "C" then goto begin
$ if choice .eqs. "D" then goto datafi
$ if choice .eqs. "E" then goto enport
$ if choice .eqs. "O" then goto begin
$ if choice .eqs. "X" then goto endit
$ write sys$output "enter  c  or  o  or  e  or x  only try again"
$ goto again
$
$ enport:
$ assign/user_mode sys$command sys$input
$ enport
$ goto again
$
$ datafi:
$ cntdf = 1
$ write sys$output ""
$ write sys$output ""
```

```
$ write sys$output ""
$ write sys$output "    ****************************************"
$ write sys$output "    * How many Datafiles are you creating  *"
$ write sys$output "    *                                      *"
$ write sys$output "    *  Note: Each set of system driving    *"
$ write sys$output "    *        conditions must be described  *"
$ write sys$output "    *        by a separate datafile.       *"
$ write sys$output "    ****************************************"
$ inquire numdf ""
$
$ entdf:
$ write sys$output ""
$ write sys$output "    ****************************************"
$ write sys$output "    * Create/Edit datafile No: "'cntdf'"    *"
$ write sys$output "    *                        ---           *"
$ write sys$output "    * Please enter Datafile Name           *"
$ write sys$output "    *    Example: DRSYS1 (limit 6 char)    *"
$ write sys$output "    *                                      *
$ write sys$output "    *  Note: Do not enter filetype .dat    *"
$ write sys$output "    ****************************************"
$ inquire dfnam ""
$ dfnmck:
$ write sys$output ""
$ write sys$output "    ****************************************"
$ write sys$output "    * You Entered:  "'dfnam'"               *"
$ write sys$output "    *               -------                *"
$ write sys$output "    * Is this correct (Y or N)             *"
$ write sys$output "    ****************************************"
$ inquire dfcor ""
$ if dfcor .eqs. "Y" then goto dfedit
$ if dfcor .eqs. "N" then goto entdf
$ goto dfnmck
$
$ dfedit:
$ assign/user_mode sys$command sys$input
$ edit/edt 'dfnam'.dat
$ if cntdf .eq. numdf then goto again
$ cntdf = cntdf + 1
$ goto entdf
$
$ begin:
$ lnkchk = 1
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output "     ****************************************"
$ write sys$output "     * Please enter model name from ENPORT *"
$ write sys$output "     *                                      *"
$ write sys$output "     *    EXAMPLE:   MODEL0                 *"
$ write sys$output "     *                                      *"
$ write sys$output "     *    Note: Do not include filetype     *"
$ write sys$output "     ****************************************"
$ inquire staeqs ""
$
```

```
$ eqns:
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output "     ****************************************"
$ write sys$output "     * You entered:                          *"
$ write sys$output "     *                                       *"
$ write sys$output "     *           "'staeqs'"                  *"
$ write sys$output "     *              ------                   *"
$ write sys$output "     * Is this correct (y or n)              *"
$ write sys$output "     ****************************************"
$ inquire filcor ""
$ if filcor .eqs. "Y" then goto sysparm
$ if filcor .eqs. "N" then goto begin
$ goto eqns
$
$ sysparm:
$ edp :== "edit/edt/command="'USEFI'"enpmod.edt"
$ edp 'staeqs'.usr
$ fortran zzzzzz.for
$ syp :== "edit/edt/command="'USEFI'"sysprm.edt"
$ syp 'staeqs'.dat
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output "  Please note the system parameter values"
$ write sys$output ""
$ type zzzzzz.dat
$ del zzzzzz.dat;*
$ syf :== "edit/edt/command="'USEFI'"sysdat.edt"
$ syf 'staeqs'.dat
$ if choice .eqs. "C" then goto chkfi
$ if choice .eqs. "O" then goto optdes
$
$ chkfi:
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output "     ****************************************"
$ write sys$output "     *             CHECKFILE MENU           *"
$ write sys$output "     ****************************************"
$ write sys$output "     * Please make the following selection *"
$ write sys$output "     *    Enter  C = compile all user progs*"
$ write sys$output "     *           I = compile indiv programs*"
$ write sys$output "     *           L = link programs         *"
$ write sys$output "     *           R = run RUNPRO.EXE        *"
$ write sys$output "     *           M = return to main menu   *"
$ write sys$output "     *           X = end this session      *"
$ write sys$output "     ****************************************"
$ inquire compl    ""
$ if compl .eqs. "C" then goto compall
$ if compl .eqs. "I" then goto compind
$ if compl .eqs. "R" then goto runpro
$ if compl .eqs. "M" then goto again
$ if compl .eqs. "L" then goto linkfi
$ if compl .eqs. "X" then goto enditt
$ write sys$output "     enter  A  or  I  or  L  only try again"
```

```
$ goto chkfi
$
$ runpro:
$ assign/user_mode sys$command sys$input
$ run runpro
$ goto chkfi
$
$ compall:
$ if lnkchk .eq. 2 then goto lnkmsg
$ fortran usrgen.for
$ fortran usrmen.for
$ fortran usrafn.for
$ goto linkfi

$ compind:
$ count = 1
$ write sys$output ""
$ write sys$output "  ******************************************"
$ write sys$output "  *   The number of programs to be compiled*"
$ write sys$output "  ******************************************"
$ inquire numcom   ""

$ loop1:
$ write sys$output ""
$ write sys$output "  ******************************************"
$ write sys$output "  *          Enter the program name        *"
$ write sys$output "  ******************************************"
$ inquire prog     ""
$ fortran/debug/nooptimize 'prog'.for
$ if count .eq. numcom then goto linkfi
$ count = count + 1
$ goto loop1

$ linkfi:
$ if lnkchk .eq. 2 then goto lnkmsg
$ del 'enpfi'runpro.exe;*
$ link/executable=runpro 'usefi'anamain.obj, 'usefi'anasub.obj, -
                         'enpfi'zzzzzz.obj,  'genfi'usrgen.obj, -
                         'menfi'usrmen.obj,  'afnfi'usrafn.obj
$
$ write sys$output  "                                            "
$ write sys$output  "   Executable file RUNPRO has been created "
$ write sys$output  "                    ------                  "
$ del zzzzzz.for;*
$ del zzzzzz.obj;*
$ lnkchk = 2
$ goto chkfi
$
```

```
$ optdes:
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output "  ****************************************"
$ write sys$output "  *             OPTIMIZATION MENU        *"
$ write sys$output "  ****************************************"
$ write sys$output "  *  Select one of the following options  *"
$ write sys$output "  *          C = compile all user progs   *"
$ write sys$output "  *          I = compile individual progrs*"
$ write sys$output "  *          L = link to optimization files*"
$ write sys$output "  *          S = create setup executable  *"
$ write sys$output "  *         RC = run CONVEN.EXE           *"
$ write sys$output "  *         RS = run SETUP.EXE            *"
$ write sys$output "  *         RD = run CDESIGN.EXE          *"
$ write sys$output "  *          M = return to main menu      *"
$ write sys$output "  *          X = end session              *"
$ write sys$output "  ****************************************"
$ inquire optdl ""
$ if optdl .eqs. "C" then goto comana
$ if optdl .eqs. "I" then goto indana
$ if optdl .eqs. "L" then goto linana
$ if optdl .eqs. "S" then goto creset
$ if optdl .eqs. "RC" then goto conven
$ if optdl .eqs. "RS" then goto setup
$ if optdl .eqs. "RD" then goto cdesign
$ if optdl .eqs. "M" then goto again
$ if optdl .eqs. "X"  then goto enditt
$ goto optdes
$
$ conven:
$ assign/user_mode sys$command sys$input
$ run conven
$ goto optdes
$
$ setup:
$ assign/user_mode sys$command sys$input
$ run setup
$ goto optdes
$
$ cdesign:
$ assign/user_mode sys$command sys$input
$ run cdesign
$ goto optdes
$
$ comana:
$ if lnkchk .eq. 2 then goto lnkmsg
$ fortran anasub.for
$ fortran zzzzzz.for
$ fortran usrgen.for
$ fortran usrmen.for
$ fortran usrafn.for
$ goto optdes
$
```

```
$ indana:
$ count = 1
$ write sys$output ""
$ write sys$output "  ****************************************"
$ write sys$output "  *   The number of programs to be compiled*"
$ write sys$output "  ****************************************"
$ inquire numcom   ""

$ loop1:
$ write sys$output ""
$ write sys$output "  ****************************************"
$ write sys$output "  *          Enter the program name       *"
$ write sys$output "  ****************************************"
$ inquire prog     ""
$ fortran/debug/nooptimize 'prog'.for
$ if count .eq. numcom then goto optdes
$ count = count + 1
$ goto loop1

$ linana:
$ if lnkchk .eq. 2 then goto lnkmsg
$ delete 'enpfi'conven.exe;*
$ link/executable='enpfi'conven 'optfi'conven.obj, -
                                'optfi'consub.obj, -
                                'optfi'meneds.obj, -
                                'optfi'menutl.obj, -
                                'optfi'menpro.obj, -
                                'optfi'menout.obj, -
                                'optfi'sysvms.obj, -
                                'usefi'anasub.obj, -
                                'enpfi'zzzzzz.obj, -
                                'genfi'usrgen.obj, -
                                'menfi'usrmen.obj, -
                                'afnfi'usrafn.obj
$ write sys$output  "                                          "
$ write sys$output  "  Executable file CONVEN has been created "
$ write sys$output  "                  ------                  "
$
$ delete 'enpfi'cdesign.exe;*
$! cdes2/options
$ link/executable='enpfi'cdesign -
                      'usefi'anasub.obj, 'enpfi'zzzzzz.obj, -
                      'genfi'usrgen.obj, 'menfi'usrmen.obj, -
                      'afnfi'usrafn.obj, 'usefi'cdes2/options
$ write sys$output  "                                          "
$ write sys$output  " Executable file CDESIGN has been created "
$ write sys$output  "                  -------                 "
$ del zzzzzz.for;*
$ del zzzzzz.obj;*
$ lnkchk = 2
$ goto optdes
$
$ lnkmsg:
$ write sys$output  " "
$ write sys$output  " The files have been linked already.  You "
$ write sys$output  " must return to the main menu to create a "
$ write sys$output  " new executable file.                     "
```

```
$ if choice .eqs. "C" then goto chkfi
$ if choice .eqs. "O" then goto optdes
$ goto again
$
$ creset:
$ link/executable='enpfi'setup   'optfi'setup.obj,   -
                                 'optfi'dissub.obj,  -
                                 'optfi'varsub.obj,  -
                                 'optfi'funsub.obj,  -
                                 'optfi'comsub.obj,  -
                                 'optfi'sdbsub1.obj, -
                                 'optfi'sdbsub2.obj, -
                                 'optfi'menset.obj,  -
                                 'optfi'meneds.obj,  -
                                 'optfi'menutl.obj,  -
                                 'optfi'menpro.obj,  -
                                 'optfi'menout.obj,  -
                                 'optfi'sysvms
$ write sys$output  "                                                "
$ write sys$output  "    Executable file SETUP has been created"
$ write sys$output  "                    -----                       "
$ goto optdes
$
$ enditt:
$ del zzzzzz.dat;*
$ endit:
$ exit
```

F.5 SYSDAT.EDT

This is the DEC command edit file. This file will execute a series of edit commands. These commands are used to edit the ENPORT .DAT output file to read the number of state variables, outputs, and inputs when ANAPRE is executed.

COMPUTER PROGRAM

SYSPRM.EDT

WRITTEN BY

MARK DAVIS

```
set nofnf
set nonum
set nosummary
set noverify
del 24 thru 400
del 1 thru 17
sub/outputs/OUTPUTS (Y-VECTOR)/1 thru 100/notype
sub/INPUTS/INPUTS (U-VECTOR)/1 thru 100/notype
SU/STATES/STATES (X-VECTOR)/1 THRU 100/NOTYPE
ex zzzzzz.dat
```

```
set nofnf
set nonum
set nosummary
set noverify
del 24 thru 400
del 22
del 20
del 1 thru 18
ex zzzzzz.dat
```

F.6 CDES2.OPT

This is the DEC command options file. This file is used when an executable line is to long for exection from within the command file. This file contains all the names of the subroutines used for linking with ANASUB.FOR to create the executable file CDESIGN.EXE.

COMPUTER PROGRAM

CDES2.OPT

WRITTEN BY

MARK DAVIS

```
PROJECT6:[optdes]design.obj, -
PROJECT6:[optdes]calsub.obj,    PROJECT6:[optdes]displa.obj,  -
PROJECT6:[optdes]profun.obj,    PROJECT6:[optdes]propro.obj,  -
PROJECT6:[optdes]params.obj,    PROJECT6:[optdes]runalg.obj,  -
PROJECT6:[optdes]setvar.obj,    PROJECT6:[optdes]histor.obj,  -
PROJECT6:[optdes]explor.obj,    PROJECT6:[optdes]grgalg1.obj, -
PROJECT6:[optdes]grgalg2.obj,   PROJECT6:[optdes]sqpalg.obj,  -
PROJECT6:[optdes]qpsolv.obj,    PROJECT6:[optdes]slpalg.obj,  -
PROJECT6:[optdes]lpsolv.obj,    PROJECT6:[optdes]approx1.obj, -
PROJECT6:[optdes]approx2.obj,   PROJECT6:[optdes]tolers.obj,  -
PROJECT6:[optdes]combin1.obj,   PROJECT6:[optdes]combin2.obj, -
PROJECT6:[optdes]plosub.obj,    PROJECT6:[optdes]grasub.obj,  -
PROJECT6:[optdes]scrtek.obj,    PROJECT6:[optdes]penHP.obj,   -
PROJECT6:[optdes]meneds.obj,    PROJECT6:[optdes]menutl.obj,  -
PROJECT6:[optdes]menpro.obj,    PROJECT6:[optdes]menout.obj,  -
PROJECT6:[OPTDES]sysvms.obj
```

APPENDIX G

APPLICATION EXAMPLE DATA

APPENDIX G

APPLICATION EXAMPLE DATA

G.1 Introduction

The following information was used to execute the application example solved in chapter 3. The input vector information describes the operating parameters of the bond graph. Items such as mass, trf, frequencies and amplitudes are set from these data files. The ENPORT output files are generated by ENPORT when the Fortran file is generated. The optimization data files were generated from OPTDES. These files contain information describing the setup of the optimization problem

G.2 Input Vector (U) Datafiles

These data files profile the bond graph model used in the application example.

DATA FILE

PROF1.DAT

WRITTEN BY

MARK DAVIS

```
3.9270000
9.81750000
1.0000000000
0.0
0.5000000
0.8666667
0.8
1.12
1.067
1.387
.267
1.638
4.094
-1.638
-.500000
-.8666667
9.01
139.75
3416.7
3.0
3.0
0.0
0.0
0.0
0.0
0.0
0.83300000000
3.33330000000000
6.66667000000
```

DATA FILE

PROF2.DAT

WRITTEN BY

MARK DAVIS

```
6.283
0.0
0.0
0.0
0.7071
0.7071
100.0
0.0
100.0
0.0
.125
1.0493
0.0
-1.0493
0.0
0.0
9.01
139.75
3416.7
3.0
3.0
0.0
0.0
0.0
0.0
0.0
0.8333
3.3333
6.6667
```

DATA FILE

PROF3.DAT

WRITTEN BY

MARK DAVIS

```
50.2655
0.0
0.0
0.0
1.0
0.0
100.0
0.0
100.0
0.0
0.1250
8.394
0.0
-8.394
1.0
0.0
9.01
139.75
3416.7
3.0
3.0
0.0
0.0
0.0
0.0
0.0
0.8333
3.3333
6.6667
```

G.3 ENPORT Output Files

These files were produced by ENPORT. They describe the bond graph model. The .USR file contains the state equations written in compilable fortran code. The remaining data files contain information about the structure of the bond graph model.

FORTRAN FILE

USR01.USR

GENERATED FROM

ENPORT

```
C****** MATRIXx User_Code_Block file written by ENPORT ***********
C
C  File Name: USER01.USR
C
C  5 Dof Suspension System W/Variable Flow Source - WO/Grav.
C  Effects
C  System_Build User_Code_Block USR01
C
C*****************************************************************
C
      SUBROUTINE USR01(INFO,T,U,NU,X,XDOT,NX,Y,NY,RP,IP)
C
      IMPLICIT DOUBLE PRECISION (A-Z)
      CHARACTER         STRNG*8
      DOUBLE PRECISION T, U(*), X(*), XDOT(*), Y(*), RP(*)
      INTEGER           IP(*), INFO(4), NU, NX, NY, I
      LOGICAL           INIT, STATE, OUTPUT
      EXTERNAL          MATWR
C
C*****************************************************************
C
      INIT   = INFO(2).NE.0
      STATE  = INFO(3).NE.0
      OUTPUT = INFO(4).NE.0
C
C---- Check the problem size...
C
      IF (INIT) THEN
        STRNG='****'
        IF (NU.NE.29) STRNG='inputs.'
        IF (NX.NE.10) STRNG='states.'
        IF (NY.NE. 6) STRNG='outputs.'
        IF (STRNG.NE.'****') THEN
          INFO(1) = -2
          CALL MATWR(' ')
          CALL MATWR(' *** Error calling USR01.')
          CALL MATWR('     Incorrect number of '//STRNG)
          RETURN
        ENDIF
      ENDIF
C
C---- Get the time and the states X...
C
      IF (STATE.OR.OUTPUT) THEN
       TIME= T
C
       Q5Q   = X( 1)
       P5M   = X( 2)
       Q3Q   = X( 3)
       P2M   = X( 4)
       P3J   = X( 5)
       P1M   = X( 6)
       Q1Q   = X( 7)
       Q2Q   = X( 8)
       P4M   = X( 9)
       Q4Q   = X(10)
```

```
C
C
        W1    = U( 1)
        W2    = U( 2)
        PHI1  = U( 3)
        PHI2  = U( 4)
        PHI3  = U( 5)
        PHI4  = U( 6)
        T1    = U( 7)
        T2    = U( 8)
        T3    = U( 9)
        T4    = U(10)
        T5    = U(11)
        AMP1  = U(12)
        AMP2  = U(13)
        AMP3  = U(14)
        PHI5  = U(15)
        PHI6  = U(16)
        I1    = U(17)
        I2    = U(18)
        I3    = U(19)
        I4    = U(20)
        I5    = U(21)
        SE1   = U(22)
        SE2   = U(23)
        SE3   = U(24)
        SE4   = U(25)
        SE5   = U(26)
        TF1   = U(27)
        TF2   = U(28)
        TF3   = U(29)
C
C
        C1    = RP( 1)
        C2    = RP( 2)
        C3    = RP( 3)
        R1    = RP( 4)
        R2    = RP( 5)
        R3    = RP( 6)
C
C---- Evaluate the system equations...
C
        C4    =  1.80000E+04
        C5    =  1.80000E+04
        R4    =  6.00000E+01
        R5    =  6.00000E+01
        S46S  =  AMP3        * SIN( W1          *TIME  )
        S2S   =  AMP1        * SIN( W1          *TIME  )
C......STEP
        IF (TIME  .GE. T1         ) THEN
          S48S  =  1.00000E+00
        ELSE
          S48S  = 0.
        ENDIF
        S49S  =  AMP2        * SIN( W2          *TIME  )
```

```
C......STEP
       IF (TIME  .GE. 0.00000E+00) THEN
         S6S   =  PHI2
       ELSE
         S6S   = 0.
       ENDIF
       S51S  =  AMP2        * COS( W2          *TIME  )
 C......STEP
       IF (TIME  .GE. 0.00000E+00) THEN
         S8S   =  PHI1
       ELSE
         S8S   = 0.
       ENDIF
 C......STEP
       IF (TIME  .GE. T2          ) THEN
         S9S   = -1.00000E+00
       ELSE
         S9S   = 0.
       ENDIF
 C......STEP
       IF (TIME  .GE. 0.00000E+00) THEN
         S21S  =  PHI3
       ELSE
         S21S  = 0.
       ENDIF
       S22S  =  AMP1        * COS( W1          *TIME  )
 C......STEP
       IF (TIME  .GE. 0.00000E+00) THEN
         S23S  =  PHI4
       ELSE
         S23S  = 0.
       ENDIF
 C......STEP
       IF (TIME  .GE. T4          ) THEN
         S29S  = -1.00000E+00
       ELSE
         S29S  = 0.
       ENDIF
 C......STEP
       IF (TIME  .GE. 0.00000E+00) THEN
         S25S  =  PHI5
       ELSE
         S25S  = 0.
       ENDIF
 C......STEP
       IF (TIME  .GE. 0.00000E+00) THEN
         S28S  =  PHI6
       ELSE
         S28S  = 0.
       ENDIF
 C......STEP
       IF (TIME  .GE. T3          ) THEN
         S24S  = -1.00000E+00
       ELSE
         S24S  = 0.
       ENDIF
```

```
C......STEP
       IF (TIME  .GE. T5         ) THEN
         S54S  =  1.00000E+00
       ELSE
         S54S  = 0.
       ENDIF
 C......STEP
       IF (TIME  .GE. T3         ) THEN
         S55S  =  1.00000E+00
       ELSE
         S55S  = 0.
       ENDIF
       S57S  =  1.00000E+00
       S58S  =  1.00000E+00
       E5W   =  SE5
       E2W   =  SE2
       E1W   =  SE1
       E4W   =  SE4
       MTF1  =  TF1
       MTF2  =  TF2
       MTF3  =  TF3
       S10S  =  1.00000E+00 *S2S    *S48S
       S11S  =  1.00000E+00 *S49S   *S6S
       S12S  = -1.00000E+00 *S51S   *S8S
       S30S  =  1.00000E+00 *S46S   *S21S
       S31S  =  1.00000E+00 *S22S   *S23S
       S37S  =  1.00000E+00 *S25S   *S49S
       S36S  = -1.00000E+00 *S28S   *S51S
       S15S  =  1.00000E+00 * S46S
     *         +1.00000E+00 * S10S
       S45S  =  1.00000E+00 * S11S
     *         +1.00000E+00 * S12S
       S44S  =  1.00000E+00 * S30S
     *         +1.00000E+00 * S31S
       S38S  =  1.00000E+00 * S37S
     *         +1.00000E+00 * S36S
       S34S  =  1.00000E+00 *S24S   *S44S
       S33S  =  1.00000E+00 *S44S   *S54S
       S39S  =  1.00000E+00 *S55S   *S38S
       E5Q   =  C5          * Q5Q
       F5M   = (1./ I5         ) * P5M
       E3Q   =  C3          * Q3Q
       F2M   = (1./ I2         ) * P2M
       F3J   = (1./ I3         ) * P3J
       F1M   = (1./ I1         ) * P1M
       E1Q   =  C1          * Q1Q
       E2Q   =  C2          * Q2Q
       F4M   = (1./ I4         ) * P4M
       E4Q   =  C4          * Q4Q
       F1R1  =  1.00000E+00 *MTF1   *F3J
       F2R1  =  1.00000E+00 *MTF2   *F3J
       F3R1  =  1.00000E+00 *MTF3   *F3J
       S16S  =  1.00000E+00 *S48S   *S45S
       S17S  =  1.00000E+00 *S9S    *S45S
       S40S  =  1.00000E+00 *S29S   *S38S
       S18S  =  1.00000E+00 * S16S
     *         +1.00000E+00 * S17S
```

```
  S19S  =  1.00000E+00 * S15S
*         +1.00000E+00 * S18S
  S35S  =  1.00000E+00 * S34S
*         +1.00000E+00 * S33S
  S41S  =  1.00000E+00 * S40S
*         +1.00000E+00 * S39S
  S42S  =  1.00000E+00 * S35S
*         +1.00000E+00 * S41S
  F4VS  =  1.00000E+00 *S57S    *S19S
  F5VS  =  1.00000E+00 *S58S    *S42S
  F5V   =  1.00000E+00 * F5M
*         -1.00000E+00 * F5VS
  E5F   =  R5          * F5V
  F3V   = -1.00000E+00 * F3R1
*         +1.00000E+00 * F2M
*         -1.00000E+00 * F5M
  E3F   =  R3          * F3V
  F1V   =  1.00000E+00 * F1R1
*         -1.00000E+00 * F1M
*         +1.00000E+00 * F2M
  E1F1  =  R1          * F1V
  F2V   =  1.00000E+00 * F2R1
*         +1.00000E+00 * F2M
*         -1.00000E+00 * F4M
  E2F   =  R2          * F2V
  F4V   =  1.00000E+00 * F4M
*         -1.00000E+00 * F4VS
  E4F   =  R4          * F4V
  E5V   =  1.00000E+00 * E5Q
*         +1.00000E+00 * E5F
  E3V   =  1.00000E+00 * E3Q
*         +1.00000E+00 * E3F
  E1V   =  1.00000E+00 * E1Q
*         +1.00000E+00 * E1F1
  E2V   =  1.00000E+00 * E2Q
*         +1.00000E+00 * E2F
  E4V   =  1.00000E+00 * E4Q
*         +1.00000E+00 * E4F
  E1R2  =  1.00000E+00 *MTF1    *E1V
  E2R2  =  1.00000E+00 *MTF2    *E2V
  E3R2  =  1.00000E+00 *MTF3    *E3V
  S43S  = S38S
  S56S  = S38S
  S53S  = S44S
  S32S  = S44S
  S13S  = S45S
  S14S  = S45S
  S1S   = S46S
  S20S  = S46S
  S3S   = S48S
  S4S   = S48S
  S5S   = S49S
  S50S  = S49S
  S7S   = S51S
  S52S  = S51S
  E5VM  = E5V
  E5VS  = E5V
```

```
        F5Q    = F5V
        F5F    = F5V
        E5M    = -1.00000E+00 * E5W
     *           +1.00000E+00 * E3V
     *           -1.00000E+00 * E5V
        F5W    = F5M
        F3VS   = F5M
        F5VM   = F5M
        E3R1   = E3V
        E3T    = E3V
        E3VS   = E3V
        F3Q    = F3V
        F3F    = F3V
        E2M    = -1.00000E+00 * E2W
     *           -1.00000E+00 * E2V
     *           -1.00000E+00 * E3V
     *           -1.00000E+00 * E1V
        F2W    = F2M
        F2T    = F2M
        F3T    = F2M
        F1T    = F2M
        F3R2   = F3J
        F2R2   = F3J
        E3J    =  1.00000E+00 * E3R2
     *           -1.00000E+00 * E2R2
     *           -1.00000E+00 * E1R2
        F1R2   = F3J
        E1R1   = E1V
        E1VM   = E1V
        E1T    = E1V
        F1VM   = F1M
        E1M    =  1.00000E+00 * E1V
     *           -1.00000E+00 * E1W
        F1W    = F1M
        F1Q    = F1V
        F1F1   = F1V
        E2R1   = E2V
        E2T    = E2V
        E2VS   = E2V
        F2Q    = F2V
        F2F    = F2V
        E4M    = -1.00000E+00 * E4W
     *           +1.00000E+00 * E2V
     *           -1.00000E+00 * E4V
        F4W    = F4M
        F2VS   = F4M
        F4VM   = F4M
        E4VM   = E4V
        E4VS   = E4V
        F4Q    = F4V
        F4F    = F4V
C
      ENDIF
```

```
C
C---- Store the state derivatives XDOT...
C
      IF (STATE) THEN
       XDOT( 1) = F5V
       XDOT( 2) = E5M
       XDOT( 3) = F3V
       XDOT( 4) = E2M
       XDOT( 5) = E3J
       XDOT( 6) = E1M
       XDOT( 7) = F1V
       XDOT( 8) = F2V
       XDOT( 9) = E4M
       XDOT(10) = F4V
      ENDIF
C
C---- Store the output list Y...
C
      IF (OUTPUT) THEN
       Y( 1) = Q1Q
       Y( 2) = E1M
       Y( 3) = Q2Q
       Y( 4) = Q3Q
       Y( 5) = Q4Q
       Y( 6) = Q5Q
      ENDIF
C
      RETURN
      END
C
C*****************************************************************
```

DATA FILE

USR01.RPD

GENERATED FROM

ENPORT

```
// MATRIXx RP-name exec file written by ENPORT
//
// File name: USER01.RPD
//
// Assignment of U and NP variables for RP use
C1            =  1.20000E+03
C2            =  3.60000E+03
C3            =  3.60000E+03
R1            =  1.20000E+02
R2            =  3.00000E+02
R3            =  3.00000E+02
//
RP(  1) = C1
RP(  2) = C2
RP(  3) = C3
RP(  4) = R1
RP(  5) = R2
RP(  6) = R3
// End of the file.
```

DATA FILE

USR01.DAT

GENERATED FROM

ENPORT

```
// Matrix-x MATSETUP file written by ENPORT.
//
// File name: USER01.DAT
//
//
// Create a superblock ENPSUP
BUILD
EDIT SUPER-BLOCK
ENPSUP
0
//
// Define user_code_block
DEFINE BLOCK
2
USER CODE BLOCK
BLOCK NAME
ENPUSR
NUMBER OF INPUTS
 29
NUMBER OF OUTPUTS
  6
NUMBER OF STATES
 10
PARAMETER ENTRY
1
Y
   6,  0
// RP Parameters
   1.2000E+03
   3.6000E+03
   3.6000E+03
   1.2000E+02
   3.0000E+02
   3.0000E+02
N
// Initial conditions
   0.0000E+00
   0.0000E+00
   0.0000E+00
   0.0000E+00
   0.0000E+00
   0.0000E+00
   0.0000E+00
   0.0000E+00
   0.0000E+00
   0.0000E+00
//
// Define connections
CONNECT BLOCKS
EXTERNAL OUTPUT
  6
  2
Y
//
// Return to main menu
TOP
//
// End-of-file
```

G.4 Optimization Datafiles

These datafiles were produced by OPTDES. They describe the optimization problem set up for each problem solved in the application example.

DATA FILE

FINAL1.DAT

GENERATED FROM

OPTDES

```
AV
  1200.000000           0
AV
  3600.000000           0
AV
  3600.000000           0
AV
  120.0000000           0
AV
  300.0000000           0
AV
  300.0000000           0
AF
 0.1722275019           4    con1
AF
  249.4954987           4    con2
AF
 0.6293675303           4    con3
AF
 0.3334497809           4    con4
AF
 0.1466183811           4    con5
AF
 0.8452641964E-01       4    con6
END
NDV =      6   NROWV =      6   NDF =      6   NROWF =      6
  DV#      #MP              MINIMUM              MAXIMUM
    1        1   P       600.0000000          6000.000000
    2        1   P       2400.000000          12000.00000
    3        1   P       2400.000000          12000.00000
    4        1   P       24.00000000          600.0000000
    5        1   P       60.00000000          960.0000000
    6        1   P       60.00000000          960.0000000
  ROW      AV#         COEFF
    1        1      1.000000000
    2        2      1.000000000
    3        3      1.000000000
    4        4      1.000000000
    5        5      1.000000000
    6        6      1.000000000
  DF#      #MP             ALLOWABLE              RANGE
    1        1   <P     0.1670000000         0.1670000000
    2        1   vP     0.0000000000E+00      100.0000000
    3        1   <P     0.4170000000         0.4170000000
    4        1   <P     0.4170000000         0.4170000000
    5        1   <P     0.1670000000         0.1670000000
    6        1   <P     0.1670000000         0.1670000000
  ROW      AF#         COEFF
    1        1      1.000000000
    2        2      1.000000000
    3        3      1.000000000
    4        4      1.000000000
    5        5      1.000000000
    6        6      1.000000000
NCM =      0   NROWC =      0
```

DATA FILE

FINAL2.DAT

GENERATED FROM

OPTDES

```
AV
  1200.000000           11   seat spring
AV
  3600.000000           12   front spring
AV
  3600.000000           11   rear spring
AV
  120.0000000           11   seat damper
AV
  300.0000000           12   front damper
AV
  300.0000000           11   rear damper
AV
  500.0000000            7   opj fcn
AF
 0.1075993851           17   p1 seat sprg disp
AF
 -348.1452789           13   p1 seat force
AF
 0.3541334569           16   p1 frt sprg disp
AF
 0.1583947688           17   p1 rear sprg disp
AF
 0.7996477932E-01       17   p1 frt tire compr
AF
 0.3714266419E-01       18   p1 rear tire compr
AF
 0.4795477167E-01       17   p2 seat sprg disp
AF
 -359.1307068           13   p2 seat force
AF
 0.2524786294           16   p2 frt sprg disp
AF
 0.2371307760           17   p2 rear sprg disp
AF
 0.1370741427           17   p2 frt tire compr
AF
 0.1428910345           18   p2 rear tire compr
AF
  250000.0000           18   objective function
END
NDV =      7   NROWV =      7   NDF =     13   NROWF =     13
  DV#      #MP                 MINIMUM              MAXIMUM
    1        1    P          600.0000000          6000.000000
    2        1    P          2400.000000          12000.00000
    3        1    P          2400.000000          12000.00000
    4        1    P          24.00000000          600.0000000
    5        1    P          60.00000000          960.0000000
    6        1    P          60.00000000          960.0000000
    7        1    P       0.0000000000E+00        500.0000000
```

ROW	AV#	COEFF
1	1	1.000000000
2	2	1.000000000
3	3	1.000000000
4	4	1.000000000
5	5	1.000000000
6	6	1.000000000
7	7	1.000000000

DF#	#MP		ALLOWABLE	RANGE
1	1	<P	0.1670000000	0.1670000000
2	1	<P	0.0000000000E+00	1.000000000
3	1	<P	0.4170000000	0.4170000000
4	1	<P	0.4170000000	0.4170000000
5	1	<P	0.1670000000	0.1670000000
6	1	<P	0.1670000000	0.1670000000
7	1	<P	0.1670000000	0.1670000000
8	1	<P	0.0000000000E+00	1.000000000
9	1	<P	0.4170000000	0.4170000000
10	1	<P	0.4170000000	0.4170000000
11	1	<P	0.1670000000	0.1670000000
12	1	<P	0.1670000000	0.1670000000
13	1	vP	0.0000000000E+00	1.000000000

ROW	AF#	COEFF
1	1	1.000000000
2	2	1.000000000
3	3	1.000000000
4	4	1.000000000
5	5	1.000000000
6	6	1.000000000
7	7	1.000000000
8	8	1.000000000
9	9	1.000000000
10	10	1.000000000
11	11	1.000000000
12	12	1.000000000
13	13	1.000000000

NCM = 0 NROWC = 0

DATA FILE

FINAL3.DAT

GENERATED FROM

OPTDES

```
AV
  1200.000000            0
AV
  3600.000000            0
AV
  3600.000000            0
AV
  120.0000000            0
AV
  300.0000000            0
AV
  300.0000000            0
AF
 0.1722275019            4    con1
AF
  249.4954987            4    con2
AF
 0.6293675303            4    con3
AF
 0.3334497809            4    con4
AF
 0.1466183811            4    con5
AF
 0.8452641964E-01        4    con6
AF
 0.1075993851            4    con7
AF
  151.8547211            7    obj fcn
AF
 0.3541334569            4    con8
AF
 0.1583947688            4    con9
AF
 0.7996477932E-01        5    con10
AF
 0.3714266419E-01        5    con11
END
NDV =      6   NROWV =      6   NDF =     12   NROWF =     12
  DV#      #MP              MINIMUM              MAXIMUM
    1        1   P       600.0000000          6000.000000
    2        1   P       2400.000000          12000.00000
    3        1   P       2400.000000          12000.00000
    4        1   P       24.00000000          600.0000000
    5        1   P       60.00000000          960.0000000
    6        1   P       60.00000000          960.0000000
  ROW      AV#          COEFF
    1        1      1.000000000
    2        2      1.000000000
    3        3      1.000000000
    4        4      1.000000000
    5        5      1.000000000
    6        6      1.000000000
```

DF#	#MP		ALLOWABLE	RANGE
1	1	<P	0.1670000000	0.1670000000
2	1	<P	500.0000000	500.0000000
3	1	<P	0.4170000000	0.4170000000
4	1	<P	0.4170000000	0.4170000000
5	1	<P	0.1670000000	0.1670000000
6	1	<P	0.1670000000	0.1670000000
7	1	<P	0.1670000000	0.1670000000
8	1	vP	0.0000000000E+00	100.0000000
9	1	<P	0.4170000000	0.4170000000
10	1	<P	0.4170000000	0.4170000000
11	1	<P	0.1670000000	0.1670000000
12	1	<P	0.1670000000	0.1670000000

ROW	AF#	COEFF
1	1	1.000000000
2	2	1.000000000
3	3	1.000000000
4	4	1.000000000
5	5	1.000000000
6	6	1.000000000
7	7	1.000000000
8	8	1.000000000
9	9	1.000000000
10	10	1.000000000
11	11	1.000000000
12	12	1.000000000

NCM = 0 NROWC = 0

G.5 ANAPRE Data Entry List

In order to execute the optimization problems the following information is required.

Table G.5.1
ANAPRE Data Entry List

VARIABLE	PROBLEM 1	PROBLEM 1	PROBLEM 1
NX	10	10	10
NY	6	6	6
NU	29	29	29
NFF	1	2	2
DATAFILE	PROF1.DAT	PROF2.DAT	PROF1.DAT
	---	PROF3.DAT	PROF3.DAT
T_O	0	0	0
T_F	4.0	4.0	4.0
NO. INTG STEPS	1000	1000	1000
INITIAL CONDS.	N	N	N
PMM	X	X	X
OBJ ECTIVE SET SIZE	1	2	1
AV OBJECTIVE NO.	---	7	---
PM NO.	---	2	---
OBJ. FUNCTION AF NO.	2	13	8
NO OF AV FCNS	6	7	6
NO. OF AF FCNS.	6	13	12
AV(1)	1200	1200	1200
AV(2)	3600	3600	3600
AV(3)	3600	3600	3600
AV(4)	120	120	120
AV(5)	300	300	300
AV(6)	300	300	300
AV(7)	---	500	---

G.6 Signal Generator Bond Graph

The following bond graph was used to produce the velocity input to the wheels of the suspension system.

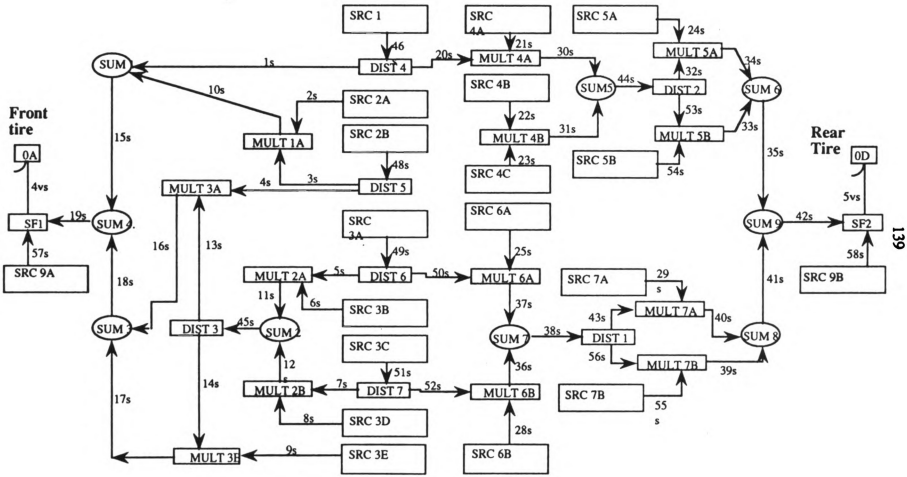


FIGURE G.6.1
Signal Generator Bond Graph

LIST OF REFERENCES

[1] Rosenberg, R., *The ENPORT Reference Manual*, 7th Edition, Rosencode Associates, Lansing Mi., 1990.

[2] Balling, R., and Free, and J., Parkinson, A., *OPTDES.BYU User's Manual, Release 4.0*, Design Synthesis, INC., Provo, UT, 1989.

[3] Haug, E.J., and Arora, J.S., *Applied Optimal Design, Mechanical and Structural Systems*, John Wiley & Sons, 1979.