To the Graduate Council:

I am submitting herewith a thesis written by Stephen Douglas Howard entitled "Development Of A Hypercard-Based Graphical Users Interface Program For Implementing Liquid Rocket And Test Facility Component Performance Models". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Aerospace Engineering.

Dr. Roy J. Schulz, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Stephen M. Howard

Date November 3, 1993

DEVELOPMENT OF A HYPERCARD-BASED GRAPHICAL USERS INTERFACE PROGRAM FOR IMPLEMENTING LIQUID ROCKET AND TEST FACILITY COMPONENT PERFORMANCE MODELS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Stephen Douglas Howard

December, 1993

ACKNOWLEDGMENTS

This thesis was made possible through NASA grant (NAG13-20), entitled "Technology Development of Thermal Property Data Subroutine for Liquid Hydrogen and Liquid Oxygen". I want to express my thanks to the NASA project manager Dr. William St. Cyr of the Stennis Space Center, Stennis, Mississippi, for his support and patience throughout this project.

I would like to express my thanks and appreciation to Dr. Roy Schulz for being my major professor and providing me with guidance in every phase of this project; and to Dr. C.F. Lo and Dr. Roger Crawford for their advice and for agreeing to serve on my thesis committee. I would also like to express my sincere gratitude to Dr. Edwin Gleason for all the help he afforded me during my time at UTSI. Special thanks go to my parents, Sterling and Dorothy Howard, for all their love and support. A special word of thanks goes to Lt. James Clegern for his advice and suggestions. This thesis would have taken much longer to complete without it.

ABSTRACT

This project involved the development of a graphical users interface (GUI) that linked a thermal properties data subroutine to liquid rocket component performance models. Three generic liquid rocket engine components were selected to demonstrate the computer program and its interactive interface. These components were a pump, a valve, and a propellant tank. The program, entitled LRE-CP^2, combines a modified thermophysical properties routine with a graphical user interface (GUI) using shell-based stacks created using Hypercard® software. The user enters a set of required conditions for a given component on a CRT screen in a screen specific format, and the LRE-CP^2 then calculates the component performance and displays it in a tabular format, both on screen and in hard copy, if desired.

Basic theory, and the format of the GUI development is discussed in detail for each component. Sample calculations are included for each component, and comparison of results with independently calculated performance is included where applicable.

In conclusion, LRE-CP^2 provides propulsion engineers with a convenient and easy to use graphical users interface (GUI) that integrates fluid property data into rocket or test facility component performance models. This enables component performance predictions to be obtained that are based on the best and most complete tabulations of thermodynamic state data, for 11 different fluids. Future work recommended is the development of other engine component models. In addition, the integration of the LRE-CP^2 into an expert design system architecture would broaden its usefulness to the rocket engine design community.

Table of Contents

CHAPTER		PAGE
1.	INTRODUCTION	1
	Background	1
	Description of the LRE-CP^2	2
2.	THERMODYNAMIC PROPERTY SUBROUTINE	5
	Discussion of the Three Candidates	5
	MIPROPS	6
	Modifications to the MIPROPS Program	7
3.	INTRODUCTION TO THE GRAPHICAL SHELL	15
4.	COMPONENT PERFORMANCE MODELS	17
	LRE-CP^2 Pump Performance Program	17
	Background Theory	17
	Development of the Pump Performance Program	22
	Sample Calculations	38
	LRE-CP^2 Valve Performance Program	44
	Background Theory	44
	Development of the Valve Performance Program	46
	Sample Calculations	58
	LRE-CP^2 Propellant Tank Performance Program	60
	Background Theory	60
	Development of the Propellant Tank Performance Program	62
	Sample Calculations	71
5.	CONCLUSIONS AND RECOMMENDATIONS	74
BIBLIOGRAPHY		78

APPENDICES 81
APPENDIX A. Modified MIPROPS Fortran 77 Code 82
APPENDIX B. Run Program Scripts 112
VITA 128

List of Tables

Table		Page
1.	Valid Pressure and Temperature Ranges for MIPROPS ...	8
2.	Estimated Uncertainties of MIPROPS Properties	8
3.	Critical and Triple Points for Hydrogen, Nitrogen, and Oxygen	14
4.	Recommended Impeller Shapes for Ranges of N_s	20
5.	A-1 Stage Engine Calculation Results	44

List of Figures

Figure		Page
1.	LRE-CP² Main Menu Screen	4
2.	Graphical Representation of the Newton-Raphson Method for $f(x) = x^2 - 2$.	11
3.	Hydrogen T-h diagram showing discontinuity in liquid-vapor region.	12
4.	Hydrogen T-s diagram showing discontinuity in liquid-vapor region.	13
5.	PPP Introductory Card	24
6.	Selection Screen 1 and help card	25
7.	Selection Screen 2 and help card	26
8.	Selection Screen 3 and help card	28
9.	Pump Data Entry and help card	29
10.	Example of a filled Hypercard background field	30
11.	Pump Computations and help card	32
12.	Pump Inlet Flow Conditions and help card	34
13.	Pump Outlet Flow Conditions and help card	36
14.	Properties of Pump and help card	39
15.	PPP End Card	40
16.	PPP Data Entry and Pump Properties for A-1 Stage Engine	42
17.	PPP Inlet and Outlet Flow Conditions for A-1 Stage Engine	43
18.	VPP Introductory Card	48
19.	Valve Selection Screen and help card	49
20.	Valve Computations and help card	50
21.	Valve Flow Conditions and help card	54

Figure		Page
22.	Valve Properties and help card	56
23.	Examples of performance curves for a butterfly valve.	57
24.	VPP End Card	59
25.	VPP Data Entry and Results for A-1 Stage Engine	61
26.	PTPP Introductory Card	63
27.	Propellant Tank Selection Screen and help card	64
28.	Propellant Tank Computations and help card	67
29.	Propellant Tank Data Screen (Case #1) and help card	69
30.	Propellant Tank Data Screen (Cases #2 and #3) and help card	70
31.	PTPP End Card	72
32.	PTPP Data Entry and Results for Sample Calculation	73
33.	Example of a Flow Schematic for an EDS - LRE-CP^2 System	76

List of Symbols

Symbol	Definition	Units
α	volume fraction of liquid in tank	
$\dot{Q}$	heat transfer rate	Btu/sec
Q	volumetric flow rate	ft^3/sec
$\dot{m}$	mass flow rate	lbm/sec
$\dot{W}$	work rate or power	Btu/sec, hp
C_d	valve flow coefficient	
C_p	specific heat (constant pressure)	Btu/lbm °R
C_v	specific heat (constant volume)	Btu/lbm °R
C_v	flow coefficient	
d	diameter	inches
E	total energy	Btu
e	total energy per unit mass	Btu/lbm
fhp	pump flow horsepower	horsepower
g	acceleration of gravity	ft/s^2
γ	specific heat ratio, C_p/C_v	
η	efficiency	
h	enthalpy	Btu/lbm
h_o	stagnation enthalphy	Btu/lbm
H_2	hydrogen	
H_{elev}	elevation head	feet
H_f	friction head	feet
K	Kelvin	Kelvin
K_v	valve loss coefficient	
L	characteristic length	feet
N	pump or impeller rotational speed	rpm
N_s	pump specific speed	
N_2	nitrogen	
O_2	oxygen	
P	pressure	psia
θ_v	valve opening angle	degrees
ρ	density	lbm/ft^3
rpm	revolutions per minute	

Symbol	Definition	Units
s	entropy	Btu/(lbm °R)
T	temperature	°R, K
u	internal energy	Btu/lbm
V	volume	ft^3
$\dot{V}$	volumetric flow rate	gal/min
f	friction coefficient	
ΔH	pump developed head	feet

Subscripts

i - initial

f - final

CHAPTER 1

INTRODUCTION

Background

In the research, design, and testing of rocket engines, knowledge of component performance capabilities is extremely important. To make performance calculations, propulsion engineers need access to thermodynamic state data for propellant and other fluids to reasonably predict system performance. There are a number of thermodynamic property programs in existence that can determine the thermal state of a fluid (propellant) given knowledge of two thermodynamic or state properties (such as pressure and temperature). For liquid rocket propellants, it is necessary to have the widest range of thermal data possible because, in the engine components, pressures can range up to thousands of pounds per square inch, and temperatures can range from cryogenic values all the way up to thousands of degrees Rankine. The need for a convenient way of integrating thermal property data into component performance models was the driving force behind the development of the Liquid Rocket Engine Component Performance Program (LRE-CP^2), which is the subject of this thesis. The objective of this project is to provide the propulsion engineer with fluid thermal data and component performance parameters in a single, integrated organized structure. Instead of having to sort through assorted fluid tables and performance equations, or have to develop thermal property data files or curve fits, the engineer can now

utilize existing data bases for thermal properties in component performance models.

In designing the LRE-CP^2 it was judged that, at a minimum, it should meet the following objectives:

1) Contain the performance models in a readily accessible computer format, preferably with a graphical shell having a user-friendly graphical users interface, that is quick and simple to use;

2) Occupy a minimal amount of computer memory so that it can be transported on a single diskette and work on virtually any size machine (the existing program was specifically developed for a Macintosh II computer using a 14" monitor).

3) Include property data for liquid hydrogen, liquid oxygen, and liquid nitrogen, as well as a reasonable number of other fluids that are associated with propulsion systems.

4) Have pressure and temperature as typical, or usual inputs.

Description of the LRE-CP^2

The LRE-CP^2 is a graphical shell program that runs on an Apple Macintosh™ computer (with at least a 14 inch screen). It represents a union between two separate applications: a modified thermophysical property program written in Fortran (called MIPROPS (McCarty,1986)); and individually designed rocket engine component performance subroutines, called *stacks*, made using Hypercard® software, Version 2.1 (Claris Corp.,1990). The scheme of the LRE-CP^2 is such that a user never sees the property routines when running. For a given component, the user enters the necessary inputs, runs the performance calculation program, and receives the performance prediction, or output, in an organized structure.

All calculations done in the LRE-CP^2 are intentionally kept in the background to minimize the amount of detail for the user.

The LRE-CP^2 runs under the Hypercard software; therefore, Hypercard needs to be on the hard drive to run it. To implement the general scheme for the graphical users interface three typical liquid rocket engine components were selected for implementation of the performance models. These performance models are for a pump, a valve, and a tank. They are denoted the Pump Performance stack, the Valve Performance stack, and the Propellant Tank Performance stack. Figure 1 shows the startup screen for the LRE-CP^2. A user can go to any one of the three stacks mentioned by clicking on the appropriate picture. These "pictures" are Hypercard buttons that act to open the stacks. Pressing the Quit Hypercard button will quit the Hypercard application. The individual stacks or performance models are discussed in detail beginning in Chapter 4.

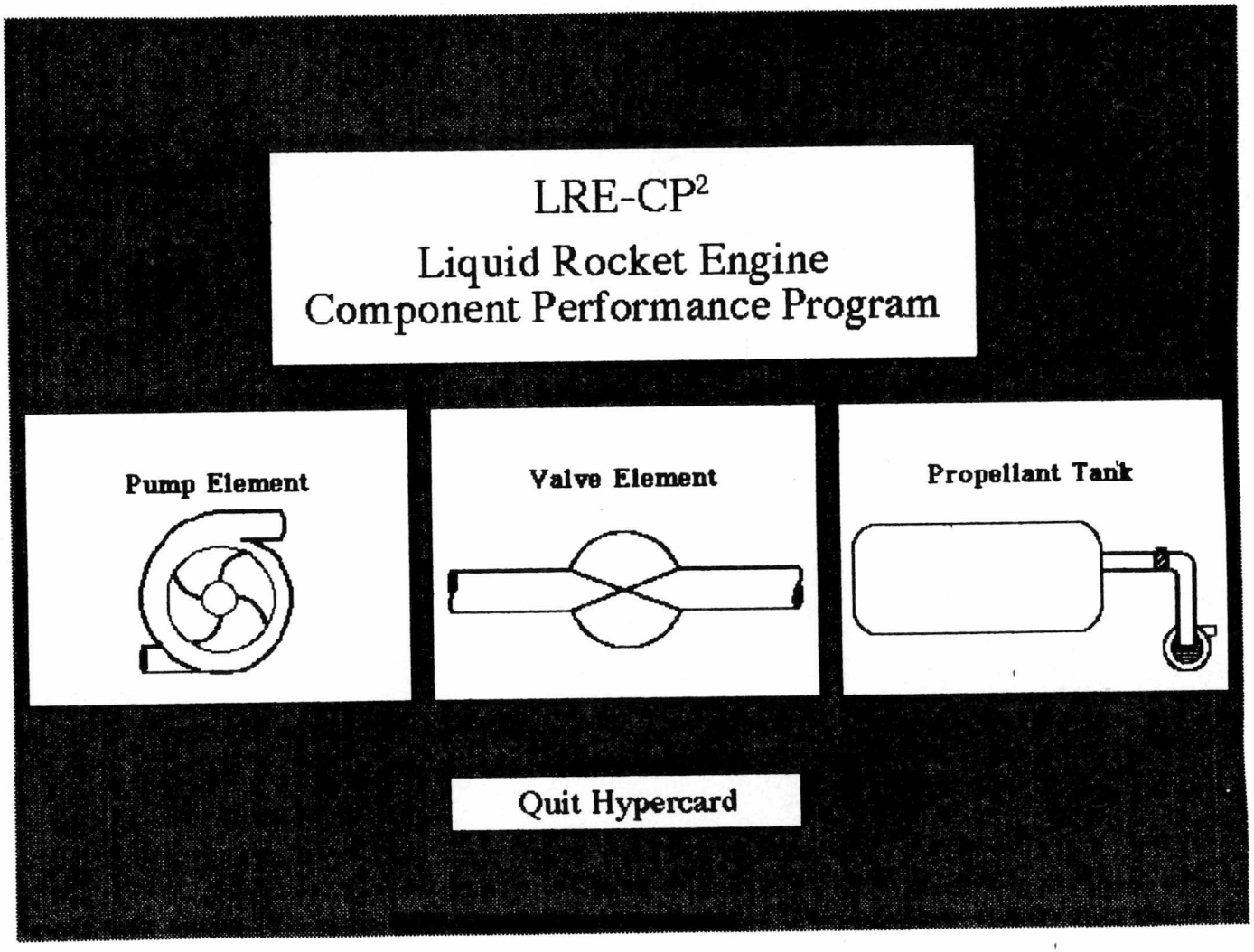


Figure 1. LRE-CP^2 Main Menu Screen

CHAPTER 2

THERMODYNAMIC PROPERTY SUBROUTINE

Discussion of the Three Candidates

At the beginning of the project three different property programs were under consideration for LRE-CP[2]:

1.) FLUID, a generalized property program from NASA Lewis Research Center (Fessler,1977),

2.) Digital Transient Model (DTM), subroutines developed for the Space Shuttle main engine (Spinn,1986),

3.) MIPROPS, a thermophysical property program developed at the National Bureau of Standards (McCarty,1986).

The criteria which drove the selection process for a thermal property routine was twofold. One, the routine must encompass the widest range of pressures and temperatures possible (including cryogenic temperatures); and two, the routine must be relatively easy to modify for incorporation and operation within a graphics-based architecture.

All three programs satisfactorily met the range requirement. Both MIPROPS and DTM have the same pressure and temperature limits, for example $\mathbf{H_2}$ from 0 to 17,404 psia, and 24.8 to 720 °R, and $\mathbf{O_2}$ from 0 to 17,404 psia, and from 97.85 to 720 °R. The range limits of FLUID are based on user supplied gas tables, which are not included with the program.

The second requirement turned out to be the deciding factor in making a selection. Both DTM and FLUID appeared to be poor choices in terms of adaptability and usefulness.

The DTM is composed of several different subroutines that call the accompanying hydrogen and oxygen data tables. These data tables are extensive, encompassing ranges greater than those to which the DTM interpolation subroutine (CPMRO2) is restricted. They occupy a large amount of memory, and would therefore make the overall size of the LRE-CP^2 unfavorable. In addition, none of the DTM subroutines for H_2 and O_2 have an option with pressure and temperature as the inputs.

The main difficulty with the FLUID program is that the individual fluid tables are not included with it.

With these factors in mind, MIPROPS appeared to be the best alternative. The program occupies a relatively small amount of memory (120 Kb) and contains data tables for 11 different fluids. These fluids are hydrogen, oxygen, nitrogen, argon, nitrogen trifluoride, methane, ethane, ethylene, propane, isobutane, and normal butane.

MIPROPS

MIPROPS is a computer program that calculates the thermodynamic properties of 11 different fluids. It uses an empirical model for the equation of state (McCarty,1986), which in theory is more accurate than the generalized one used in FLUID. For a given fluid, the user selects two input variables and enters data for them. From these inputs, MIPROPS determines the complete thermodynamic state.

MIPROPS is valid over a wide range of pressures and temperatures for each fluid. In certain areas on the thermodynamic surface of a fluid, MIPROPS cannot always be expected to give an accurate answer. These areas are identified as points near the critical point, saturation boundaries,

and phase change boundaries (McCarty,1986). Tables 1 and 2 show the acceptable pressure and temperature ranges and estimated uncertainties of MIPROPS for hydrogen, nitrogen, and oxygen.

Modifications to the MIPROPS Program

The Fortran code for MIPROPS, as written, was not suitable for integration into a graphics based operational shell. Some changes needed to be made. To make these adjustments, the code was transferred to the MPW Shell™ (Apple Computer,1991) from MacFortran™ (Absoft Corp.,1986). It was necessary to restructure the tabulation of the fluid data in making the shift to MPW Shell because MPW did not recognize the semicolon ";" that separated the data values.

The first change involved removing all the extraneous program statements at the beginning of the MIPROPS code. This freed space and minimized the amount of required memory (one of the primary goals of the LRE-CP[2]). These unnecessary statements included:

- **CHARACTER** statements for fluid properties and units
- all of the **WRITE** statements for printing program name, development information, and entry information.

The statement asking whether the input is coming from a file or the keyboard was changed so that MIPROPS would always read from a file. Also, an OPEN statement was added so MIPROPS system messages would be sent to a data file labeled "messages", to be available for the user to view. Examples of these messages are for values that are out of range and errors incurred when MIPROPS failed to converge or find a valid

Table 1. Valid Pressure and Temperature Ranges for MIPROPS

Fluid	Pressure Range	Temperature Range
H_2	0-17404.8 psia	24.84-720 °R
N_2	0-145,040 psia	113.67-3420 °R
O_2	0-17404.8 psia	97.85-720 °R

Table 2. Estimated Uncertainties of MIPROPS Properties

	% Uncertainty		
	Liquid below T_c	Gas below T_c	Fluid above T_c
H_2			
P	5.0	0.25	0.20
ρ	0.1	0.25	0.20
T	0.1	0.25	0.20
N_2			
P	5.0	0.30	0.30
ρ	0.5	0.30	0.20
T	0.5	0.30	0.20
O_2			
P	5.0	0.25	0.15
ρ	0.1	0.25	0.15
T	0.1	0.20	0.10

Source: McCarty, Robert D. "Interactive FORTRAN Programs for Micro Computers to Calculate the Thermophysical Properties of Twelve Fluids [MIPROPS]", NBS Technical Note 1097, May 1986.

thermodynamic state. A READ statement and DO loop were added to read the number of data points that the user of LRE-CP[2] entered for the component performance calculations, and then to execute MIPROPS the corresponding number of times.

The second major modification to MIPROPS pertained to the entry options that were added in a later revision to the program by the Sverdrup Technology, Inc., Stennis Space Center group. In the original MIPROPS version, only four entry options were available:

- pressure and temperature
- pressure and density
- temperature and density
- saturation conditions.

In the revised MIPROPS, four additional entry options were added:

- pressure and entropy
- density and internal energy
- pressure and enthalpy
- pressure and internal energy.

For these four new entry options a reference temperature was assigned to run the iterative subroutine required to converge on to the thermodynamic states. This reference temperature T_0 was 459.67 °R.

However, when using any one of the new entry options for low pressures and cryogenic temperatures, MIPROPS would produce an error. It was discovered, after extensive debugging under MPW Shell, that the error was occurring in the MIPROPS subroutine NUITER. NUITER is responsible for the iterations in MIPROPS used to converge on a thermodynamic state. The iterative technique it uses is based upon the commonly applied Newton-Raphson method for finding roots.

Graphically, the Newton-Raphson method is shown in Figure 2. From an initial guess of x_n, a tangent is drawn from $f(x_n)$ to a new abscissa location x_{n+1}. At this location a new tangent is drawn from $f(x_{n+1})$ to x_{n+2}. The process is repeated until $f(x_r)$ is zero. The value of x_r is what is being sought in the process, that is, the method will converge upon the root at x_r. In equation form, the Newton-Raphson method is defined by (Chapra,1988):

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad (2\text{-}1)$$

The Newton-Raphson method works well for most functions. However, there are cases where the method breaks down. Examples of this are multiple roots, roots near inflection points, and zero slope functions (Chapra,1988).

In the cases of the four new entry options for thermodynamic state variables described above, the problem appeared to be due to the assigned reference temperature, $T_0 = 459.67$ °R, that starts the iteration process. As can be seen from Figures 3 and 4 for H_2, a discontinuity exists at the liquid-vapor line for the H_2 fluid. This discontinuity prevents NUITER from converging to the correct value for fluids in the liquid region. Changing the reference temperature T_0 alleviated this problem. Instead of using an arbitrary value of $T_0 = 459.67$ °R, a separate value for the initial guess temperature was defined for each region. These values were based on the critical and triple points of each fluid, and were taken from data in the Journal of Physical and Chemical Reference Data (Younglove,1982). Specifically, if the input pressure is below P_c, then the

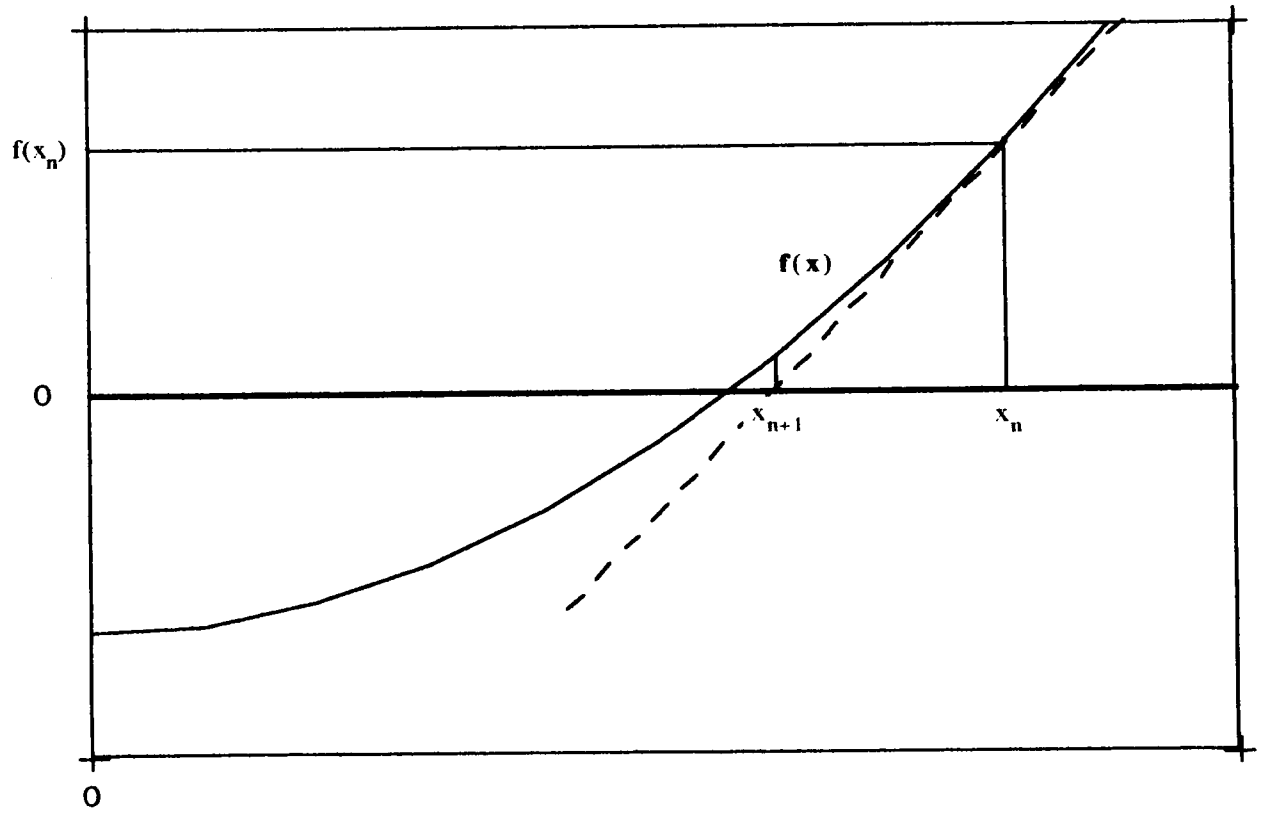


Figure 2. Graphical Representation of the Newton-Raphson Method for $f(x) = x^2 - 2$.

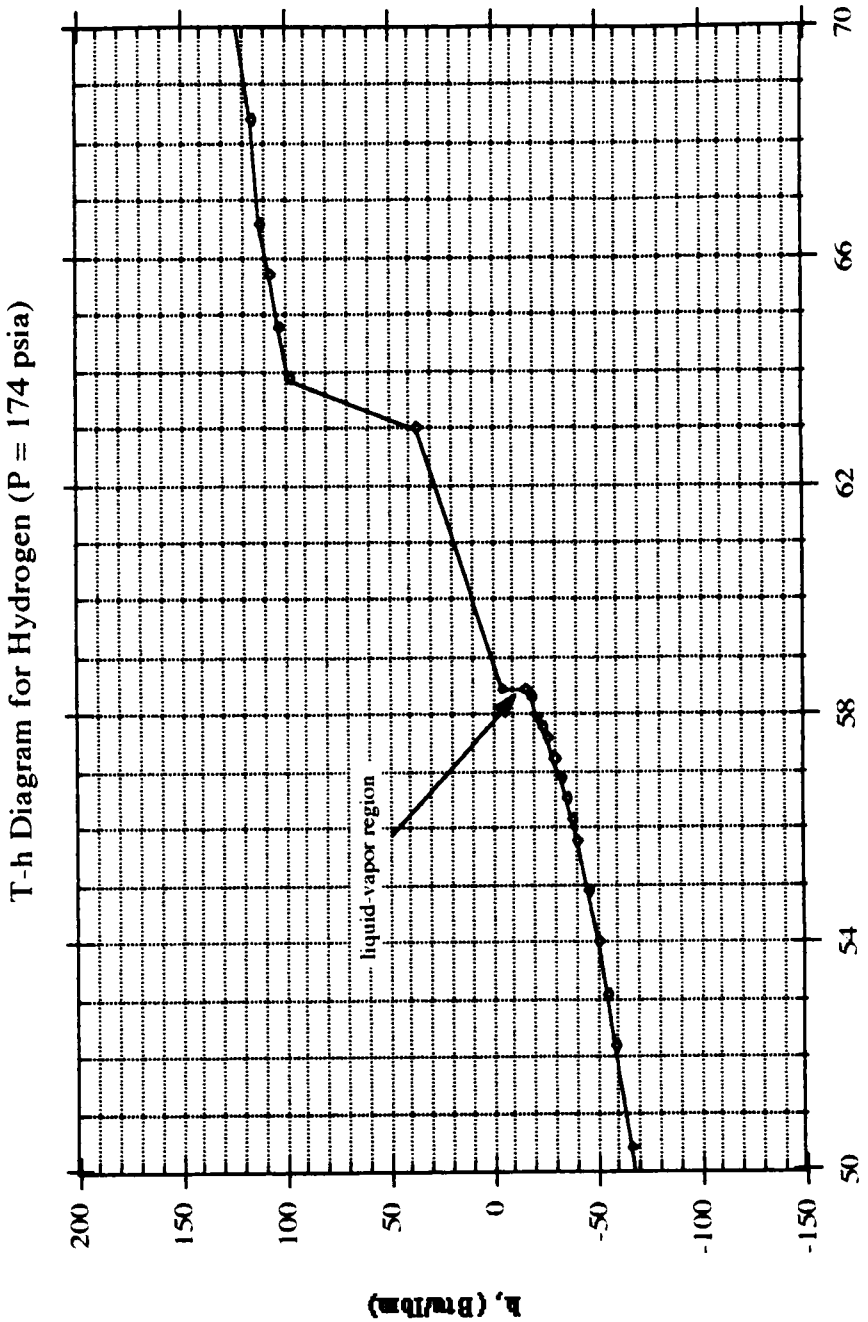


Figure 3. Hydrogen T-h diagram showing discontinuity in liquid-vapor region.

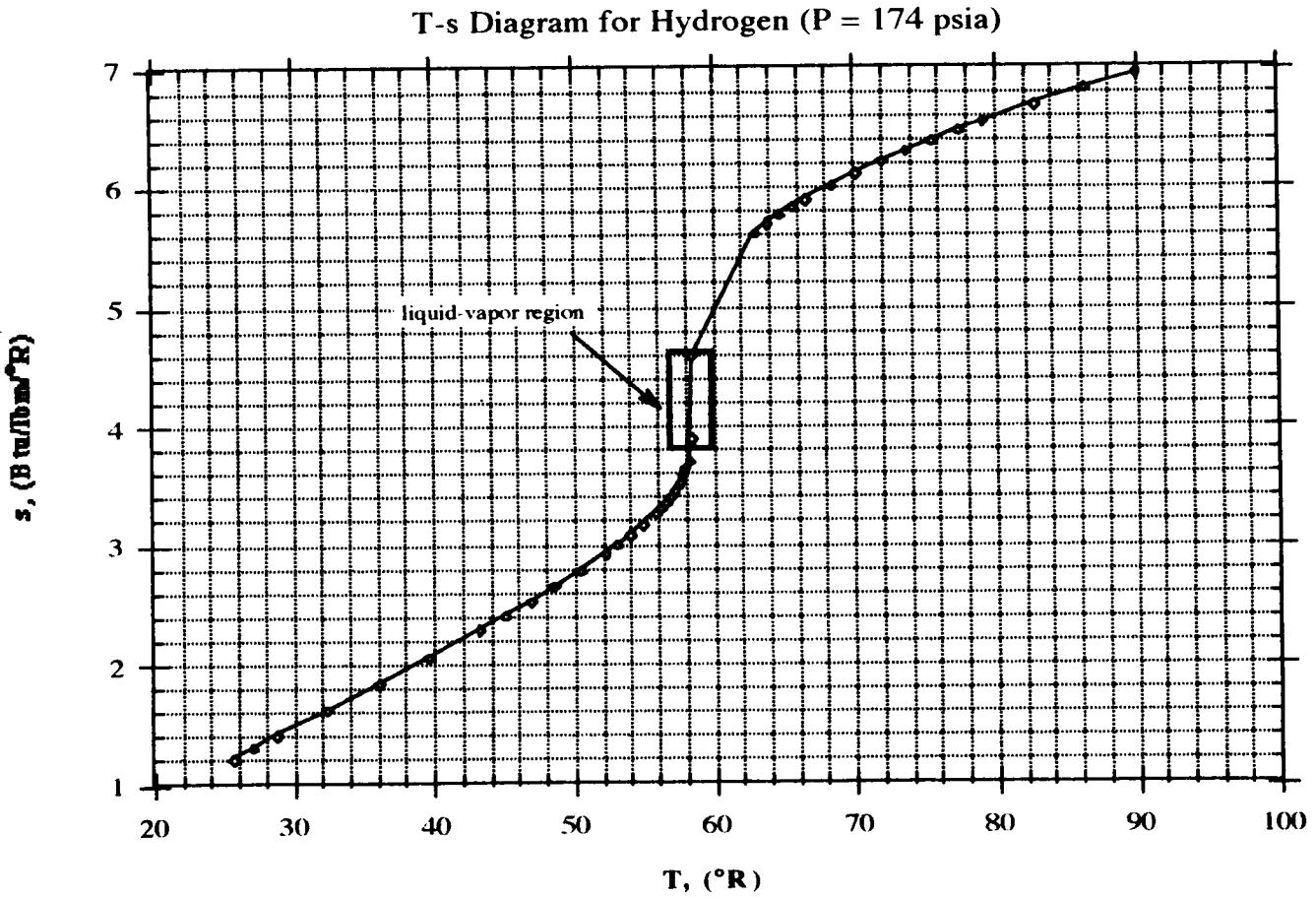


Figure 4. Hydrogen T-s diagram showing discontinuity in liquid-vapor region.

triple point temperature T_r is used for T_0; if the input pressure lies above P_c, the critical temperature T_c is used. Table 3 displays these values for hydrogen, nitrogen, and oxygen.

After making this change with the reference temperature that begins the Newton-Raphson iteration process for the thermodynamic state determination, MIPROPS ceased executing in an infinite loop when using one of the newer entry options for points in the liquid region of a fluid.

Table 3. Critical and Triple Points for Hydrogen, Nitrogen, and Oxygen

Fluid	P_c	T_c	T_t
H_2	186.20	59.29	24.84
N_2	493.003	227.30	113.67
O_2	731.44	278.25	97.85

Source: McCarty, Robert D. "Interactive FORTRAN Programs for Micro Computers to Calculate the Thermophysical Properties of Twelve Fluids [MIPROPS]", NBS Technical Note 1097, May 1986.

A copy of the modified MIPROPS Fortran code (minus the fluid data tables, which were not changed) is listed in Appendix A. MIPROPS was now ready for connection to a graphics based shell.

CHAPTER 3

Introduction to the Graphical Shell

At the beginning of the project it was necessary to choose a software development tool to create the graphical shell for the LRE-CP[2]. It was decided that Macintosh's Hypercard would be the best choice because of its ease of use and broad applicability.

Hypercard is a development program for constructing graphics-based "stacks" for a wide variety of applications. Hypercard "stacks" are custom designed documents composed of *cards* that provide easy interaction and utilization. This interaction is furnished by graphics, buttons, text fields, and Hypertalk. Hypertalk is a hypertext programming language built into Hypercard. The language is used to write "scripts" that allow the different Hypercard elements to communicate with each other. With Hypertalk a person can construct elaborate and multi-functional stacks. An excellent technical example of a Hypercard stack is a design and optimization program for solid rocket motors (Clegern,1993).

An important feature of Hypercard is its ability to work with other applications through Hypertalk scripts. This was critical because of the need to connect the modified MIPROPS with the stacks created in Hypercard. For a complete discourse on Hypercard and its capabilities, refer to the Hypercard Reference Manual and the Hypercard Script Reference Guide (Claris Corp.,1990). These two books come with the *Hypercard Development Kit* from the Claris Corporation, which is available in most computer warehouse stores. It is beyond the scope of this thesis to review the Hypercard shell and the various capabilities it has to

provide interactive graphical user interfaces. Hypercard was used in the present thesis to link the thermal property program MIPROPS, in its modified form, to property component performance models for a pump, valve and propellant tank. Hypercard was also used to create CRT screens that served as the graphical input format for the performance conditions desired from the performance models. Hypercard also provided the graphical output CRT screens, and the corresponding data files that resulted from the component performance calculations.

CHAPTER 4

COMPONENT PERFORMANCE MODELS

LRE-CP^2 Pump Performance Program

The pump performance program is one of the three rocket engine component "stacks" contained in the LRE-CP^2. The purposes of this chapter are to discuss the basic theory and equations underlying the Pump Performance Program (PPP) and to describe how the pump program was developed.

Background Theory

The component performance characteristics to be determined by the PPP are:

- the *complete thermodynamic state* of the fluid at the inlet and outlet stages of the pump
- the *pump developed head*, ΔH
- the *flow horsepower* produced by the pump, fhp
- the *net positive suction head* for the pump, NPSH
- the *pump specific speed*, N_s

The following is a brief review of these characteristics.

Thermodynamic state: The thermodynamic state of a fluid can be expressed in terms of the following properties

pressure, P
temperature, T
density, ρ
internal energy, u
enthalpy, h
entropy, s

Any two of these properties will determine the equilibrium thermodynamic state of the fluid. These properties are referred to as *point functions*, being independent of any process or path to the point (Black & Hartley,1985).

Pump Developed Head (ΔH): The pump developed head represents the difference between the pump discharge head and pump suction head (Sutton,1986)

$$\Delta P = P_{disch.} - P_{suct.} \quad (psia) \tag{4-1}$$

Pump head is more commonly expressed in units of feet. The following equation converts psia of head to feet of head:

$$\Delta H\ (ft) = \frac{\Delta P(psia) * 144}{\rho * \frac{g_0}{g_c}} \tag{4-2}$$

Pressure rise in a rocket engine pump is normally accomplished with an impeller that accelerates the flow to higher velocity and then channels the flow to the pump diffuser, where velocity head is converted to pressure head through the diffusion process (Sutton,1986). For high compression ratios it is often necessary to have more than one impeller (a multi-stage compressor).

Flow Horsepower (fhp): The flow horsepower represents the amount of power the pump puts into the flow to achieve a given set of conditions, usually the desired pressure ratio across the pump. This work is supplied by the turbine as shaft power. How much of the shaft power is actually utilized by the pump depends on the pump efficiency (discussed later in this chapter).

Net Positive Suction Head (NPSH): This is an important parameter for predicting the safe and efficient operation of a turbopump system. The

pressure of the fluid at the entrance (suction side) to the pump must be maintained at a minimum level to avoid cavitation. Cavitation is indicated by the presence of vapor bubbles or pockets in a liquid flow and occurs when the flow pressure drops below the vapor pressure at the current temperature. Two different definitions of NPSH are usually mentioned: $NPSH_R$, which is the required NPSH; and $NPSH_A$, which is the available NPSH for controlling cavitation. $NPSH_R$ is determined by manufacturers for specific pumps. The LRE-CP[2] computes $NPSH_A$. Any further mention of NPSH will pertain to $NPSH_A$ unless otherwise stated. NPSH is defined as (Walker,1972):

$$NPSH = H_{ab} + H_{elev} - H_f - H_{vp} \qquad (4\text{-}3)$$

where H_{ab} is the absolute tank pressure, H_{elev} is the pressure difference due to height differences between the pump inlet and the tank, H_f is the friction pressure loss of the fluid on its way to the pump, and H_{vp} is the fluid vapor pressure. All quantities are expressed in units of feet.

To prevent cavitation, $NPSH_A$ must be greater than $NPSH_R$ for the pump, with an extra few feet as a safety margin (Walker,1972). Otherwise, a pump may lose its "prime" and fail to sufficiently move the fluid through to the rest of the engine. This could lead to extremely damaging situations. Experimental curves for pump flows show that as the inlet pressure (suction head) is steadily reduced, the pump head will eventually begin to drop off dramatically. This drop-off occurs at the point where $NPSH_A = NPSH_R$ (Sutton,1986).

Pump Specific Speed (N_s): N_s is a dimensionless quantity and a primary indicator of optimum pump design for a distinct set of flow conditions. It is computed as (Sutton,1986)

$$N_s = \frac{21.2 * N * \sqrt{Q}}{(\Delta H)^{0.75}} \qquad (4\text{-}4)$$

where N is pump speed in rpm, Q is volume flow rate in ft^3/sec, and ΔH is pump developed head given by equation (4.2).

Depending on the value of N_s, a certain pump type (impeller shape) is recommended for the required pumping operation. Sutton in his Rocket Propulsion Elements lists four different ranges of N_s along with the ideal impeller contour for each. This information is shown in Table 4.

Table 4. Recommended Impeller Shapes for Ranges of N_s

N_s	Pump Geometry
500 - 1000	radial (centrifugal)
1000 - 2000	Francis type
2000 - 3000	mixed flow
3000 +	axial

Source: Sutton, George P., Rocket Propulsion Elements, Fifth Edition, John Wiley & Sons, New York, 1986.

As a rule, axial flow pumps are employed for high speed flows requiring small pressure rises, while centrifugal pumps are high performance devices used for large pressure rises.

There are a number of important pump parameters necessary for determining the above performance characteristics. They are:

- pump efficiency, η_{pump}
- pressure rise ratio, P_{out}/P_{in}
- flow rate, $\dot{m}_{pump}$
- shaft speed, N

- friction loss, H_f
- elevation head, H_{elev}

Following is a brief review of these parameters.

Efficiency: There are several different ways of representing efficiency for a pump or compressor. There is the compressor efficiency, the compressor polytropic efficiency, and the compressor stage efficiency (Oates,1988). All three have in common the relationship of ideal work to actual work. Specifically,

$$\eta_{pump} = \frac{\text{ideal work interaction}}{\text{actual work interaction}} \tag{4-5}$$

Assuming the pump is adiabatic and the fluid flow across it is calorically perfect (constant specific heats),

$$\eta_{pump} = \frac{\left(\frac{P_{out}}{P_{in}}\right)^{\frac{\gamma - 1}{\gamma}} - 1}{\frac{T_{out}}{T_{in}}} \tag{4-6}$$

For a rocket engine turbopump, η_{pump} can be expressed as (Huzel,1971)

$$\eta_{pump} = \frac{\text{pump flow horsepower}}{\text{brake horsepower from turbine}} \tag{4-7}$$

Pressure Rise Ratio: This is a measure of the pump pressure rise from the inlet to the outlet. If the pump is multi-staged, this ratio is for all the stages combined.

Flow Rate: This quantity is the flow rate circulating through the pump. Assuming steady flow, the equation of continuity for the pump is

$$\dot{m}_{inlet} = \dot{m}_{outlet} = \dot{m}_{pump} \tag{4-8}$$

Shaft Speed: This quantity is a measure of the rotational speed of the pump impeller or compressor. Centrifugal flow pumps tend to have lower shaft speeds than axial flow pumps. Shaft speed is typically expressed in revolutions per minute (rpm).

Friction Loss, (H_f): This system factor expresses the loss in suction head due to friction in transferring the fluid from the tank to the pump. It is used in computing the NPSH for the pump. A typical equation for depicting friction loss is (Tullis,1989),

$$H_f = \frac{fLQ^2}{1.23*gd^5} \tag{4-9}$$

where f is the friction factor, **L** is a representative pipe length, and **d** is pipe diameter. A generalized theoretical method for determining friction loss is virtually impossible. Its value is dependent on a specific system and is best found experimentally. For an ideal system, H_f is considered negligible.

Elevation Head, (H_{elev}): This quantity represents the difference in pressure head due to elevation of the propellant tank above the pump inlet. If the tank lies below the pump, H_{elev} is negative.

Development of the Pump Performance Program

The Pump Performance Program's stack is composed of 21 total cards (11 program cards, 10 help cards). The graphics for the cards were created using MacDraw II and MacPaint software. The pump pictures are based on

centrifugal pump schematics. Each card in the stack has a help card associated with it to give further information if needed. Access to the help cards is attained by clicking on the question mark (?) buttons at the top of the cards.

Pressing the Pump Element button on the LRE-CP^2 main menu screen will take you to the Pump Performance Program. The first card is an introductory card that explains the basics of the stack, and is shown in Figure 5. The Main Menu button at the top of the card will take you back to the LRE-CP^2 main screen. The Quit button will open the last card in the stack where the program session can be ended. To continue to the next card, press the arrow button located at the bottom right.

The second card in the stack is **Selection Screen 1**, shown in Figure 6. On this card the fluid type and entry options for the pump are chosen. The entry option should be selected based on what properties are known at the pump inlet. Both a fluid and an entry option must be selected for the PPP to run. The radio button scripts are designed so that only one fluid and entry option can be chosen at a time (radio buttons are the round buttons located next to the selections). Each item in the lists has a number associated with it that MIPROPS can read. Once the desired selections have been made, the information is sent to a hidden background field for later transmittal to MIPROPS.

The next card is **Selection Screen 2**. This card is shown in Figure 7. This is where values for pump efficiency, pressure rise ratio, and flow rate are entered. The pump efficiency will default to 1.0 if no value is entered. Quantities for these variables must be entered if the pump performance characteristics are to be determined.

Main Menu | Quit

Pump Performance Program

This program is a tool to assist in measuring the performance capabilities for a generic pump. The user enters data for certain parameters (propellant type, input option, and pump characteristics) and the program determines the inlet flow conditions, outlet flow conditions, and various pump properties based on this data. Each program card also has a help card associated with it to give further information and assistance as needed. These help cards can be accessed by clicking on the question mark (?) buttons located at the top of the cards.

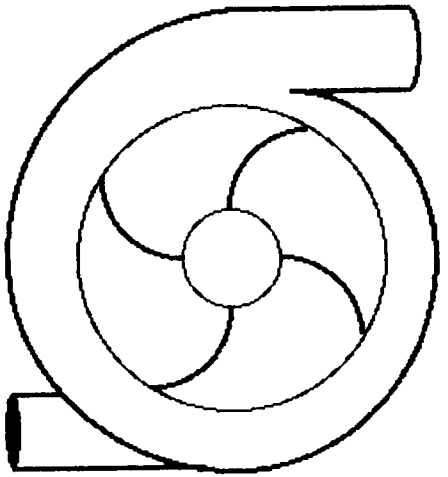
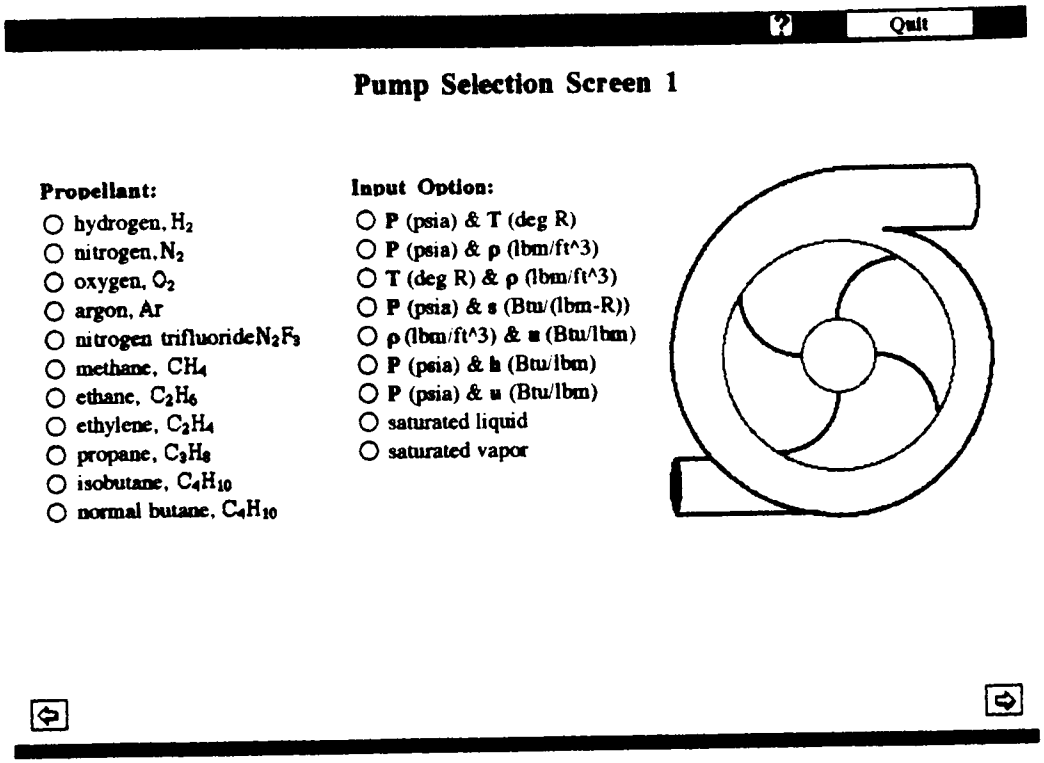


Figure 5. PPP Introductory Card



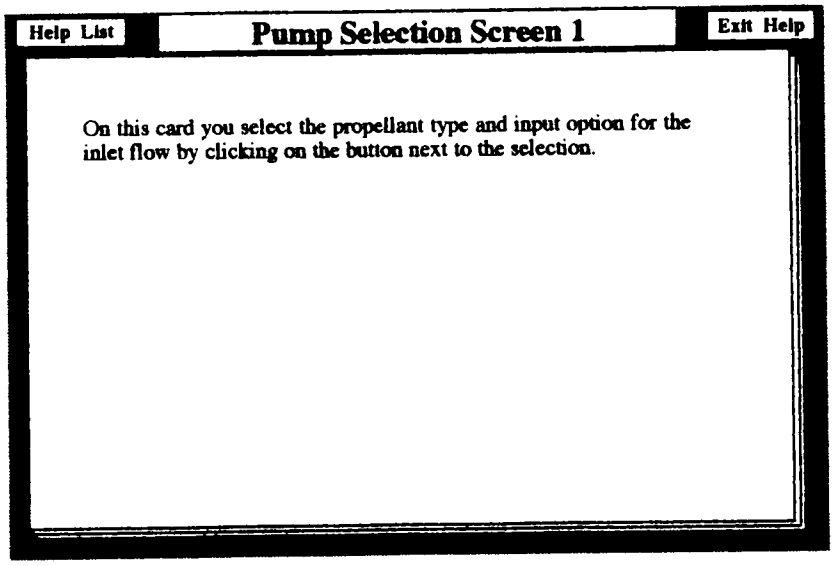


Figure 6. Selection Screen 1 and help card

Quit

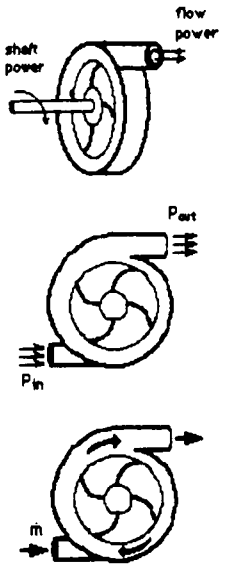
Pump Selection Screen 2

Please enter following pump information:

Efficiency, η_{pump} $\left[\frac{\text{ideal work}}{\text{actual work}}\right]$ .....
(default is 1)

Overall Pressure Ratio, $\frac{P_{out}}{P_{in}}$ .....

Pump Flow Rate, $\dot{m}\left[\frac{\text{lbm}}{\text{sec}}\right]$ .....



Help List | Exit Help

Pump Selection Screen 2

On this card you enter the *efficiency*, *pressure ratio*, and *flow rate* for the pump. If the efficiency is unknown or if the pump is being considered ideal, this input can be skipped and the program will use a default value of 1.0. It is necessary, however, to enter values for pressure ratio and flow rate in order to run all aspects of the program.

Figure 7. Selection Screen 2 and help card

The fourth card in the stack is **Selection Screen 3**, displayed in Figure 8. This card is similar to **Selection Screen 2** because values for three pump parameters (speed, friction loss, and elevation head) are entered. If the friction loss and elevation head are unknown or negligible, the PPP will default them to zero. If needed, **Selection Screen 2** can be brought back by pressing the arrow at bottom left.

The fifth card in the stack is the **Pump Data Entry** screen, shown in Figure 9. This card is where the inlet state thermodynamic data are entered. The *Selection Status* column displays the current conditions specified. To change any of these, simply click on the value to be changed and the program will switch back to the corresponding card where the change can be made.

Once the conditions in the *Selection Status* column are satisfactory, the number of data points to run have to be entered. The PPP can handle a maximum of 25 points at a time. This arbitrary number was chosen only because of the limited card size display of Hypercard. It in no way reflects any data size limits of the LRE-CP^2. If a number above or below the range 0-25 is entered, the field script will allocate 0 or 25 points, respectively. This number is then sent to the hidden background field.

After a number for the data points is entered, the pump picture on the right side of the screen disappears. In its place appear three columns: one for the number of points, and two entry columns for the data. The data can then be typed in. This data is also sent to the hidden background field. An example of a filled background field can be seen in Figure 10.

The Clear Data button at the bottom of the screen will clear out both entry columns as well as the line containing the number of data points. New data points can then be entered.

Quit

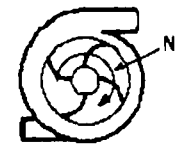
Pump Selection Screen 3

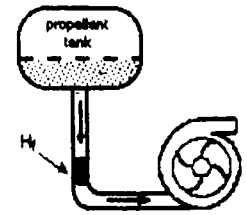
Please enter following pump information:

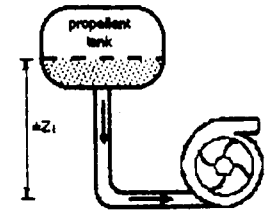
Pump Speed, N (rpm) .....

Friction Loss, H_f (ft)
(default is zero) .....

Elevation Head, Z_1 (ft)
(default is zero) .....







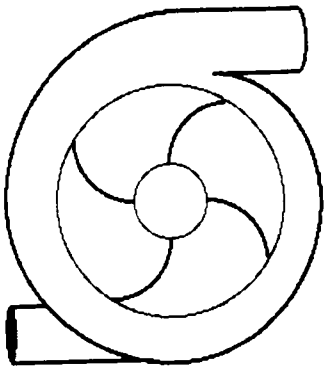
Help List | **Pump Selection Screen 3** | Exit Help

On this card you enter the *speed, friction loss*, and *elevation head* for the pump. If the friction loss to the pump and/or its elevation head is unknown or not applicable, they will default to zero. A value for pump speed is necessary in order to run all aspects of the program.

Figure 8. Selection Screen 3 and help card

Quit

Pump Data Entry



Selection Status

Propellant Type: .....
Input Option: .....
Efficiency: .....1.....
Pressure Ratio: .....
Flow Rate (lbm/s): .....
Speed (rpm): .....
Friction Loss (ft): .....0.....
Elevation Head(ft): .....0.....

Please enter number of data points: .....
(maximum number is 25)

Help List | Pump Data Entry | Exit Help

This card is where the current selection information is displayed and the data points are entered.

If you would like to make a change in one of the values in the Selection Status column, simply click on that value and the program will go back to the corresponding card where the new value can be entered.

Type in the number of data points on the line where the cursor is flashing. The maximum number of points the program will handle at one time is 25; enter any number greater and the program will assign a value of 25.

After entering a number for the data points and pressing Return, columns for placing the data will appear. After the data has been typed in you can go to the next card to run the program.

Figure 9. Pump Data Entry and help card

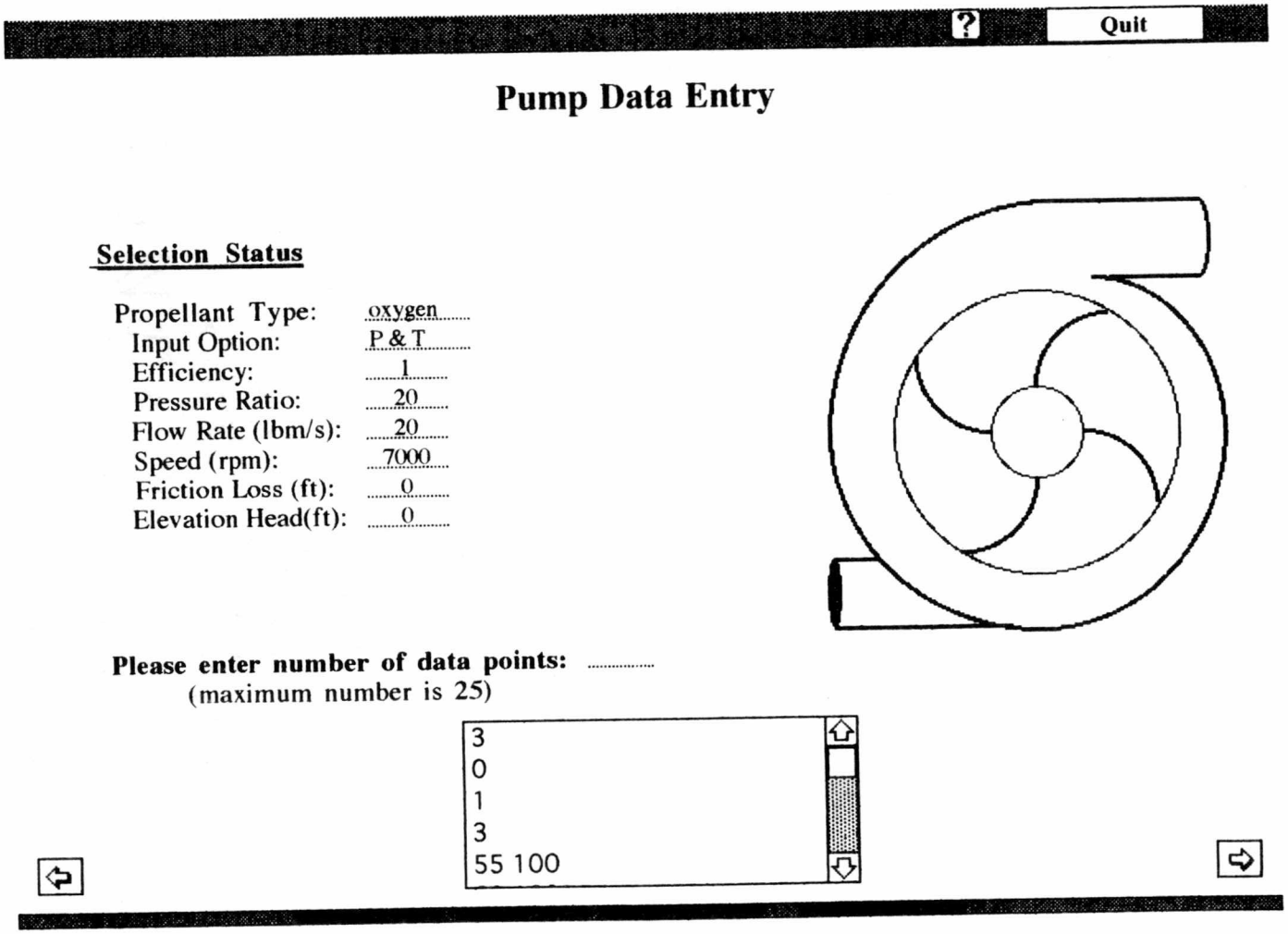


Figure 10. Example of a filled Hypercard background field

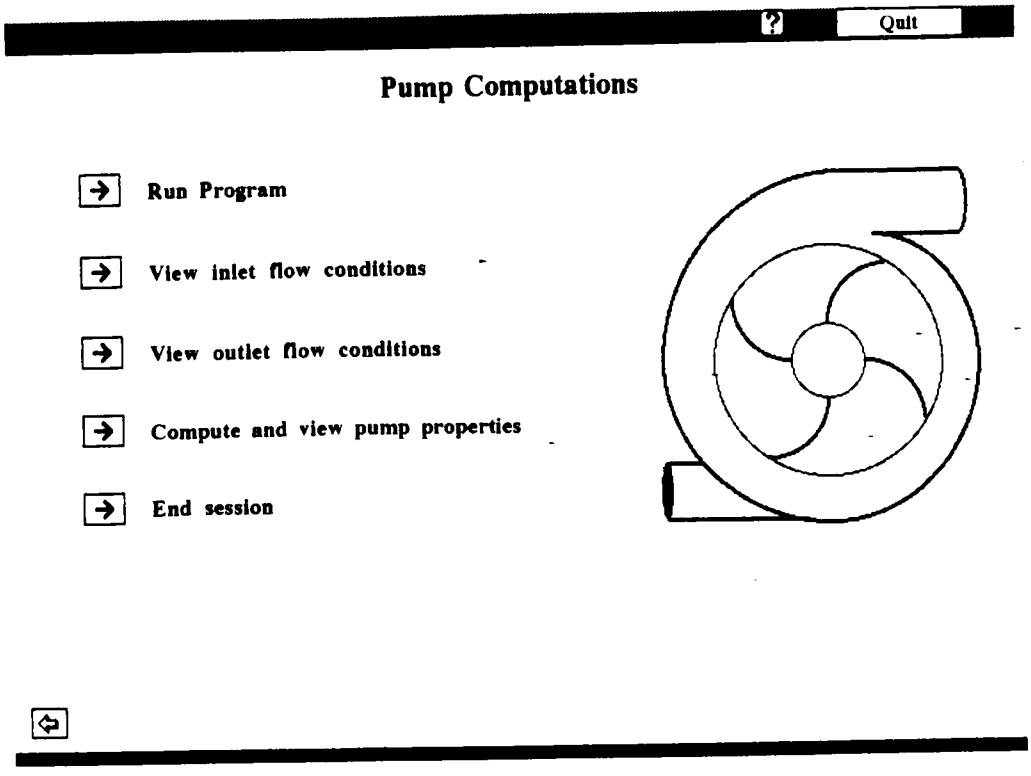
The next card in the PPP stack is the **Pump Computations** screen, shown in Figure 11. This card contains the script that connects the PPP stack with MIPROPS. The script is made active when the Run Program button is pressed.

Discussion of the Run Program script: The script of the Run Program button is the most intricate one in the PPP stack and deserves special attention. A copy of the script can be found in Appendix B. MIPROPS is called three separate times by the script to compute the inlet and outlet thermodynamic states of the pump.

The beginning section of the script is concerned with making sure all pertinent information has been entered. This includes a fluid, an input option, and all the values for those inputs. If any one of these conditions is not met, the script instructs an error message to be shown. Execution is stopped and will not resume until the condition(s) are satisfied.

Once the initial requirements are fulfilled, the script goes on to compute the complete thermodynamic state for each point at the pump inlet. The information in the hidden background field (fluid type, input option, number of data points, and the data) is transferred to an unformatted text file to be read by MIPROPS. MIPROPS is then launched with this file as input.

Once MIPROPS is finished computing, the results are sent to an output data file. It was necessary to insert a five second delay into the Run Program script before opening the output data file. This is because an error is generated if MIPROPS and Hypercard have the output file open at the same time. The output is read into the PPP stack, where a REPEAT loop organizes the data and places it onto the **Pump Inlet Flow Conditions** card



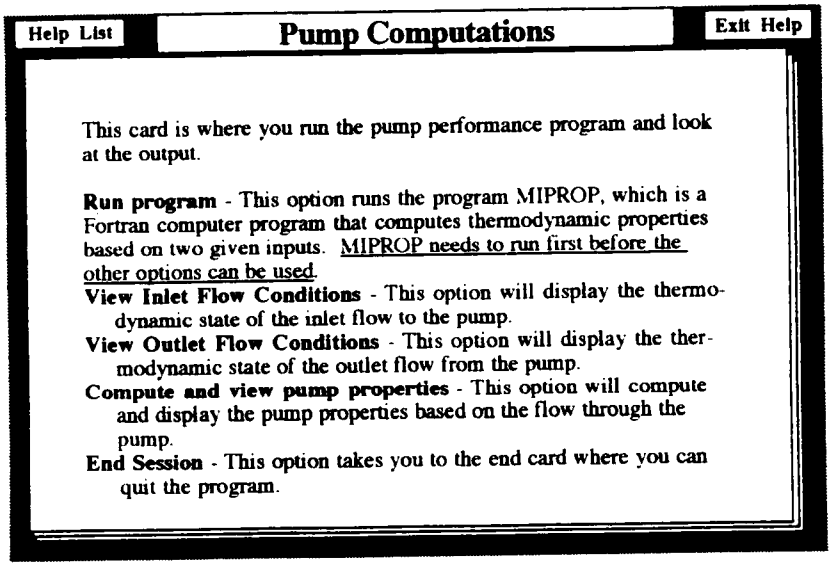


Figure 11. Pump Computations and help card

(see Figure 12). Any messages generated in MIPROPS are sent to a message file and then forwarded to the pump stack.

Predicting the fluid state at the pump outlet requires two additional calls to MIPROPS. The reason for this is seen by examining the general energy equation for open systems (Bejan,1988):

$$\frac{dE}{dt} = \dot{Q} + \dot{W} + \sum_{in} \dot{m}(e+Pv) - \sum_{out} \dot{m}(e+Pv) \qquad (4\text{-}10)$$

Assuming a steady, adiabatic system, equation (4-10) simplifies to

$$\dot{W} = \sum_{in} \dot{m}(e+Pv) - \sum_{out} \dot{m}(e+Pv) \qquad (4\text{-}11)$$

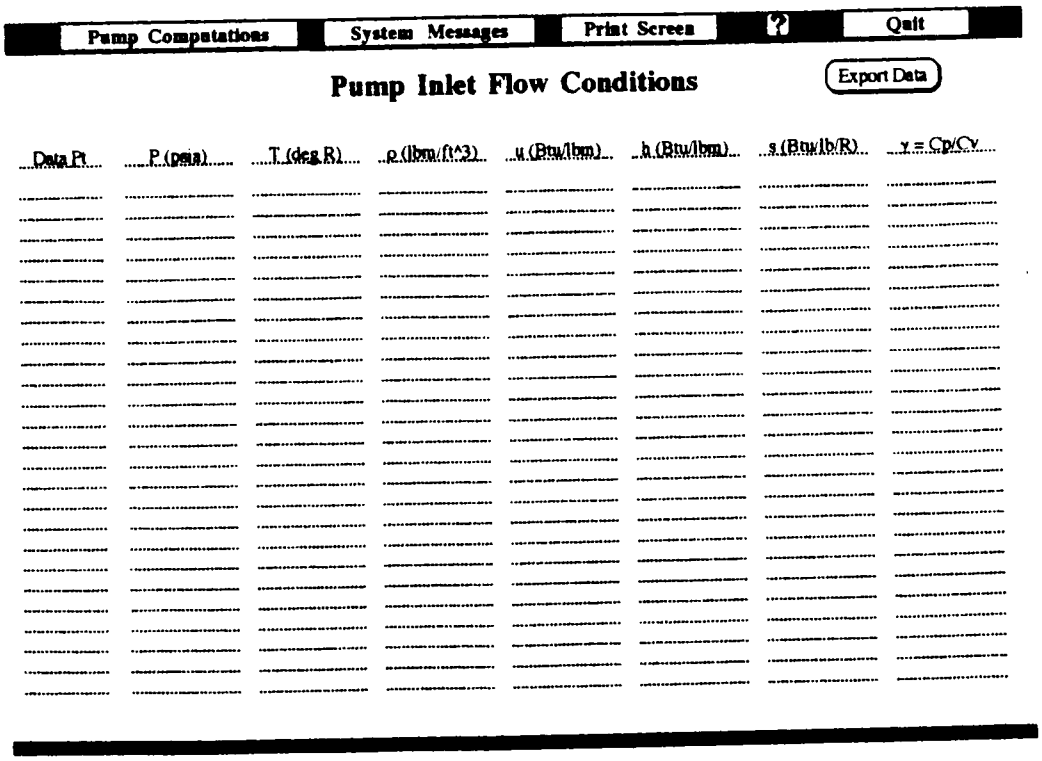
The term **e** can be represented by $\mathbf{e = (u + 1/2V^2 + gz)}$. Equation (4-11) then becomes

$$\dot{W} = \sum_{in} \dot{m}h_o - \sum_{out} \dot{m}h_o \qquad (4\text{-}12)$$

where the stagnation enthalpy $h_o = h + \frac{1}{2}v^2$, and $h = u + pv$ (potential energy is neglected). Since steady flow was already established through continuity principles in (4-8), and changes in kinetic energy can be neglected across the pump, equation (4-12) reduces to

$$\dot{W} = \dot{m}_{pump}(h_{in} - h_{out}) \qquad (4\text{-}13)$$

Here we will adopt the convention that energy into a system is positive, energy out is negative, so



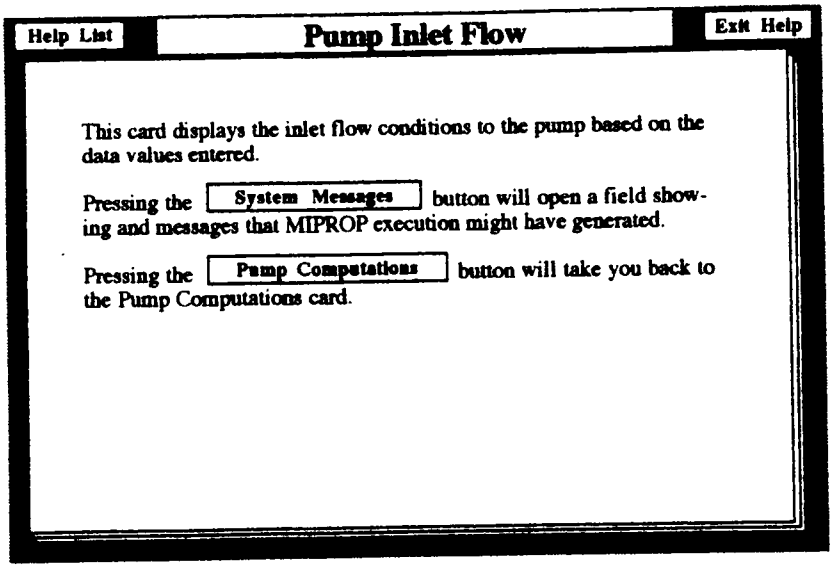


Figure 12. Pump Inlet Flow Conditions and help card

$$\boxed{\dot{W}_{pump} = \dot{m}_{pump}(h_{out} - h_{in})} \qquad (4\text{-}14)$$

The pump efficiency, η_{pump}, expresses the ideal pump power over the actual pump power

$$\eta_{pump} = \frac{\dot{W}_i}{\dot{W}_{act}} = \frac{h_{out_i} - h_{in}}{h_{out} - h_{in}} \qquad (4\text{-}15)$$

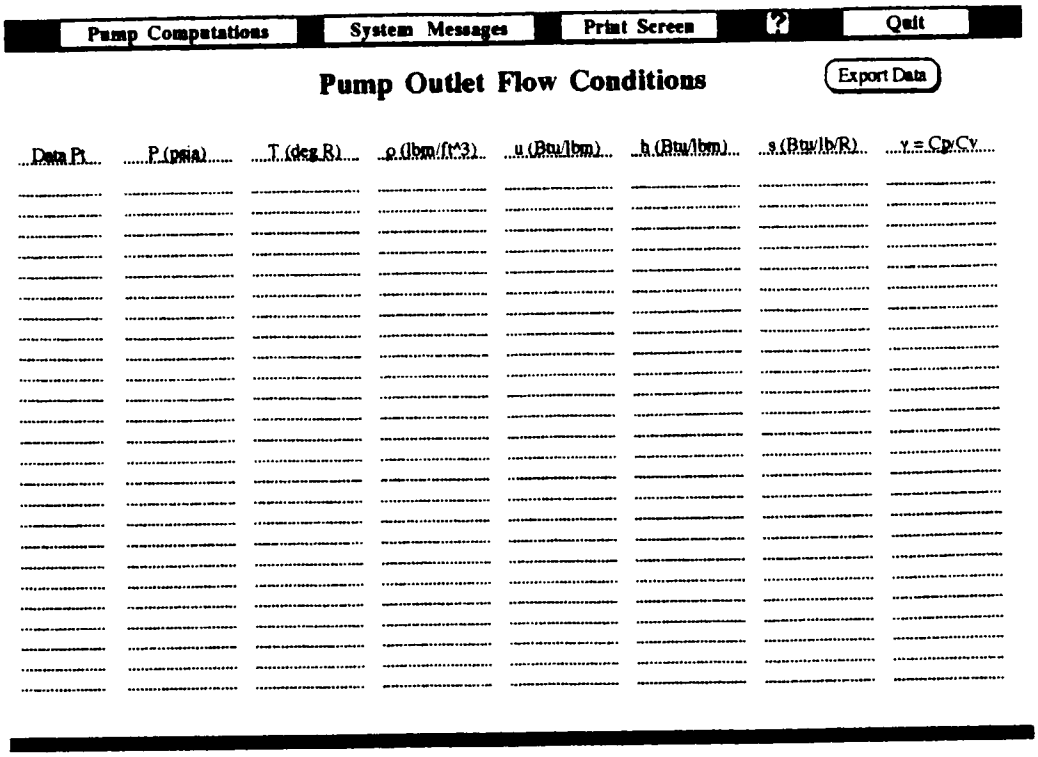
By rearranging (4-15) we obtain

$$h_{out} = h_{in} + \frac{h_{out_i} - h_{in}}{\eta_{pump}} \qquad (4\text{-}16)$$

Thus, to get the actual enthalpy (and therefore the actual thermodynamic state) at the pump outlet, the ideal outlet enthalpy must be found.

To calculate the ideal (i.e., reversible) outlet enthalpy, the Run Program script uses pressure (P) and entropy (s) as the calling inputs for the second MIPROPS run. This pressure (P_{out}) is the inlet pressure (P_{in}) multiplied by the pressure ratio across the pump, ($\frac{P_{out}}{P_{in}}$). The entropy used for the second MIPROPS run is the entropy at the pump inlet (s_{in}), which was determined in the first MIPROPS run. P_{out} and s_{in} for each data point are sent to the hidden background field and subsequently transferred to MIPROPS. MIPROPS then computes the rest of the variables.

The thermodynamic variables calculated in the second MIPROPS run represent the *ideal* (isentropic) flow state at the pump outlet. The actual flow state is established with the third MIPROPS run. For the third run, the calling inputs are P_{out} and h_{out}, where h_{out} is calculated from (4-16). MIPROPS executes a third time, and the output is sent to the **Pump Outlet Flow Conditions** card (Figure 13). Any system messages generated by



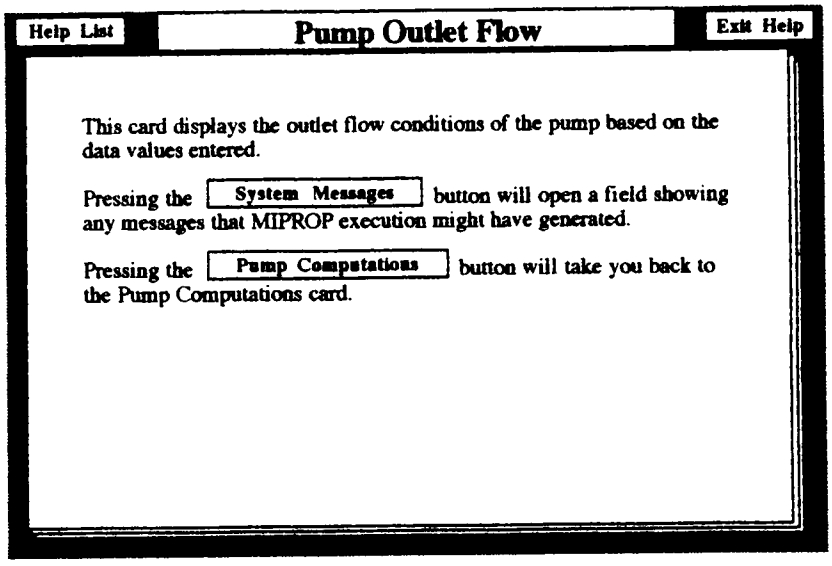


Figure 13. Pump Outlet Flow Conditions and help card

MIPROPS in the third run are sent to the messages file and then to the pump stack.

The interface between MIPROPS and the pump stack is set up so that MIPROPS is never seen. It is kept in the background while the Run Program script is executing. The user only sees a pop-up field that signals the program is running and when it is finished.

When the Run Program script is finished, three additional buttons appear on the **Pump Computations** card: View inlet flow conditions, View outlet flow conditions, and Compute and view pump properties.

Pressing the View inlet flow conditions button opens the **Pump Inlet Flow Conditions** card. This card displays the thermodynamic state variables (P,T,ρ,u,h,s,γ) at the pump inlet for each data point. To examine any messages generated by MIPROPS for the inlet conditions, press the System Messages button at the top of the card. A pop-up field will appear displaying the messages (if any), which are labeled by data point. The Pump Computations button will reopen the card by the same name. Pressing the Print Screen button will print the screen if a hard copy of the data is needed. The Export Data button gives the user the option to transfer the data to a permanent file if desired.

Pressing the View outlet flow conditions button opens the **Pump Outlet Flow Conditions** card. This card displays the thermodynamic state variables at the pump outlet for each data point. To examine any messages generated by MIPROPS, press the System Messages button. The Pump Computations, Print Screen, and Export Data buttons are the same as for the **Pump Inlet Flow Conditions** card.

The Compute and view pump properties button will calculate ΔH, pump fhp, NPSH, and N_s for each data point. The results are displayed on

the **Properties of Pump** card, shown in Figure 14. The Pump Computations, Print Screen, and Export Data buttons have the same function here as with the inlet and outlet flow condition screens.

In calculating ΔH, the density at the pump inlet is used, which is conventional (Huzel,1971).

As seen by equation (4-3), determination of the NPSH requires an evaluation of the fluid vapor pressure. A fourth order equation with empirical coefficients was used for this purpose. It is:

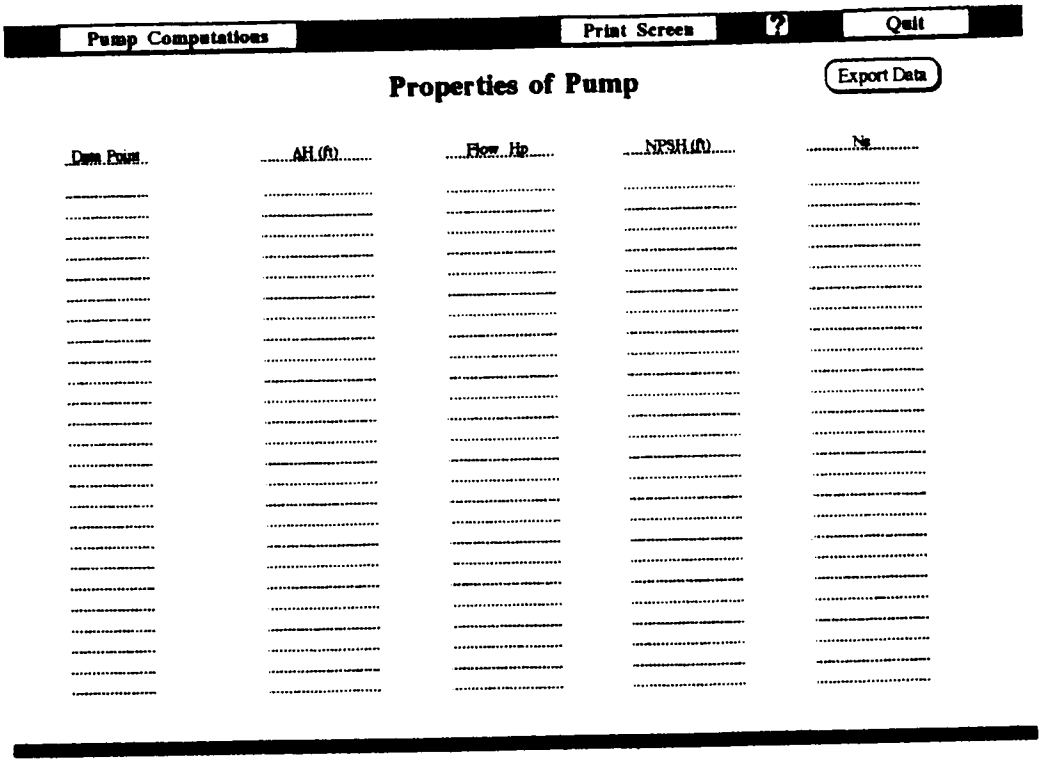
$$\ln P_v = \ln P_t + V_p(1)x + V_p(2)x^2 + V_p(3)x^3 + V_p(4)x^4 + V_p(5)x(1-x)^{Vp(6)} \quad (4\text{-}17)$$

where $x = (1 - \frac{T_t}{T})(1-\frac{T_t}{T_c})$ and the coefficients $V_p(1)$ through $V_p(6)$ are dependent on the fluid (Younglove,1982) and (Younglove & Ely,1987).

If the Quit button is pressed anywhere in the PPP, the final card in the stack is shown (see Figure 15). On this card there are three options: return to the previous card, go back to the main menu of the LRE-CP^2, or quit the Hypercard application. If the Quit Hypercard button is pressed, all current data in the pump stack will be erased.

Sample Calculations

For this example, the data were taken from the hypothetical A-1 Stage Engine described in NASA SP 425 (Huzel,1971). The oxidizer propellant used is liquid oxygen. The following information is known for the A-1 turbopump:



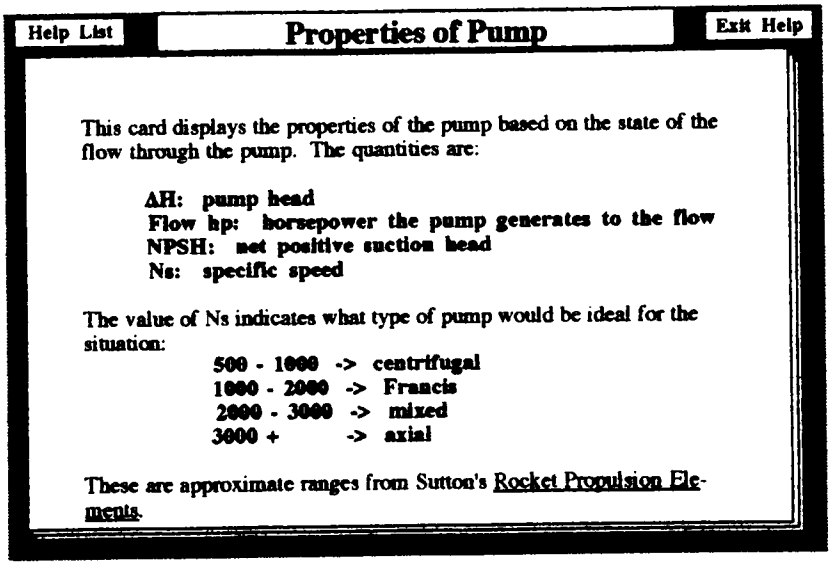


Figure 14. Properties of Pump and help card

Pump Performance Program

Return to previous card

Return to Main Menu

Quit Hypercard

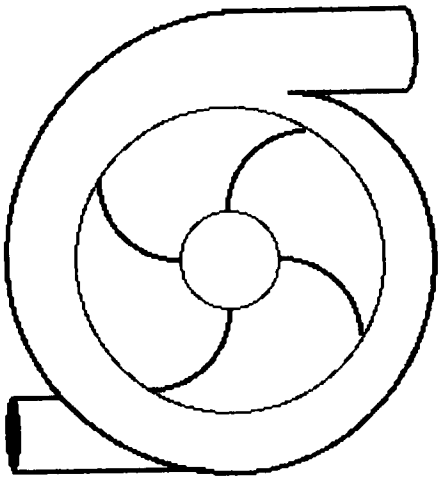


Figure 15. PPP End Card

$\rho = 71.38$ lbm/ft^3	$\eta = 0.707$
$P_{in} = 55$ psia	$T_{in} = -297.6$ °F = 162 °R
$P_{out} = 1505$ psia	$P_{tank} = 60$ psia
$\dot{m} = 1971$ lbm/sec	$H_{elev} = 3.5$ feet
$N = 7000$ rpm	$H_f = 5$ psia = 10.1 feet

Find the thermodynamic state at the pump inlet and outlet, and the properties ΔH, N_s, NPSH, and fhp developed.

Figures 16 and 17 display the data entry and output cards for this example, and Table 5 displays the percent differences with those provided independently by NASA SP 425 (Huzel,1971).

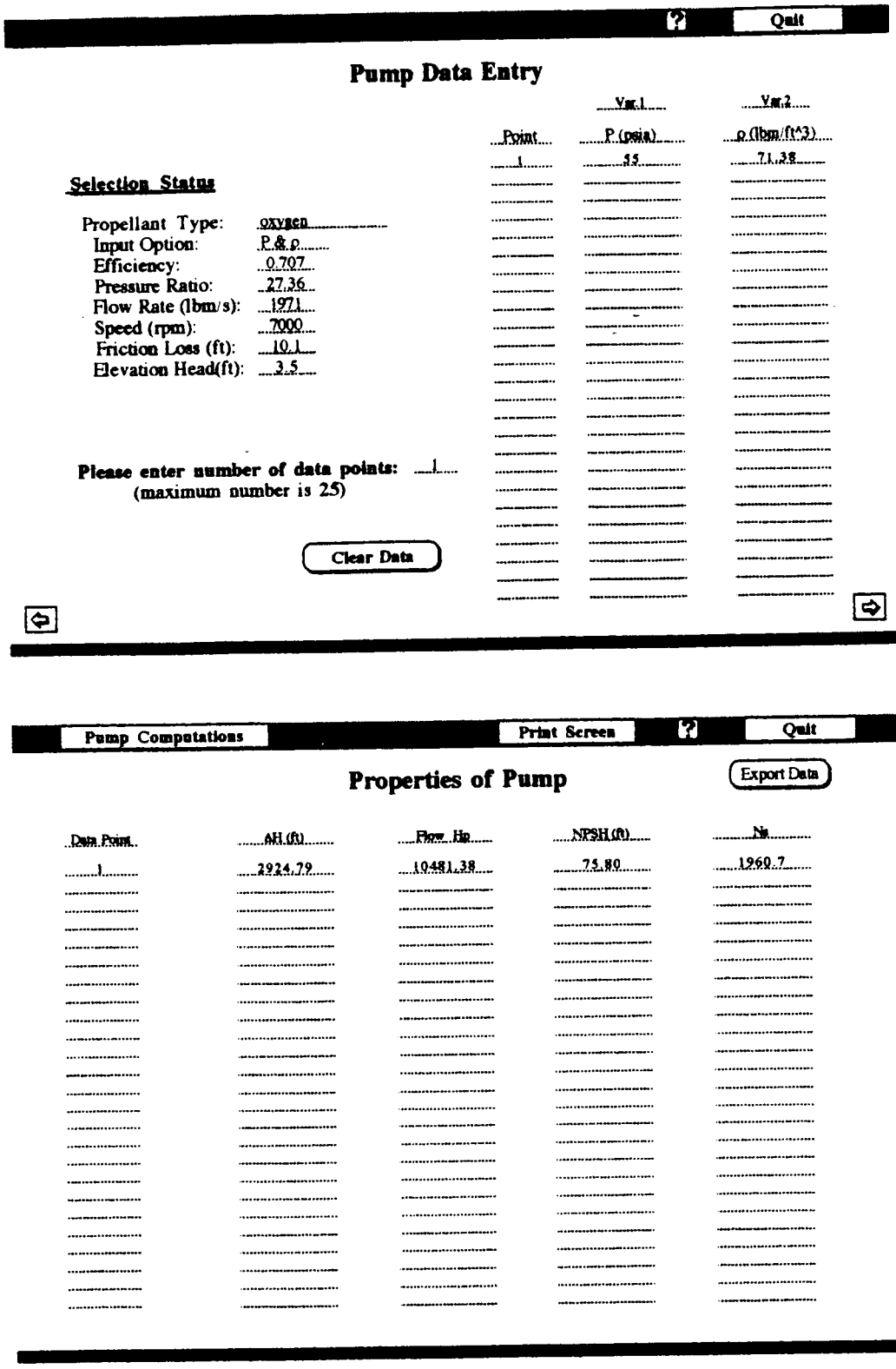


Figure 16. PPP Data Entry and Pump Properties for A-1 Stage Engine

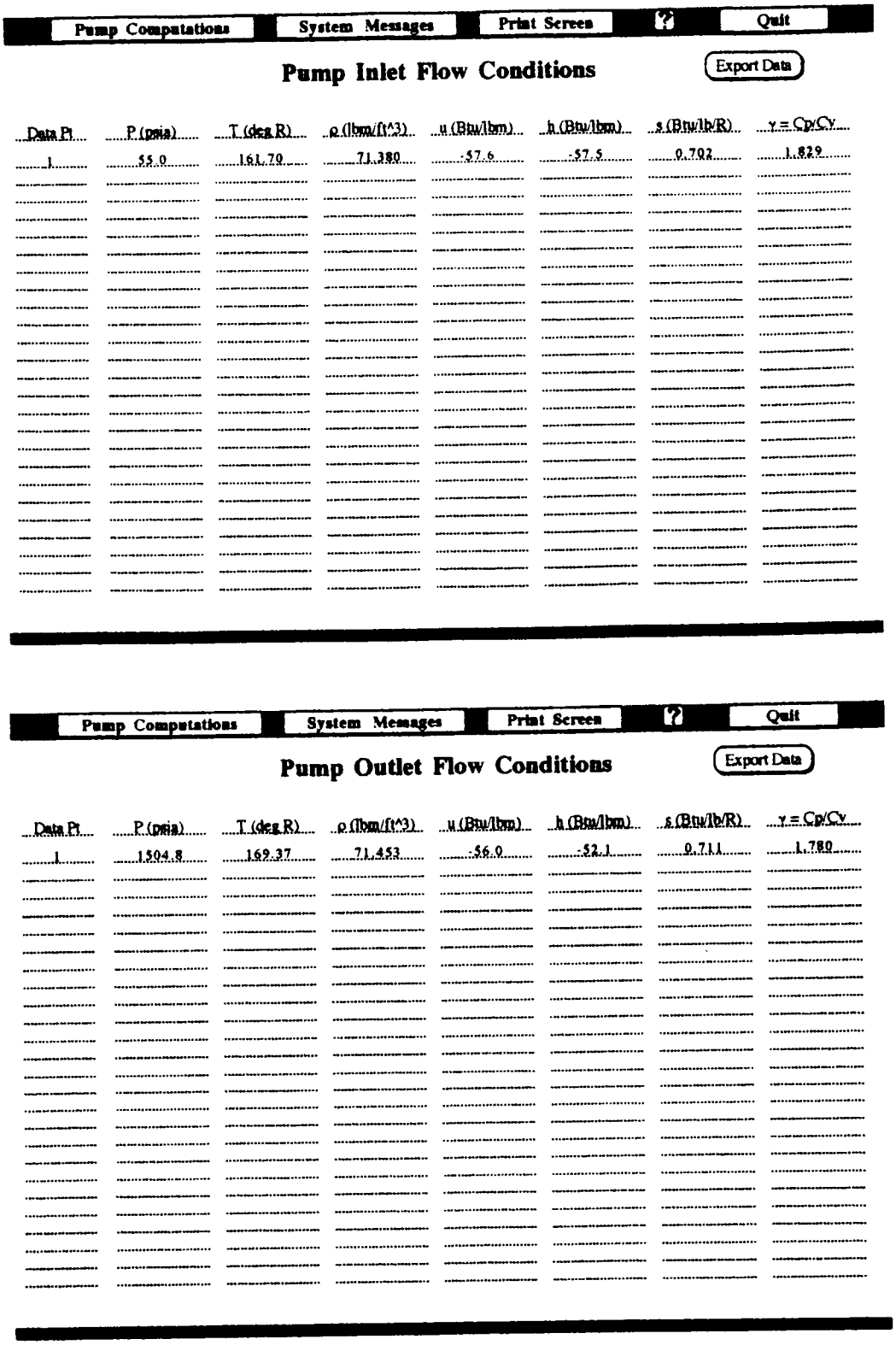


Figure 17. PPP Inlet and Outlet Flow Conditions for A-1 Stage Engine

Table 5. A-1 Stage Engine Calculation Results

Property	LRE-CP2	NASA SP 425	% Difference
ΔH (ft)	2925	2930	0.17
fhp (hp)	10481	10500	0.18
NPSH (ft)	85.90	85.00	1.05
N_s	1961	1980	0.96

The values for ΔH and fhp agree very well. The difference in NPSH is most likely due to round off errors in the empirical coefficients of the Younglove equation. A different equation for N_s was used in the NASA SP 425, which would explain the slight deviation with the LRE-CP2 value.

LRE-CP2 Valve Performance Program

The Valve Performance Program is the second of three rocket engine component stacks contained in the LRE-CP2. The purposes of this chapter are to discuss the basic theory and equations underlying the Valve Performance Program (VPP) and to describe how the program was developed.

Background Theory

There are various types of valves. Butterfly, ball, poppet, blade, sleeve, spool, and plug valves are examples of ones found in rocket engines (NASA SP-8097). Individual valve design and selection are largely experienced-based and depend on the system of integration. In contrast,

analysis of general valve performance is relatively straightforward. For a steady-state, adiabatic valve, the 1st and 2nd laws of thermodynamics reduce to (Bejan,1988):

$$h_{in} = h_{out} \tag{5-1}$$

$$\dot{S} = \dot{m}(s_{out} - s_{in}) \geq 0 \tag{5-2}$$

The performance characteristics to be determined by the VPP are:

- the *thermodynamic state* (P,T,ρ,u,h,s) at the valve inlet and outlet
- the *valve flow coefficient*, C_d
- the *valve loss coefficient*, K_v

Following is a brief review of these characteristics:

Thermodynamic state: The thermodynamic states of a fluid at the inlet and outlet of a valve in the LRE-CP^2 are represented by the quantities P, T, ρ, u, h, and s. As mentioned in Chapter 4, knowledge of any two of these properties will determine the state of the flow at a given point.

Valve flow coefficient, (C_d): The valve flow coefficient is a parameter for measuring valve performance at a certain opening angle (θ_v). It expresses the relationship between pressure drop across the valve and the valve flow rate. There are several different equations in the literature for expressing C_d. The one used in the LRE-CP^2 is (Tullis,1989):

$$C_d = \frac{1}{\sqrt{\left(\frac{890 d^4}{C_v^2} + 1\right)}} \tag{5-3}$$

where $C_v = \dfrac{\dot{V}}{\sqrt{\Delta P / \left(\frac{\rho_{fluid}}{\rho_{H_2O}}\right)}}$ (Benedict,1980)

The quantity $\dot{V}$ in the above equation represents volumetric flow rate in gallons per minute. The values for C_d are only approximate. They are heavily influenced by valve type and design (Tullis,1989).

Valve loss coefficient, (K_v): The valve loss coefficient is another parameter for measuring valve performance. It relates the total pressure drop across a valve per unit dynamic pressure of the flow to the inlet of the valve. In equation form it is expressed as (Benedict,1980):

$$K_v = \frac{144*\Delta P}{\frac{\rho V^2}{2g_c}} \qquad (5\text{-}4)$$

Also, K_v is related to C_v by (Benedict,1980):

$$K_v = \frac{890.40*d^4}{C_v^2} \qquad (5\text{-}5)$$

Development of the Valve Performance Program:

The VPP stack is composed of 16 total cards (seven program, nine help). Access to the help cards is attained by pressing on the question mark (?) buttons located at the top of the cards. The graphic for the valve element was created using MacDraw II and MacPaint software. The picture was designed to represent a generic valve device, so no specific geometry is shown. The purpose of this stack is to predict a valve's performance based on user supplied information. The selection and design of specific valve components are a complex task and difficult to generalize into a performance program.

Pressing the Valve Element button on the LRE-CP^2 main menu screen will take you to the Valve Performance Program. The first card is the introductory card that explains the basics of the stack. It is shown in Figure 18. The Main Menu button at the top of this card will take you back to the main screen. The Quit button will take you to the last card in the stack where the program session can be ended.

By pressing the arrow at bottom right the user moves to the **Valve Selection Screen**, seen in Figure 19. On this card a fluid and an entry option are selected from the available choices. Only one fluid and one entry option may be chosen at a time. Each choice has a number associated with it so that MIPROPS can read it. Once the desired selections have been made, the information is sent to a hidden background field for later transmittal to MIPROPS.

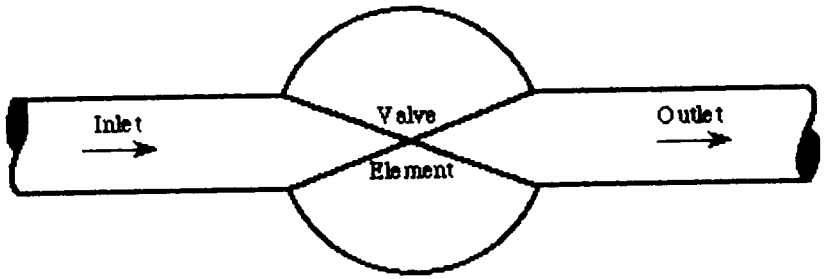
After selecting a fluid and entry option, you next proceed to the **Valve Computations** card, shown in Figure 20. Under the *Selection Status* column heading the fluid selected is listed. To change the fluid, simply press on its name or use the arrow button at bottom left to go back to the **Valve Selection** screen to make the change.

The next step is to enter the number of data points to be run. An arbitrary maximum of 10 was chosen only because of the limited space on the card screen. If a number above or below the range 0-10 is entered, the field script will allocate 0 or 10 points, respectively. Once a number is entered, its value is sent to the hidden background field and six columns (or five, if saturation conditions were chosen) appear at bottom left. From left to right they are:

Pt. Column: This column lists the number of data points chosen.

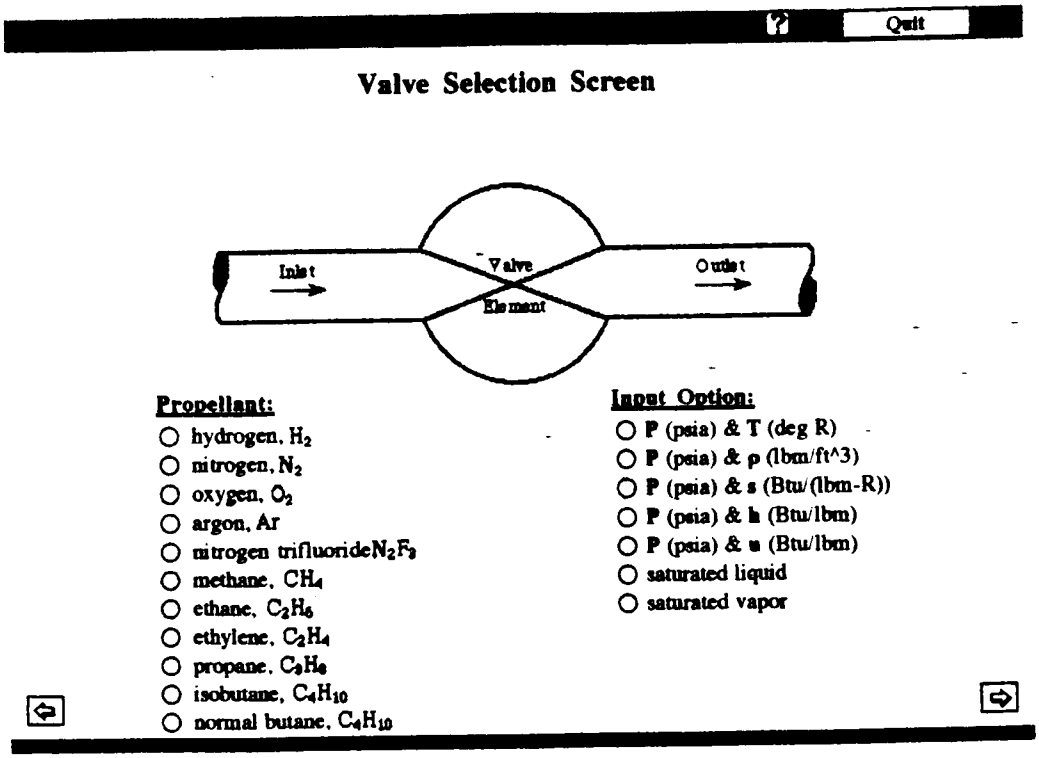
Main Menu | Quit

Valve Performance Program



This program is a tool to assist in measuring the performance for generic valves. The user inputs data on required pressure drop, two inlet quantities, mass flow, and valve opening angles. The program determines the full array of inlet and outlet flow conditions, as well as valve performance coefficients. There is also an option for heat transfer if desired. Each program card also has a help card associated with it to give further information and assistance as needed. These help cards can be accessed by clicking on the question mark (?) buttons located at the top of the cards.

Figure 18. VPP Introductory Card



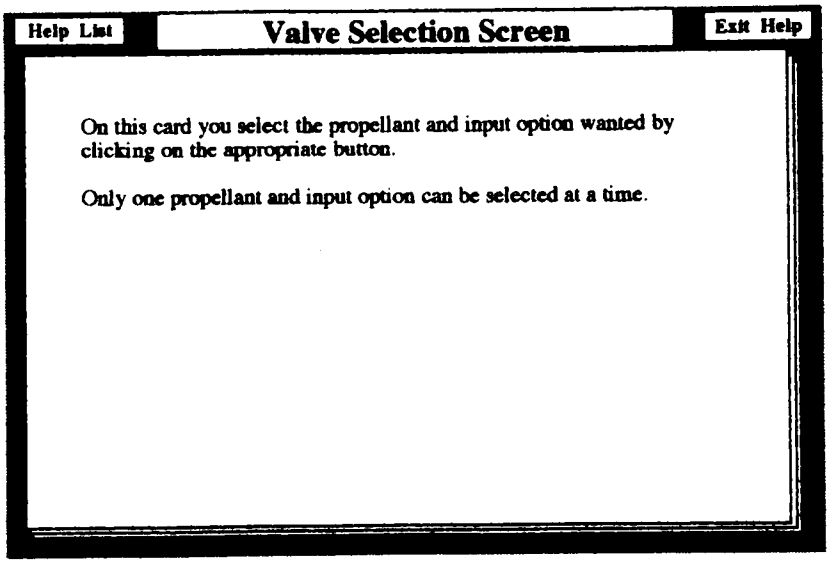
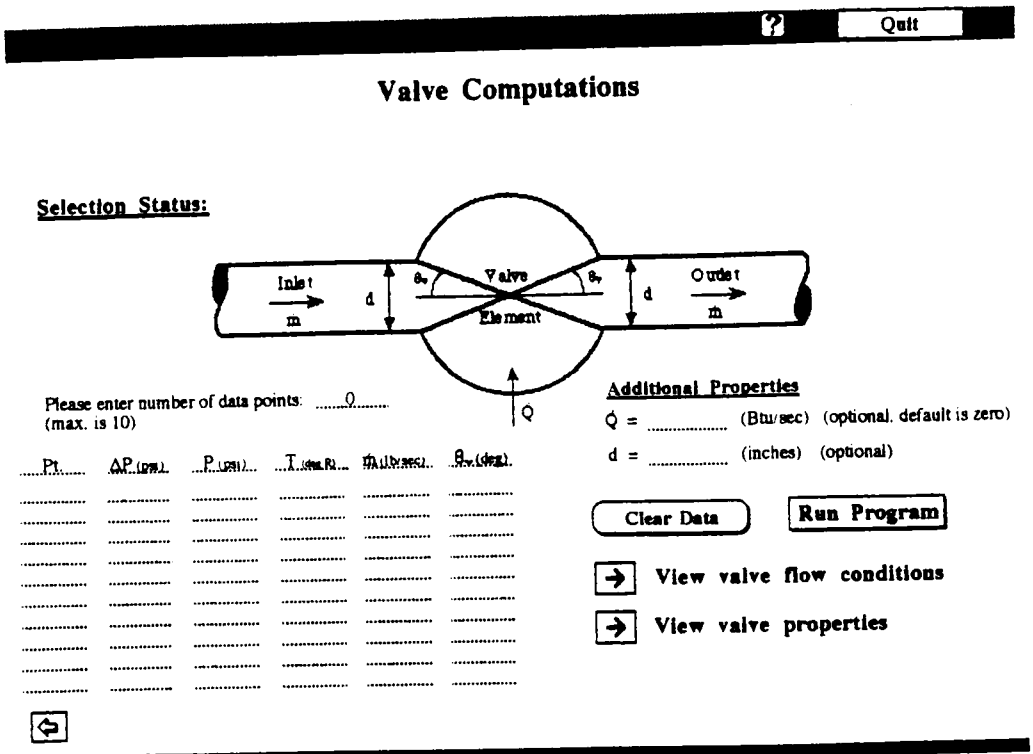


Figure 19. Valve Selection Screen and help card



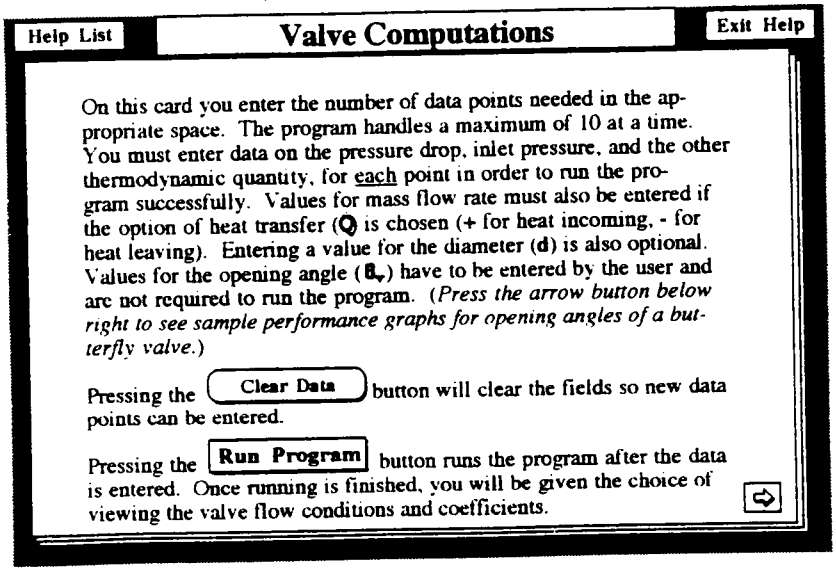


Figure 20. Valve Computations and help card

ΔP column: In this column the total pressure drop across the valve is entered. A value must be entered for each data point; otherwise the LRE-CP[2] will display an error message and stop execution.

The next two columns are for the pressure and whatever other quantity was specified in the entry option. If saturation conditions were chosen, only one column will be present.

$\dot{m}$ column: In this column the flow rate through the valve is entered for each data point (if applicable, see Discussion of the Run Program script below).

θ_v column: In this column the valve opening angle is entered for each data point (if applicable, see Discussion of the Run Program script below).

Under the *Additional Properties* heading are entry lines for heat transfer, $\dot{Q}$, and valve diameter, d. Both of these quantities are optional (see Discussion of the Run Program script below). For $\dot{Q}$, positive (+) values are for heat transfer to the valve, negative (-) values are for heat transfer from the valve.

The Clear Data button is analogous to the one in the Pump Performance Program. It clears the data from all the entry fields so new data can be entered.

Discussion of the Run Program script: As with the Pump Performance Program, the script of the Run Program button in the Valve Performance Program is quite lengthy and merits a special analysis. (A listing of the script can be found in Appendix B.) MIPROPS is called twice by the script to determine flow conditions at the valve inlet and outlet.

The initial part of the script verifies that all the necessary information has been entered. This includes a fluid, an entry option, and all the values for the inputs. An error message appears if these conditions are not met.

The next section of the script depends on whether the user entered a heat transfer to the valve system. The reason for this is best illustrated by examining the continuity and energy equations for the valve. Assuming steady, uniform flow, the continuity equation reduces to

$$\dot{m}_{in} = \dot{m}_{out} = \dot{m} \tag{5-6}$$

The general energy equation is (from Ch.4)

$$\frac{dE}{dt} = \dot{Q} + \dot{W} + \sum_{in} \dot{m}(e+Pv) - \sum_{out} \dot{m}(e+Pv) \tag{5-7}$$

If we assume the valve has steady flow, does no work, and has negligible changes in potential and kinetic energies, (5-7) reduces to

$$\dot{Q} + \dot{m}h_{in} = \dot{m}h_{out} \tag{5-8}$$

As can be observed from (5-8), if $\dot{Q} \neq 0$, then values for $\dot{m}$ *must be entered.* Otherwise, an error message is displayed. If the valve is considered adiabatic, then (5-8) becomes

$$h_{in} = h_{out} \tag{5-9}$$

which indicates the valve operates with isenthalpic throttling (i.e., it acts as a flow restricting device (Black & Hartley,1985)). Therefore, if adiabatic flow

is assumed, no values for $\dot{m}$ are necessary. The enthalpy values in equation (5-9) are the *static* enthalpies; for all liquids and for gases with low pressure changes across a valve, the kinetic energy (KE) can be neglected. In general, however, KE must be accounted for. The kinetic energy could be determined if necessary, and a velocity squared term would be added to both sides of equation (5-9). If *stagnation* quantities are being used instead of static, then the kinetic energies, if any, are already being accounted for.

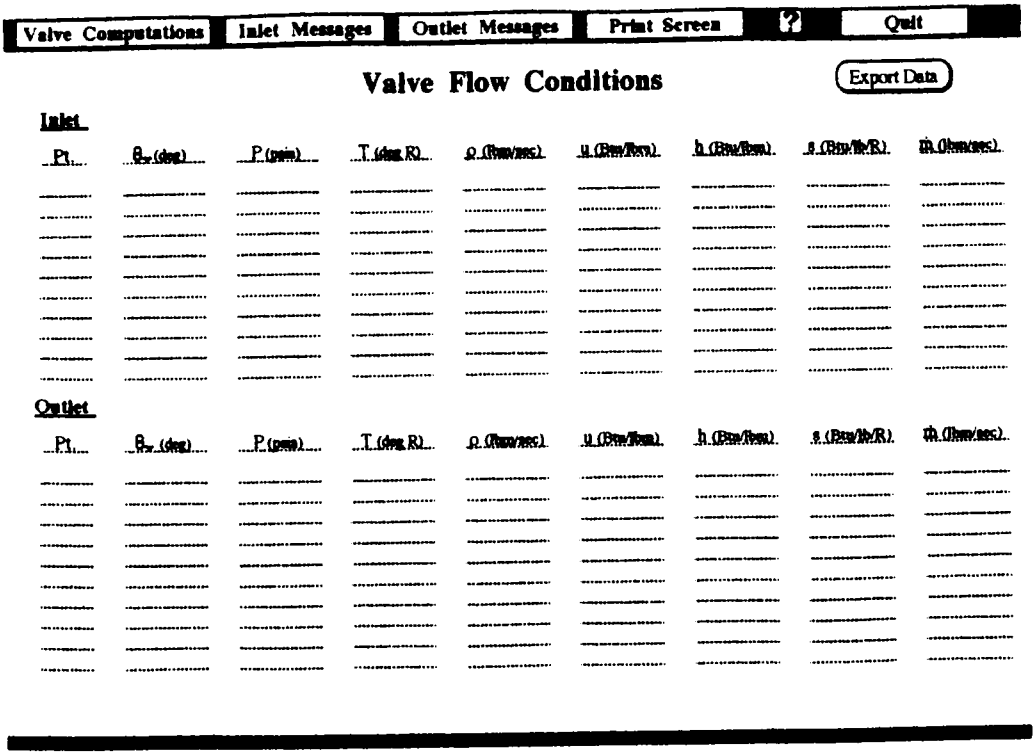
The Run Program script then continues to compute the thermodynamic variables for each point at the valve inlet. The information sent to the hidden background field is transferred to an input data file. MIPROPS is then started with this data file as input. When MIPROPS is finished, it sends the results to an output data file. The Run Program script then reads this data into the valve stack, where a REPEAT loop organizes the data and writes it to the **Valve Flow Conditions** card, shown in Figure 21. Any system messages generated by MIPROPS are sent to a message data file and subsequently written to the valve stack.

To calculate the thermodynamic variables at the valve outlet, the exit pressure and enthalpy are used as calling inputs to MIPROPS. The exit pressure is found from the pressure drop across the valve:

$$P_{out} = P_{in} - \Delta P \qquad (5\text{-}10)$$

The exit enthalpy is established by equation (5-8) or (5-9), depending on whether the valve is adiabatic or not.

These pressure and enthalpy values are then sent to the hidden background field. They are then sent to MIPROPS as input for the second run. The results from the second run are sent to the **Valve Flow Conditions**



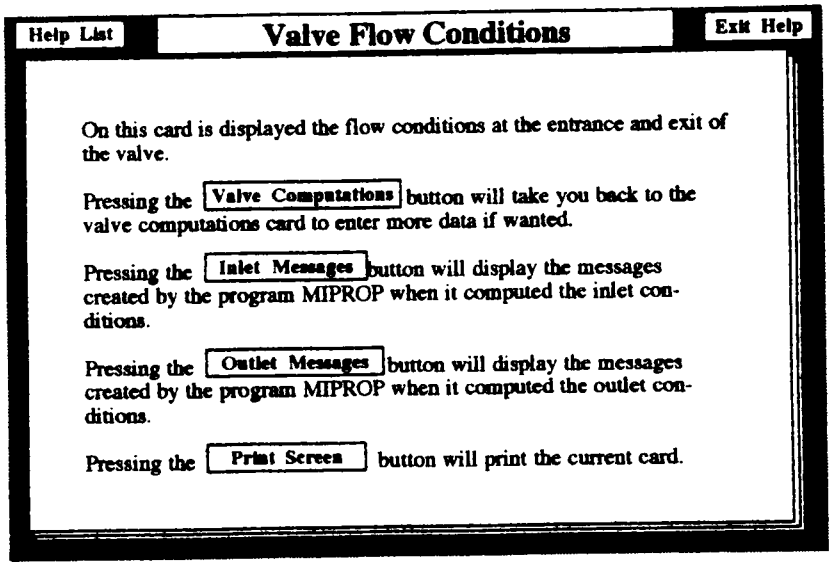


Figure 21. Valve Flow Conditions and help card

card as above. As with the inlet run, any MIPROPS messages are passed to the valve stack.

The last portion of the VPP Run Program script deals with the flow and loss coefficients, C_d and K_v. For C_d and K_v to be calculated, values for $\dot{m}$ and d must have been entered. The results are written to the **Valve Properties** card, shown in Figure 22.

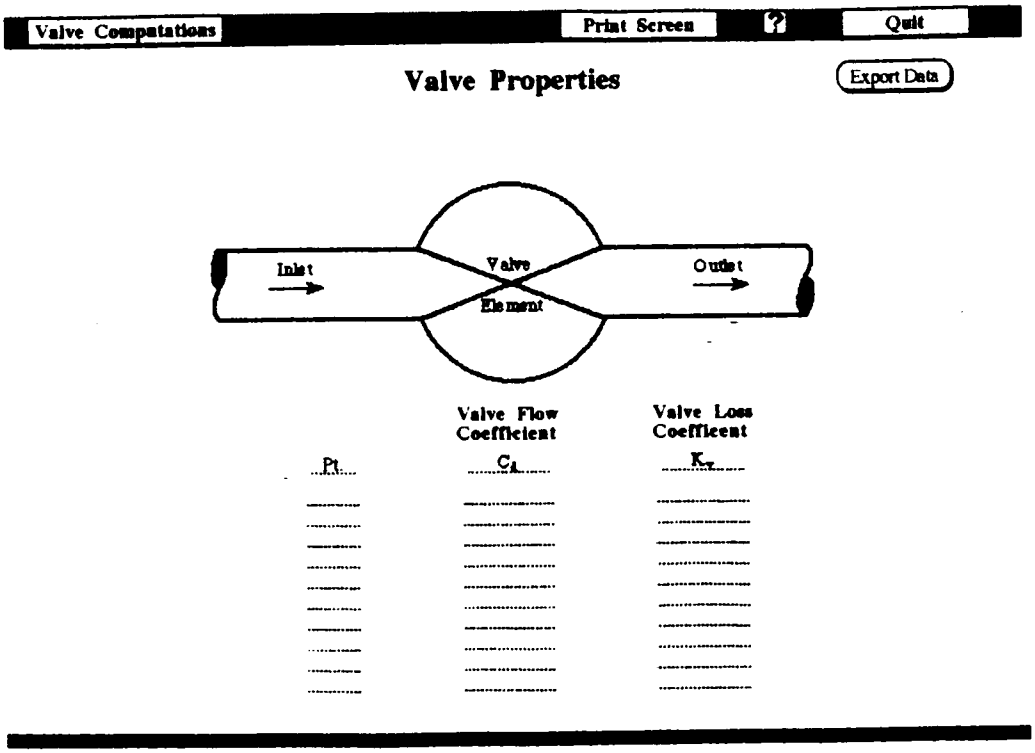
It is important to note the omission of the valve opening angle, θ_v, from the Run Program script. The reason for this is that θ_v is a function of the specific type and geometry of a valve. Since the Valve Performance Program is a generalized analysis of valve operation, it was not practical to include θ_v effects in the script calculations. The user can supply this information for his/her own benefit. Values for θ_v are usually obtained from performance curves of individual valves. Two examples of this are shown in Figure 23 for a butterfly valve. The data for these curves was taken from a report on butterfly valves (Anonymous, "Configuration, Selection, and Sizing of Butterfly Valves for Precise Pressure Control", Sverdrup and Parcel, Inc., Consulting Engineers, St. Louis, Missouri). The equations used were

$$\dot{m} = \frac{124}{144} C d^2 (2\Delta P P_{inlet} - \Delta P^2)^{0.5} T^{-0.5} \qquad (5\text{-}11)$$

$$\Delta P = P_{inlet} - \left(P_{inlet} - \left(\frac{144\dot{m}T^{0.5}}{124Cd^2}\right)^2\right)^{0.5} \qquad (5\text{-}12)$$

where $C = 2.56/10^{.02040\theta_v}$

Once the Run Program script has finished execution, a message notifies of this and two new buttons appear at bottom right of the **Valve**



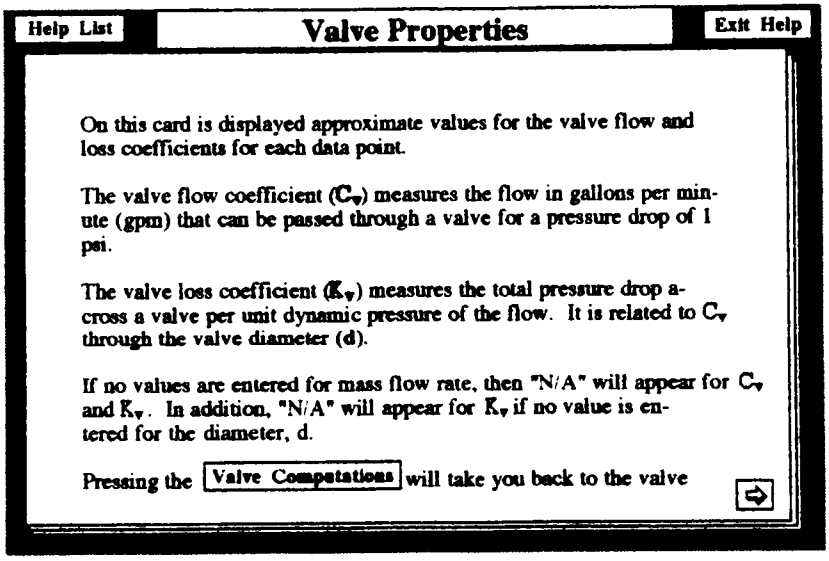


Figure 22. Valve Properties and help card

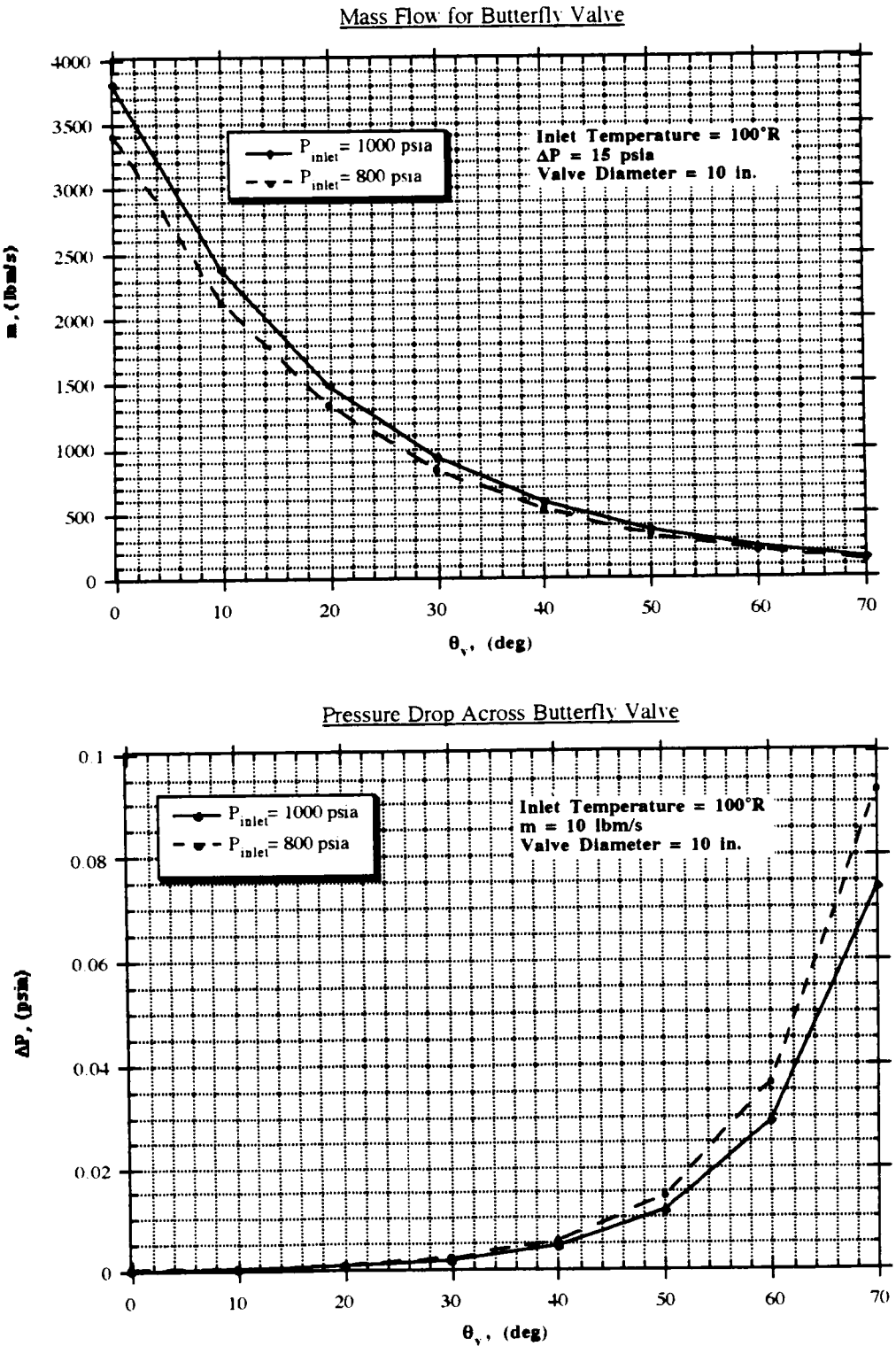


Figure 23. Examples of performance curves for a butterfly valve.
(Adapted from Anonymous, "Configuration, Selection, and Sizing of Butterfly Valves for Precise Pressure Control", Sverdrup Parcels, Inc., Consulting Engineers, St. Louis, Missouri.)

Computations card. The View valve flow conditions button will open the **Valve Flow Conditions** card, where both the inlet and outlet flow quantities are shown. Pressing the Inlet Messages and/or Outlet Messages button(s) on this card opens a pop-up field that displays any messages MIPROPS produced. The messages are listed by data point. The Print Screen button will print the current card if hard copies of the data are needed. The Export Data button gives the user the option to transfer the data to a permanent file if desired. The Valve Computations button moves control back to the **Valve Computations** card.

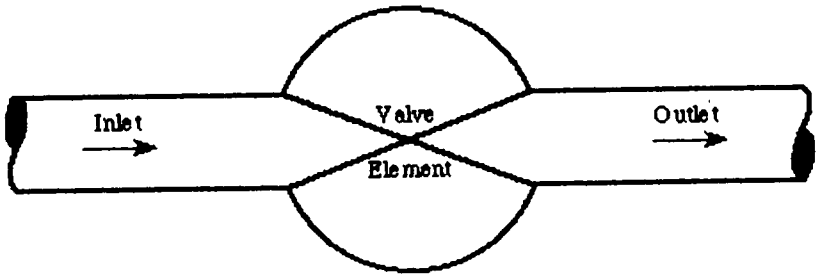
The second button that appears is View valve properties. Pressing this button opens the **Valve Properties** card where the flow and loss coefficients are displayed. The Valve Computations, Print Screen, and Export Data buttons on this card are the same as for above.

As with the Pump Performance Program, pressing the Quit button anywhere in the Valve Performance Program will open the final card in the stack, shown in Figure 24. On this card there are options to return to the previous card, return to the LRE-CP^2 main menu, or to quit Hypercard altogether. If the Quit Hypercard button is pushed, all data in the current session of the VPP will be erased.

Sample Calculations

The valve data for this example was taken from the main oxidizer valve (butterfly valve) on the hypothetical A-1 Stage Engine described in NASA SP 425 (Huzel,1971). The oxidizer propellant is liquid oxygen. The following information is known:

Valve Performance Program



Return to previous card

Return to Main Menu

Quit Hypercard

Figure 24. VPP End Card

Pressure drop across valve (ΔP) = 35 psia
Inlet valve pressure (P_{in}) = 1505 psia
Liquid oxygen density (ρ_{LO_2}) = 71.38 lbm/ft^3
Valve opening angle (θ_v) = 5 °

Find the complete inlet and outlet thermodynamic conditions for the valve. Figure 25 displays the data entry and output cards for this example.

LRE-CP^2 Propellant Tank Performance Program

The Propellant Tank Performance Program is the third component program in the LRE-CP^2. The purposes of this chapter are to discuss the basic theory underlying the Propellant Tank Performance Program (PTPP) and to describe how it was developed.

Background Theory

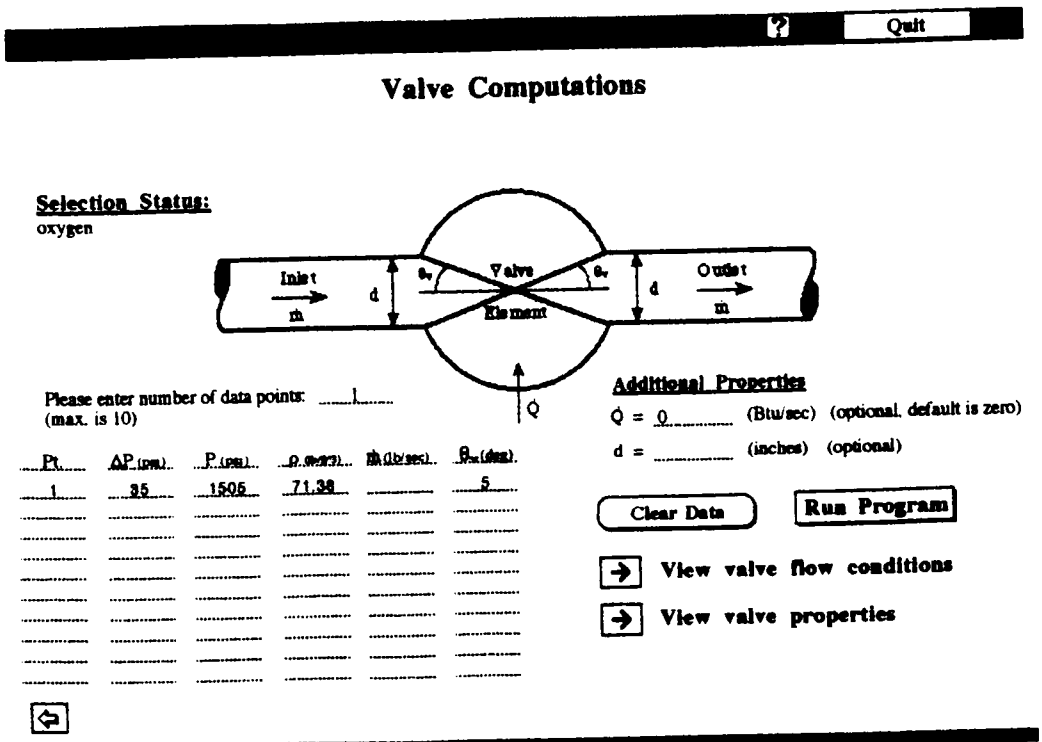
The chief interest in propellant tank performance analysis is the change in mass of the fluid during emptying or filling of the tank. For gases and vapors, the thermodynamics behind this process are quite simple. The tank is a constant volume system so that a reference volume can be defined as

$$V_o = \left(\frac{m}{\rho}\right)_{fluid} \tag{6-1}$$

For a change in gas or vapor mass the reference volume remains unchanged. This leads to

$$V_o = \frac{m_i}{\rho_i} = \frac{m_f}{\rho_f} \tag{6-2}$$

hence,



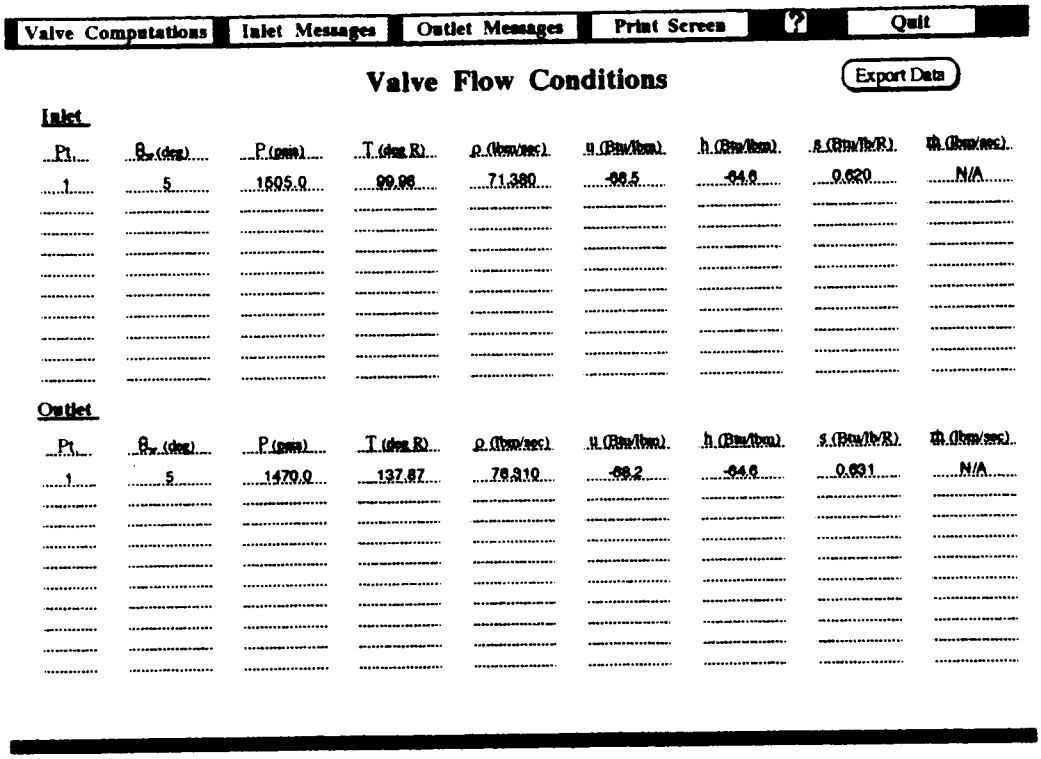


Figure 25. VPP Data Entry and Results for A-1 Stage Engine

$$m_f = \rho_f \frac{m_i}{\rho_i} \quad \text{(gases only)} \tag{6-3}$$

The final mass can be found from known initial conditions, and the final specified density, or temperature and pressure, etc. The change in mass for the tank is,

$$\Delta m = m_i - m_f \tag{6-4}$$

The user can create more general tank performance models if two-phase conditions exist in the tank, or if the dynamics of the filling/emptying process are important to predict. In the present thesis, Cases #1, #2, and #3 on the following pages were developed to show some simple examples of tank performance models.

Development of the Propellant Tank Performance Program

The PTPP stack is composed of 12 total cards (seven program, six help). Access to the help cards is attained by pressing the question mark (?) buttons located at the top of the cards. The graphic for the tank was created using MacDraw II and MacPaint software.

Pressing the Propellant Tank button on the LRE-CP^2 main menu screen will open the PTPP. The first card is an introductory card that explains the purpose of the stack. This card is shown in Figure 26. The Main Menu button at the top of this card will reopen the LRE-CP^2 main screen when pressed. The Quit button takes you to the last card in the PTPP stack where the program session can be ended. Pressing the arrow at bottom right moves you to the **Propellant Tank Selection Screen**, shown in Figure 27. On this card a selection is made in the following four categories:

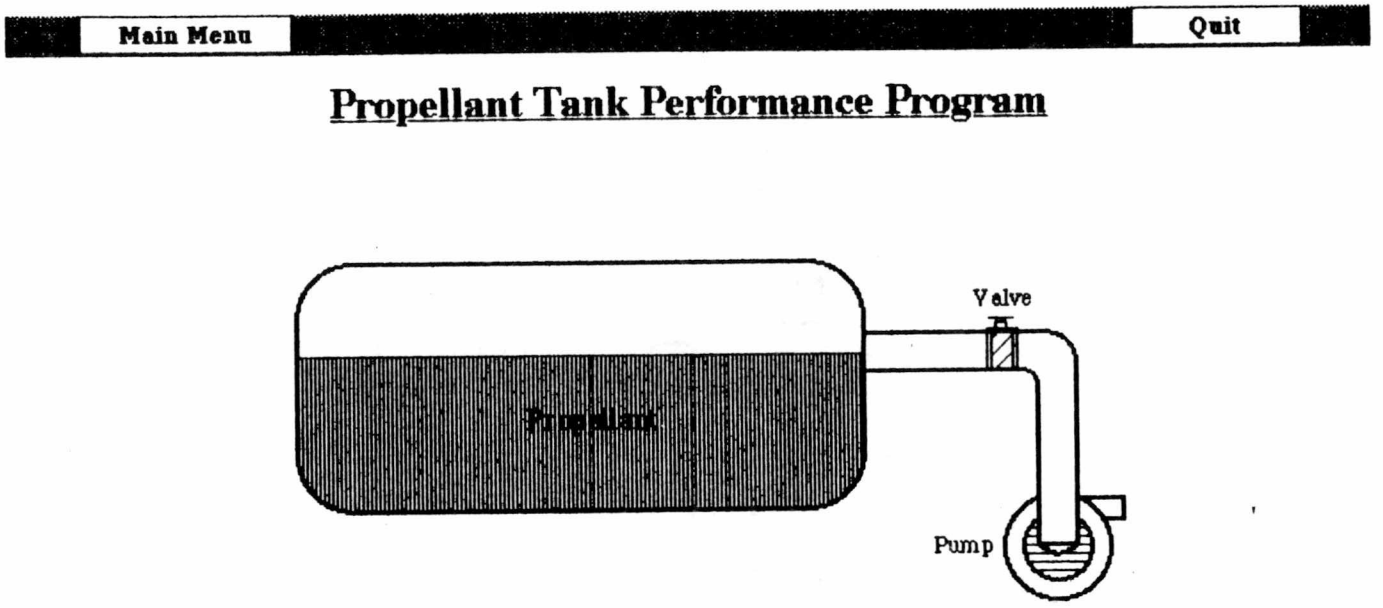
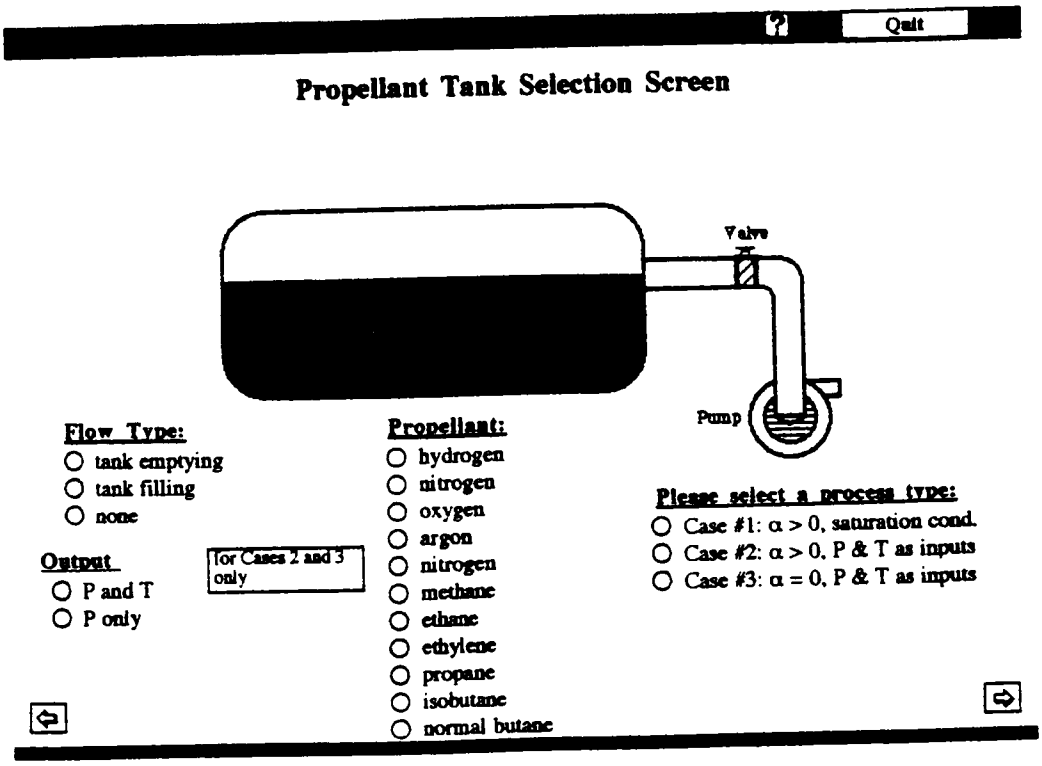


Figure 26. PTPP Introductory Card



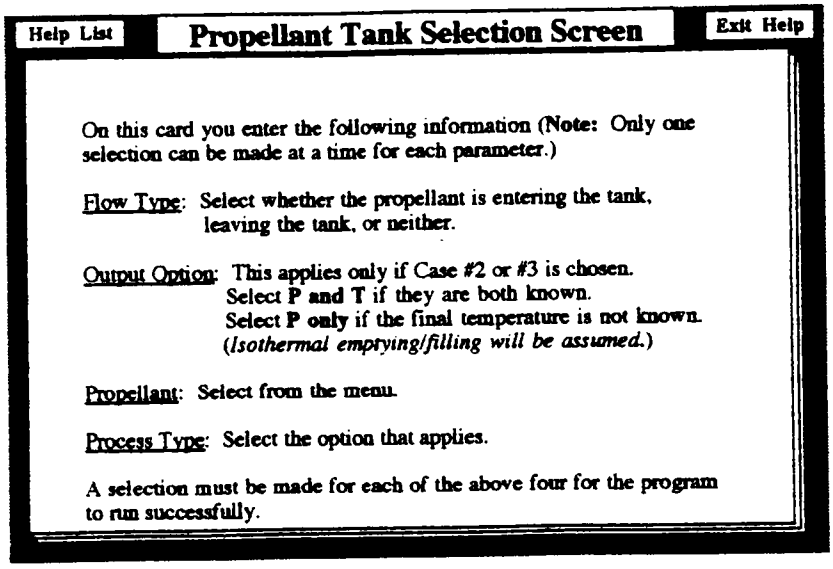


Figure 27. Propellant Tank Selection Screen and help card

Flow Type -- This is where the option for emptying of the tank, filling of the tank, or neither is chosen.

Output Option -- This section applies only if Case #2 or #3 is chosen (see Process Type below). Select the one that applies.

Propellant -- Select the fluid to be used.

Process Type -- Select the case that applies.

Each one of the choices has a number associated with it so that MIPROPS can read it. Once the user has made the selections, the information is sent to a hidden background field for transmittal to MIPROPS.

As can be seen on the **Propellant Tank Selection Screen**, there are three cases that are examined by the PTPP. Case #1 is for an equilibrium mixture of saturated liquid and vapor in the tank. The user inputs the initial temperature, T_i, the volume fraction of liquid in the tank, α_i, and the tank volume, V_o. If tank emptying is taking place, the user also inputs the final volume fraction of liquid, α_f. Temperature, pressure, and density remain unchanged for tank emptying (saturation conditions). Filling of the propellant tank is not analyzed for Case #1.

For Case #2, there is also a mixture of liquid and vapor in the tank, but saturation conditions are not present. Here the user inputs initial P and T, α_i, and V_o. For an emptying/filling process, the user also inputs final P and T or just final P, depending on what is chosen under the Output Option. If only the final pressure is entered, the PTPP assumes isothermal emptying/filling.

Case #3 is for a gaseous fluid in the tank ($\alpha_i = 0$). The user inputs initial P and T, α_i, and V_o. For an emptying/filling process, the user also inputs final P and T or just final P, similar to Case #2.

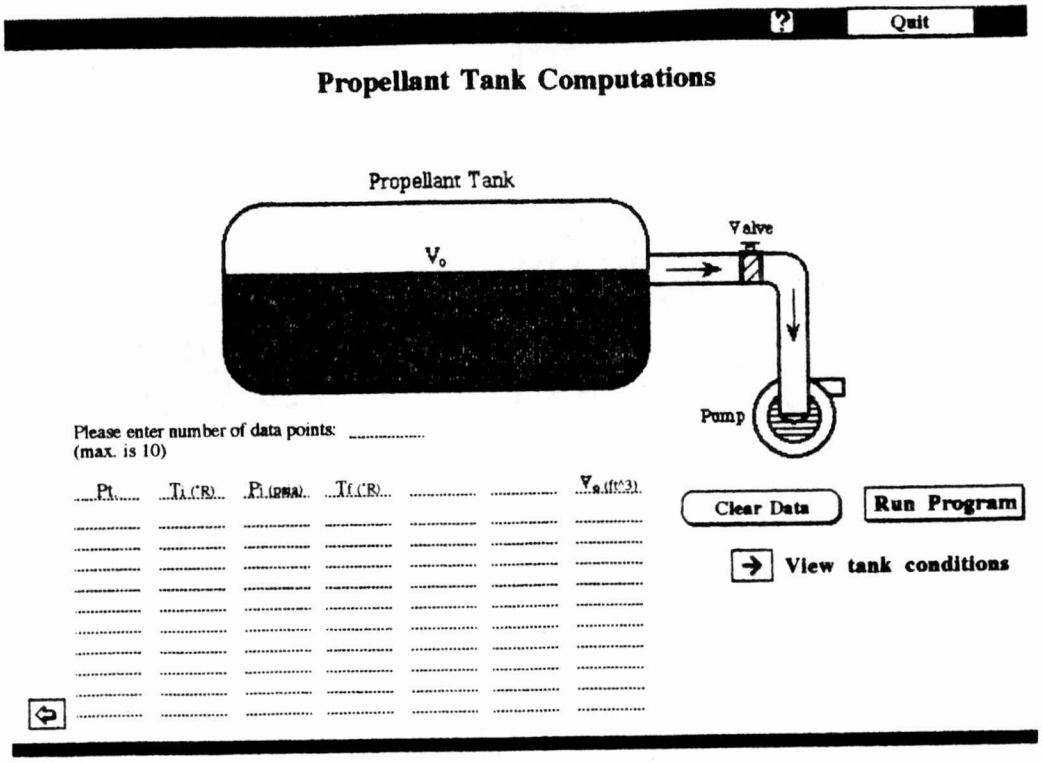
The next card is called **Propellant Tank Computations**. It can be seen in Figure 28. The *Selection Status* column at upper left lists the currently designated fluid, process type, and flow type. To make a change in any of these choices, simply click on the column or press the arrow key at bottom left to go back to the **Propellant Tank Selection Screen**.

The number of data points to be analyzed is entered next, as with the other two component stacks. The maximum number allowed for the propellant tank stack is 10 at a time. If a number above or below the range of 0-10 is entered, the field script will allocate 0 or 10, respectively. The number entered is sent to the hidden background field. A number of columns will appear at bottom left. What they are depend on which process type was chosen by the user. Values must be entered into each column appearing, for each data point, for the program to run successfully.

Also appearing with the columns are the Clear Data and Run Program buttons. The Clear Data button has the same function as in the other component stacks. The Run Program button is also similar to the others. A discussion of its script follows.

Discussion of the Run Program Script: The script of the Run Program button is the most important one in the PTPP stack. A copy of it is shown in Appendix B. MIPROPS is called twice by the script to determine the fluid state before and after the transfer process.

The script's first part is concerned with making sure all pertinent information has been entered. For the PTPP this includes a fluid, a process type, and all the values for each data point. If all these conditions are not met, an error message appears to alert of this.



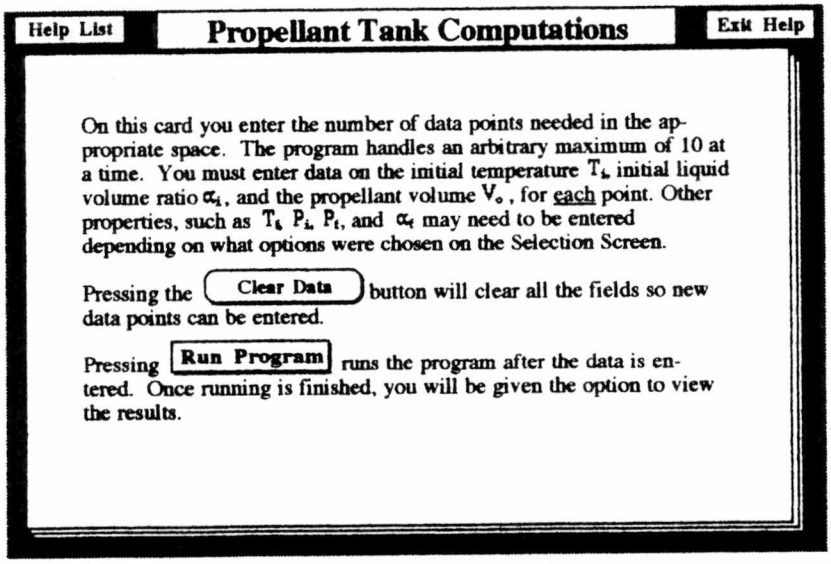
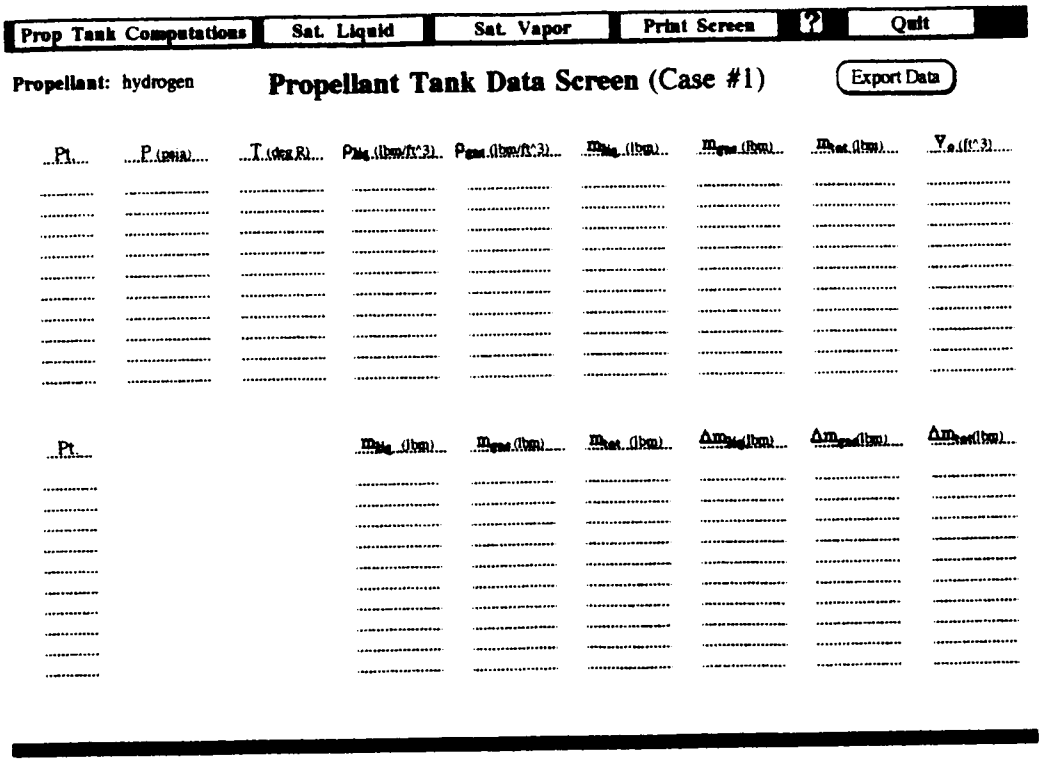


Figure 28. Propellant Tank Computations and help card

The script then proceeds to determine the initial state of the fluid in the tank based on which process type was chosen. This is done by sending the information in the hidden background field to the input data file. MIPROPS is then started. It reads from the input file and computes the initial state of the fluid, sending the results to the output data file. MIPROPS also sends any messages to the message file.

The results in the output file are then read into the PTPP stack to the **Propellant Tank Data Screen(s)**, shown in Figures 29 and 30. This is done using a Hypertext REPEAT loop. The Data Screen card for Case #1 will show the pressure, temperature, density, and mass for both the liquid and gas components, along with the propellant volume. The total mass is also displayed. If emptying of the fluid is involved, final conditions are shown, including changes in mass. Any messages in the message file are sent to hidden fields on the card. The fluid (propellant) is shown at the top left of the card. The Prop Tank Computations button takes the user back to the Computations card. Pressing the Sat. Liquid and/or Sat. Vapor button(s) opens the hidden pop-up fields containing any MIPROPS messages generated. The Print Screen button will print the card if a hard copy of the data is needed. The Export Data button gives the user the option to transfer the data to a permanent file if desired.

The Data Screen card for Cases #2 and #3 will show the pressure, temperature, density, and mass for the gas in the tank, along with the propellant volume. If emptying or filling of the tank occurs, the conditions afterward are shown, including changes in mass. Any messages in the message file are sent to hidden fields on the card. The fluid (propellant) is shown at the top left of the card. Pressing the Initial State and/or Final State button(s) will open the hidden pop-up fields containing any MIPROPS



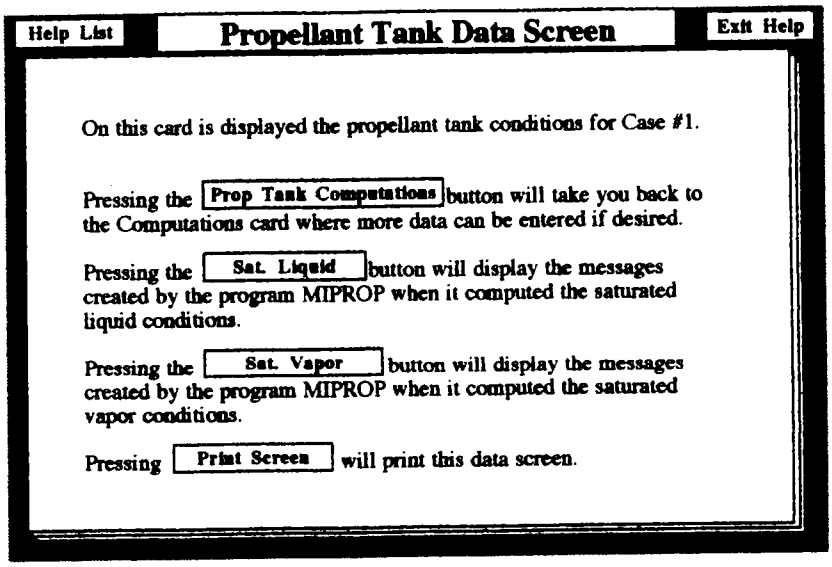
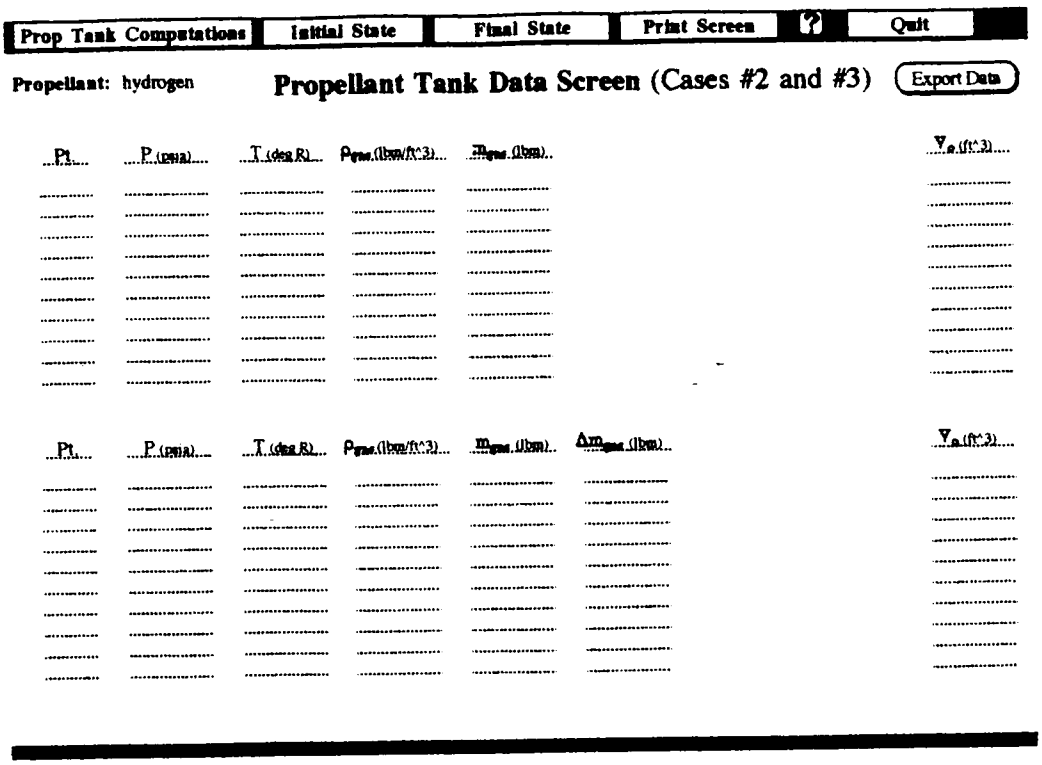


Figure 29. Propellant Tank Data Screen (Case #1) and help card



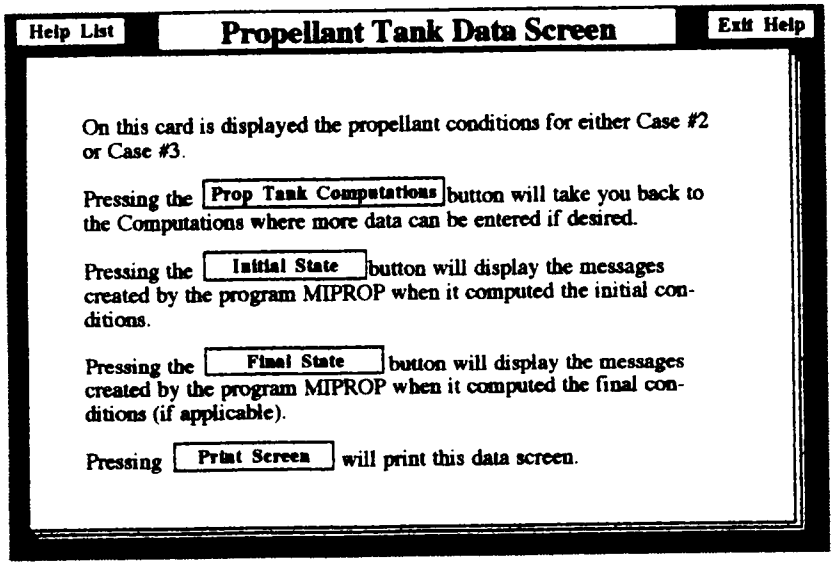


Figure 30. Propellant Tank Data Screen (Cases #2 and #3) and help card

messages generated. The Prop Tank Computations, Print Screen, and Export Data buttons have the same function as the ones for the Case #1 Data Screen.

After the Run Program script has finished, the View propellant tank conditions button appears. Pressing this button opens the appropriate **Propellant Tank Data Screen** card.

As with the other two component programs, pressing the Quit button anywhere in the stack opens the last card, shown in Figure 31. On this card there are options to return to the previous card, return to the LRE-CP^2 main menu, or to quit Hypercard altogether. If the Quit Hypercard button is pushed, all data in the current session of the PTPP will be erased.

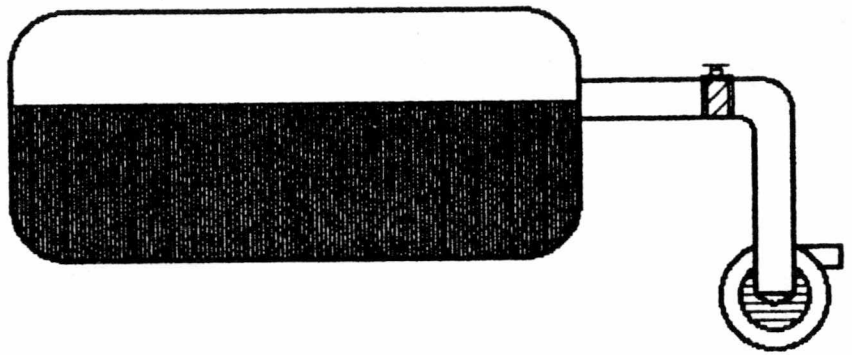
Sample Calculations

Assume a propellant tank full of oxygen in a saturated state that is to undergo an emptying process. The following information is known:

Tank reference volume (not including ullage) (V_o) = 1000 ft^3
Initial tank temperature (T_i) = 150 °R
Initial volume fraction of liquid O_2, $\alpha_i = 0.5$
Final volume fraction of liquid O_2, $\alpha_f = 0.4$

Find the mass of the liquid and gaseous components of the oxygen, before and after tank emptying. Figure 32 displays the data entry and output cards for this example.

Propellant Tank Performance Program

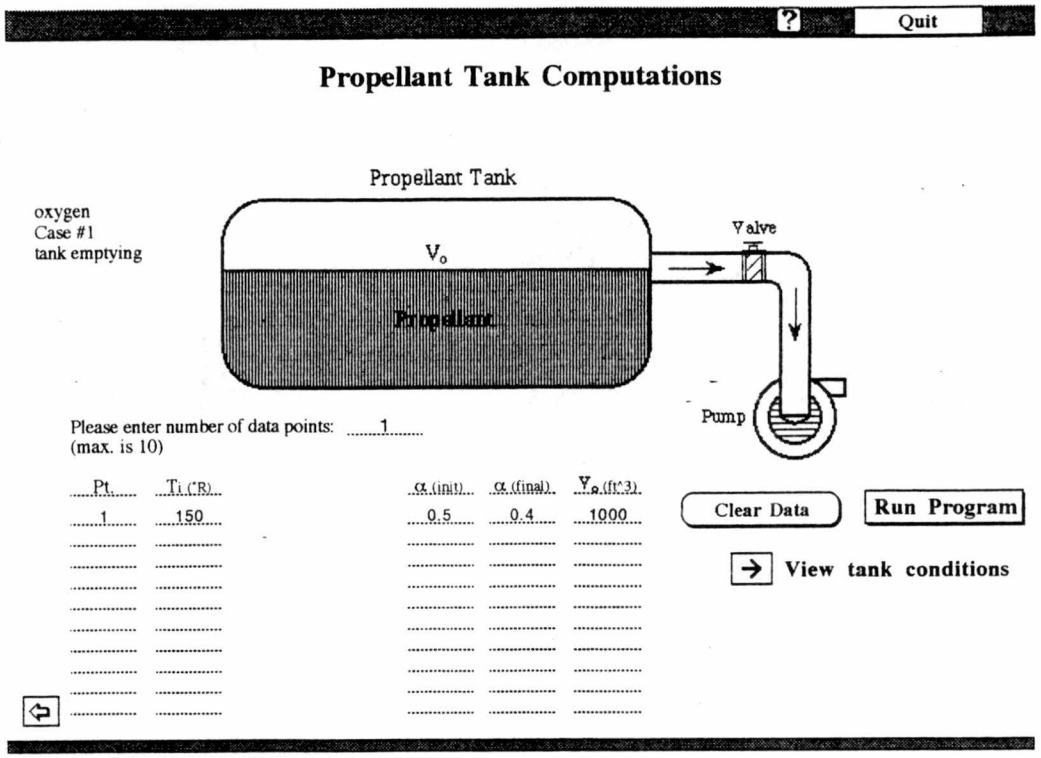


Return to previous card

Return to Main Menu

Quit Hypercard

Figure 31. PTPP End Card



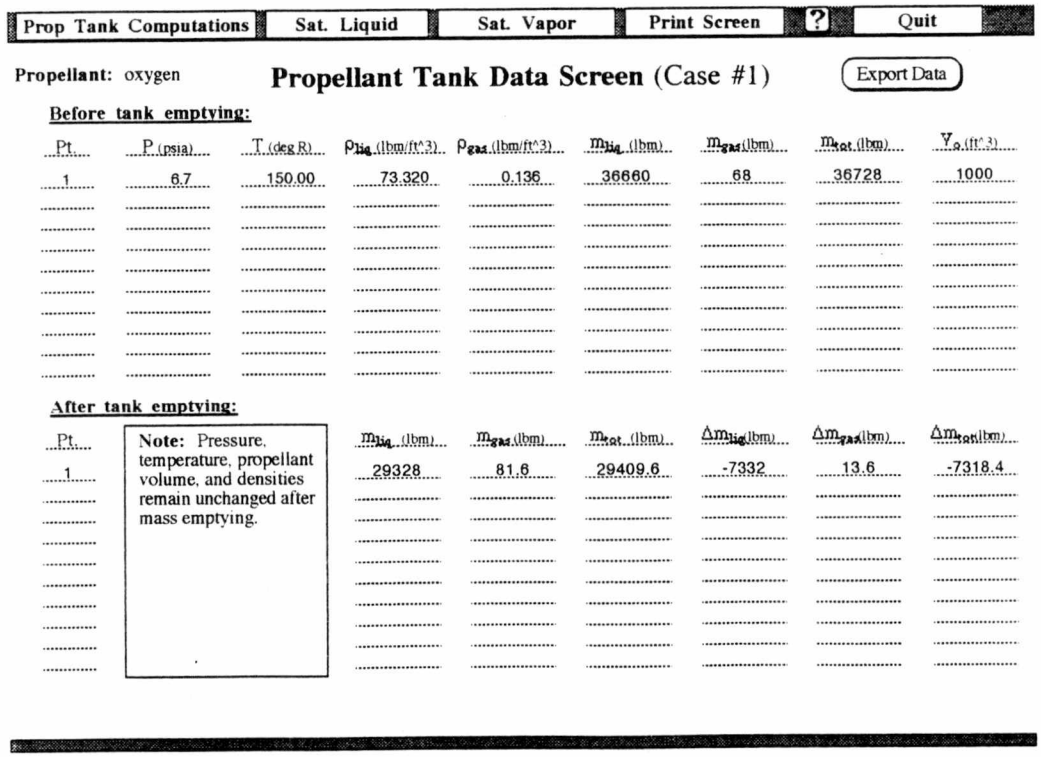


Figure 32. PTPP Data Entry and Results for Sample Calculation

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

The graphical user interface program LRE-CP^2 performs a much needed function of providing thermodynamic property data for use in component performance models for typical rocket and test facility components. It achieves this through a user-friendly, graphic based environment that utilizes extensive thermodynamic data tables provided in a modified version of the MIPROPS program. Thus, the LRE-CP^2 minimizes the amount of detail a user must confront when doing component evaluations. It has the potential to be an excellent tool for propulsion engineers who need to acquire engine component system performance estimates quickly and easily.

The current version of the LRE-CP^2 accomplished all the objectives for such a program that were defined at the beginning of the study, but it does have limitations. The accuracy of any results is restricted by the preciseness of MIPROPS. Uncertainties in the MIPROPS computations should be considered when examining output from the LRE-CP^2. In addition, the requirements on the inputs to the LRE-CP^2 are specific in terms of format. If the output (final) conditions are known, for example, it is not possible under the LRE-CP^2 to determine the input (initial) ones except manually, by trial and error.

Recommendations for the LRE-CP^2 embody several possible modifications. First, the integration of the LRE-CP^2 into an expert design system (EDS) architecture would increase its applicability. As noted above, inputs to the LRE-CP^2 are limited to single values and specific quantities. In

most engineering problems, though, there are constraints to adhere to and ranges instead of discrete numbers for input parameters. The user may not know which parameter(s) to vary, or by how much, to achieve a workable solution. This is where an EDS would be beneficial. A model for developing an EDS based on the LRE-CP^2 can be found in a paper describing the coupling of artificial intelligence with the design of aerodynamic bodies (Tong,1985). The LRE-CP^2 would be analogous to the CFD code discussed by Tong. A flow schematic for this system adapted from his example is shown in Figure 33. A user would only be responsible for entering the requirements , constraints, and an initial guess for the engine component to the system. The evaluation modules would send the input data to the LRE-CP^2 for analysis, determining the inlet and outlet states of the component and its performance parameters based on the initial conditions. The acceptability module would determine if the obtained solution satisfied all specified criteria (requirements, constraints, goals). If not, the redesign module, in conjunction with the requirement relaxation module, would adjust a certain parameter(s) to move towards a viable answer. The process would continue until an acceptable solution is obtained.

The modules in the EDS would operate according to sets of rules written by an expert. These rules, forming a *production system* (Tong,1985), would know, for example, which parameters to vary in order to move towards a right answer (i.e., decrease flow rate by 10 lb/sec, adjust efficiency range by 5%, etc.). Thus, the EDS would relieve a non-expert user from having to ascertain on his own how to proceed towards a correct solution.

Other possible modifications to the LRE-CP^2 would be to increase the number of propellant choices as needed; adding more component stacks (i.e., turbine, exhaust nozzle, combustor, etc.); and incorporating a graphing

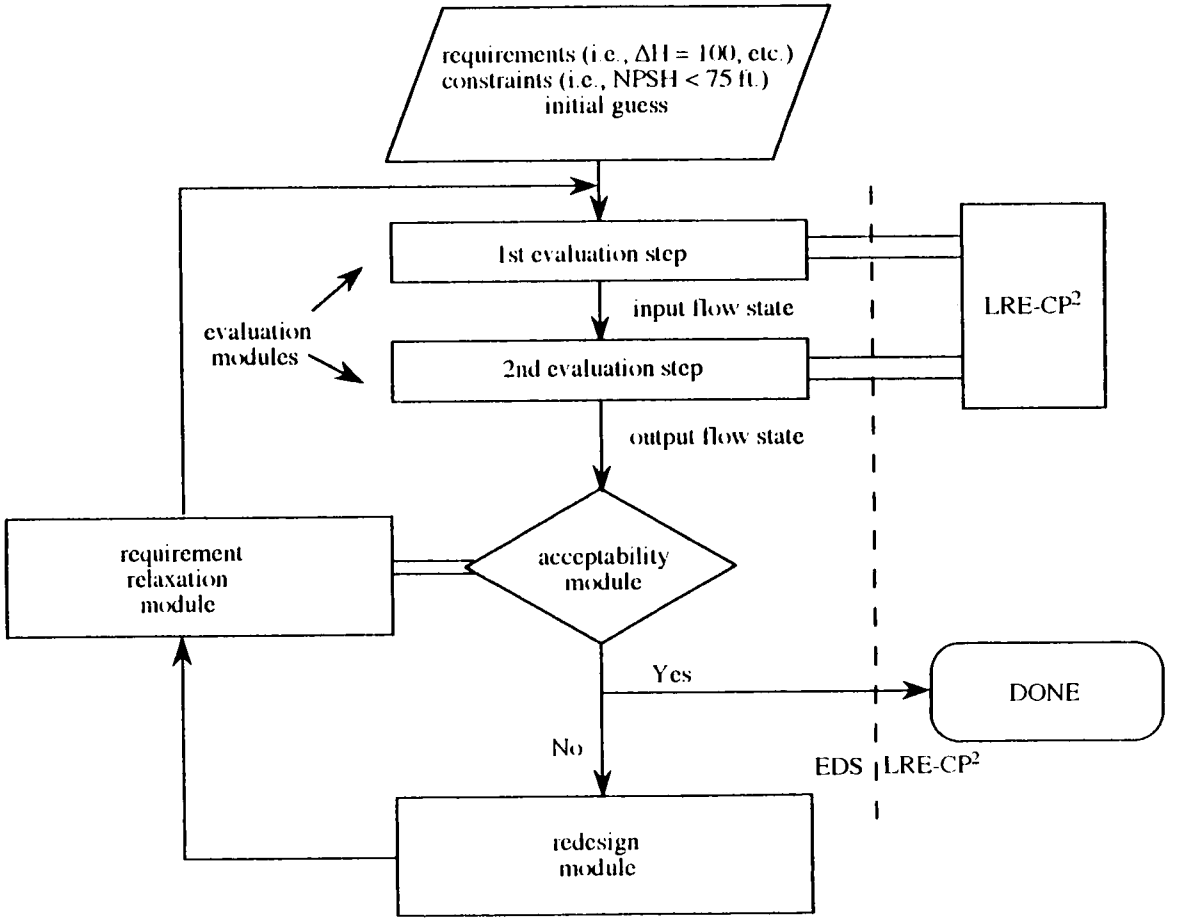


Figure 33. Example of a Flow Schematic for an EDS - LRE-CP^2 System

routine to complement the current tabular form used for the output. Nevertheless, the programming and stacks provided in the present version should allow future developments to be achieved in rapid order, since the integration of Hypertext and Fortran coding to create the graphic interface was the significant contribution of this work.

BIBLIOGRAPHY

BIBLIOGRAPHY

Anonymous, "Configuration, Selection, and Sizing of Butterfly Valves for Precise Pressure Control".

Anon., "Liquid Rocket Valve Assemblies", NASA SP-8097, November 1973.

Absoft Corporation, MacFortran, 1986.

Apple Computer, Inc., MPW (Macintosh Programmer's Workshop Development Environment), Version 3.2, 1991.

Bejan, Adrian, Advanced Engineering Thermodynamics, John Wiley & Sons, New York, 1988.

Benedict, Robert P., Fundamentals of Pipe Flow, John Wiley & Sons, New York, 1980.

Black, William Z. and Hartley, James G., Thermodynamics, Harper and Row, 1985.

Chapra, Steven C. and Canale, Raymond P., Numerical Methods for Engineers, McGraw-Hill Book Company, New York, 1988.

Claris Corporation, Hypercard Reference Manual, 1990.

Claris Corporation, Hypercard Script Language Guide, 1990.

Clegern, James B., "Solid Rocket Motor Conceptual Design: The development of a design optimization expert system with a hypertext user interface", Master's thesis, UTSI, May 1993.

Fessler, Theodore F., "FLUID: A Numerical Interpolation Procedure for Obtaining Thermodynamic and Transport Properties of Fluids", NASA TM X-3572, Lewis Research Center, Cleveland, August 1977.

Huzel, Dieter K. and Huang, David H., "Design of Liquid Propellant Rocket Engines", NASA SP 425, 1971.

McCarty, Robert D., "Interactive FORTRAN Program for Micro Computers to Calculate the Thermophysical Properties of Twelve Fluids (MIPROPS)", NBS Technical Note 1097, May 1986.

Oates, Gordon C., Aerothermodynamics of Gas Turbine and Rocket Propulsion, AIAA Education Series, Washington D.C., 1988.

Spinn, J. Paul, "Digital Transient Model", Version #2, December 1986 (Pratt & Whitney modifications to it).

Sutton, George P., Rocket Propulsion Elements, John Wiley & Sons, New York, 1986.

Tong, Siu Shing, "Design of Aerodynamic Bodies Using Artificial Intelligence/Expert System Technique", AIAA-85-0112.

Tullis, J. Paul, Hydraulics of Pipelines, John Wiley & Sons, New York, 1989.

Walker, Rodger, Pump Selection: A Consulting Engineer's Manual, Associated Engineering Services Ltd., Vancouver, B.C., Canada, 1972.

Younglove, B.A., "Thermophysical properties of fluids.I. Argon, ethylene, parahydrogen, nitrogen, nitrogen trifluoride, and oxygen", Journal of Physical and Chemical Reference Data, Vol. 11, National Bureau of Standards, 1982.

Younglove, B.A. and Ely, J.F., "Thermophysical properties of fluids.II. Methane, ethane, propane, isobutane, and normal butane", Journal of Physical and Chemical Reference Data, Vol. 16, National Bureau of Standards, 1987.

APPENDICES

APPENDIX A

Modified MIPROPS Fortran 77 Code

```
      program miprops
      implicit real*8(a-h)
      implicit real*8(o-z)
      implicit integer*4(i-n)
      common/uname/pna,tna,dna,una,sna,vna,thna,sona
      common/con/pcon,tcon,dcon,ucon,scon,vcon,thcon,socon
      common/crit/em,eok,rm,tc,dc,x,pc,sig
      common/satfl/il,ip
      save /uname/
      save /satfl/
          open(unit=12,file='data in',status='old')
      open(unit=10,file='data out',status='new')
          open(unit=11,file='messages',status='new')
C     ** Set data input option to file default **

      read(12,*) ifl
      read(12,*) iu

      read(12,*) iop
      if(iop.eq.0) then
             read(12,*) il
             read(12,*) ip
      end if
          read(12,*) loop
      do 5 icount=1,loop
            if (iop.eq.0) then
                  if (ip.eq.0) then
                        read(12,*) t
                  else
                        read(12,*) p
                  end if
            else if(iop.eq.1) then
             read(12,*) p,t
            else if(iop.eq.2) then
             read(12,*) p,d
            else if(iop.eq.3) then
             read(12,*) t,d
        else if(iop.eq.4) then
             read(12,*) p,s
                  if(ifl.eq.1) then
                        if(p.gt.17404.8d0) then
                              p=17404.8d0
                        end if
                        if(p.ge.186.2d0) then
                              t=59.29d0
                        else
```

```
                t=24.84d0
        end if
else if(ifl.eq.2) then
        if(p.gt.145040.0d0) then
                p=145040.0d0
        end if
        if(p.ge.493.003d0) then
                t=227.3d0
        else
                t=113.67d0
        end if
else if(ifl.eq.3) then
        if(p.gt.17404.8d0) then
                p=17404.8d0
        end if
        if(p.ge.731.44d0) then
                t=278.25d0
        else
                t=97.85d0
        end if
else if(ifl.eq.4) then
        if(p.gt.14648.8d0) then
                p=14648.8d0
        end if
        if(p.ge.711.52d0) then
                t=271.55d0
        else
                t=150.84d0
        end if
else if(ifl.eq.5) then
        if(p.gt.7251.8d0) then
                p=7251.8d0
        end if
        if(p.ge.646.96d0) then
                t=421.20d0
        else
                t=119.45d0
        end if
else if(ifl.eq.6) then
        if(p.gt.29007.54d0) then
                p=29007.54d0
        end if
        if(p.ge.666.87d0) then
                t=342.95d0
        else
                t=163.22d0
        end if
else if(ifl.eq.7) then
        if(p.gt.10152.64d0) then
                p=10152.64d0
        end if
        if(p.ge.706.54d0) then
                t=549.61d0
        else
```

```
                t=162.63d0
            end if
        else if(ifl.eq.8) then
            if(p.gt.5801.5d0) then
                p=5801.5d0
            end if
            if(p.ge.731.05d0) then
                t=508.22d0
            else
                t=187.17d0
            end if
        else if(ifl.eq.9) then
            if(p.gt.14503.77d0) then
                p=14503.77d0
            end if
            if(p.ge.616.07d0) then
                t=665.73d0
            else
                t=153.85d0
            end if
        else if(ifl.eq.10) then
            if(p.gt.5076.32d0) then
                p=5076.32d0
            end if
            if(p.ge.527.93d0) then
                t=734.13d0
            else
                t=204.39d0
            end if
        else if(ifl.eq.11) then
            if(p.gt.10152.64d0) then
                p=10152.64d0
            end if
            if(p.ge.550.56d0) then
                t=765.29d0
            else
                t=242.75d0
            end if
        end if
    else if(iop.eq.5) then
     read(12,*) d,u
        if(ifl.eq.1) then
            if(d.ge.0.482d0) then
                t=59.29d0
            else
                t=24.84d0
            end if
        else if(ifl.eq.2) then
            if(d.ge.0.0250d0) then
                t=227.3d0
            else
                t=113.67d0
            end if
        else if(ifl.eq.3) then
```

```
        if(d.ge.0.0266d0) then
                t=278.25d0
        else
                t=97.85d0
        end if
else if(ifl.eq.4) then
        if(d.ge.0.0209d0) then
                t=271.55d0
        else
                t=150.84d0
        end if
else if(ifl.eq.5) then
        if(d.ge.0.00696d0) then
                t=421.20d0
        else
                t=119.45d0
        end if
else if(ifl.eq.6) then
        if(d.ge.0.03949d0) then
                t=342.95d0
        else
                t=163.22d0
        end if
else if(ifl.eq.7) then
        if(d.ge.0.0142d0) then
                t=549.61d0
        else
                t=162.63d0
        end if
else if(ifl.eq.8) then
        if(d.ge.0.0170d0) then
                t=508.22d0
        else
                t=187.17d0
        end if
else if(ifl.eq.9) then
        if(d.ge.0.00707d0) then
                t=665.73d0
        else
                t=153.85d0
        end if
else if(ifl.eq.10) then
        if(d.ge.0.00414d0) then
                t=734.13d0
        else
                t=204.39d0
        end if
else if(ifl.eq.11) then
        if(d.ge.0.00421d0) then
                t=765.29d0
        else
                t=242.75d0
        end if
end if
```

```
else if(iop.eq.6) then
 read(12,*) p,h
      if(ifl.eq.1) then
            if(p.gt.17404.8d0) then
                  p=17404.8d0
            end if
            if(p.ge.186.2d0) then
                  t=59.29d0
            else
                  t=24.84d0
            end if
      else if(ifl.eq.2) then
            if(p.gt.145040.0d0) then
                  p=145040.0d0
            end if
            if(p.ge.493.003d0) then
                  t=227.3d0
            else
                  t=113.67d0
            end if
      else if(ifl.eq.3) then
            if(p.gt.17404.8d0) then
                  p=17404.8d0
            end if
            if(p.ge.731.44d0) then
                  t=278.25d0
            else
                  t=97.85d0
            end if
      else if(ifl.eq.4) then
            if(p.gt.14648.8d0) then
                  p=14648.8d0
            end if
            if(p.ge.711.52d0) then
                  t=271.55d0
            else
                  t=150.84d0
            end if
      else if(ifl.eq.5) then
            if(p.gt.7251.8d0) then
                  p=7251.8d0
            end if
            if(p.ge.646.96d0) then
                  t=421.20d0
            else
                  t=119.45d0
            end if
      else if(ifl.eq.6) then
            if(p.gt.29007.54d0) then
                  p=29007.54d0
            end if
            if(p.ge.666.87d0) then
                  t=342.95d0
            else
```

```
                t=163.22d0
            end if
        else if(ifl.eq.7) then
            if(p.gt.10152.64d0) then
                p=10152.64d0
            end if
            if(p.ge.706.54d0) then
                t=549.61d0
            else
                t=162.63d0
            end if
        else if(ifl.eq.8) then
            if(p.gt.5801.5d0) then
                p=5801.5d0
            end if
            if(p.ge.731.05d0) then
                t=508.22d0
            else
                t=187.17d0
            end if
        else if(ifl.eq.9) then
            if(p.gt.14503.77d0) then
                p=14503.77d0
            end if
            if(p.ge.616.07d0) then
                t=665.73d0
            else
                t=153.85d0
            end if
        else if(ifl.eq.10) then
            if(p.gt.5076.32d0) then
                p=5076.32d0
            end if
            if(p.ge.527.93d0) then
                t=734.13d0
            else
                t=204.39d0
            end if
        else if(ifl.eq.11) then
            if(p.gt.10152.64d0) then
                p=10152.64d0
            end if
            if(p.ge.550.56d0) then
                t=765.29d0
            else
                t=242.75d0
            end if
        end if
    else if(iop.eq.7) then
     read(12,*) p,u
        if(ifl.eq.1) then
            if(p.gt.17404.8d0) then
                p=17404.8d0
            end if
```

```
        if(p.ge.186.2d0) then
                t=59.29d0
        else
                t=24.84d0
        end if
else if(ifl.eq.2) then
        if(p.gt.145040.0d0) then
                p=145040.0d0
        end if
        if(p.ge.493.003d0) then
                t=227.3d0
        else
                t=113.67d0
        end if
else if(ifl.eq.3) then
        if(p.gt.17404.8d0) then
                p=17404.8d0
        end if
        if(p.ge.731.44d0) then
                t=278.25d0
        else
                t=97.85d0
        end if
else if(ifl.eq.4) then
        if(p.gt.14648.8d0) then
                p=14648.8d0
        end if
        if(p.ge.711.52d0) then
                t=271.55d0
        else
                t=150.84d0
        end if
else if(ifl.eq.5) then
        if(p.gt.7251.8d0) then
                p=7251.8d0
        end if
        if(p.ge.646.96d0) then
                t=421.20d0
        else
                t=119.45d0
        end if
else if(ifl.eq.6) then
        if(p.gt.29007.54d0) then
                p=29007.54d0
        end if
        if(p.ge.666.87d0) then
                t=342.95d0
        else
                t=163.22d0
        end if
else if(ifl.eq.7) then
        if(p.gt.10152.64d0) then
                p=10152.64d0
        end if
```

```
                    if(p.ge.706.54d0) then
                        t=549.61d0
                    else
                        t=162.63d0
                    end if
                else if(ifl.eq.8) then
                    if(p.gt.5801.5d0) then
                        p=5801.5d0
                    end if
                    if(p.ge.731.05d0) then
                        t=508.22d0
                    else
                        t=187.17d0
                    end if
                else if(ifl.eq.9) then
                    if(p.gt.14503.77d0) then
                        p=14503.77d0
                    end if
                    if(p.ge.616.07d0) then
                        t=665.73d0
                    else
                        t=153.85d0
                    end if
                else if(ifl.eq.10) then
                    if(p.gt.5076.32d0) then
                        p=5076.32d0
                    end if
                    if(p.ge.527.93d0) then
                        t=734.13d0
                    else
                        t=204.39d0
                    end if
                else if(ifl.eq.11) then
                    if(p.gt.10152.64d0) then
                        p=10152.64d0
                    end if
                    if(p.ge.550.56d0) then
                        t=765.29d0
                    else
                        t=242.75d0
                    end if
                end if
            end if
        call fprop(ifl,iop,iu,p,t,d,u,h,s,cp,cv,v,th,so,di,icount)
C     *-- rg=cp-cv --*
        rg=1545.4d0/em
        cz=144.d0
            rto=cp/cv
            write (10,7) p
            write (10,8) t
            write (10,9) d
            write (10,7) u
            write (10,7) h
            write (10,9) s
```

```
          write (10,9) rto
  7   format (f10.1)
  8   format (f10.2)
  9   format (f10.3)
  5 continue
    end
*************************************************************************
    subroutine fprop(ifl,iop,iu,p,t,d,u,h,s,cp,cv,v,th,so,di,icount)
    implicit real*8(a-h)
    implicit real*8(o-z)
    implicit integer*4(i-n)
    common/data/g(32),r,gam,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
    common/sen/beta,xo,delta,e1,e2,agam
    common/crit/em,eok,rm,tc,dc,x,pc,sig
    common/satc/a(20),dtpv,eg
    common/cpid/gi(11),gh(11),gl(11)
    common/data1/gv(9),gt(9),fv(4),ft(4),ev(8),et(8)
    common/isp/n,nw,nww
    common/fixpt/t0,s0,h0
    common/diel/bx(6),px(6)
    common/b/gmw,seok,ssig
    common/a/se1,g1,b1,de,bk,d1,xz,zz,x1,x2,x3,x4
    common/con/pcon,tcon,dcon,ucon,scon,vcon,thcon,socon
    common/satfl/il,ip
    save /con/
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++
    if(ifl.eq.1) then
      call hydr
    else if(ifl.eq.2) then
      call nitr
    else if(ifl.eq.3) then
      call oxyg
    else if(ifl.eq.4) then
          call argo
        else if(ifl.eq.5) then
          call ntrf
        else if(ifl.eq.6) then
          call meth
        else if(ifl.eq.7) then
          call c2h6
        else if(ifl.eq.8) then
          call c2h4
        else if(ifl.eq.9) then
          call c3h8
        else if(ifl.eq.10) then
          call isob
        else if(ifl.eq.11) then
          call norb
    end if
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++
    if(iu.eq.0) then
      pcon=14.695949d0/0.101325d0
```

```
          tcon=1.8d0
          dcon=em/16.0184637d0
        ucon=1.d0/(2.324445d0*em)
          scon=1.d0/(4.184001d0*em)
        vcon=1.d-1*0.671875652d0
          thcon=0.578176d0
          socon=3.28084d0
      else
        pcon=1.d0
          tcon=1.d0
          dcon=1.d0
          ucon=1.d0
        scon=1.d0
        vcon=1.d0
          thcon=1.d0
          socon=1.d0
      endif
      pp=p/pcon
      tt=t/tcon
      dd=d/dcon
      uu=u/ucon
      hh=h/ucon
      ss=s/scon
      cpp=cp/scon
      cvv=cv/scon
      soo=so/socon
      vv=v/vcon
      thh=th/thcon
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++
      if(iop.eq.0) then
        call satur(pp,tt,ip,icount)
        if(il.eq.0) then
          dd=findd(pp+1.d-4,tt,icount)
        else
          dd=findd(pp-1.d-4,tt,icount)
        endif
        call energy(pp,tt,dd,uu,hh,ss)
        call spheat(tt,dd,cpp,cvv,soo)
        di=diel(pp,tt,dd)
        if(ifl.ne.1) then
          call visthe(dd,tt,vv,thh)
        else
          vv=0.d0
                thh=0.d0
        endif
      else if(iop.eq.1) then
        call limits(pp,tt,icount)
        dd=findd(pp,tt,icount)
        call energy(pp,tt,dd,uu,hh,ss)
        call spheat(tt,dd,cpp,cvv,soo)
        di=diel(pp,tt,dd)
        if(ifl.ne.1) then
          call visthe(dd,tt,vv,thh)
```

```
  else
    vv=0.d0
            thh=0.d0
  endif
else if(iop.eq.2) then
  tt=findt(pp,dd,icount)
  call limits(pp,tt,icount)
  call energy(pp,tt,dd,uu,hh,ss)
  call spheat(tt,dd,cpp,cvv,soo)
  di=diel(pp,tt,dd)
  if(ifl.ne.1) then
    call visthe(dd,tt,vv,thh)
  else
    v=0.d0
            th=0.d0
  endif
else if(iop.eq.3) then
  pp=findp(tt,dd,icount)
  call limits(pp,tt,icount)
  call energy(pp,tt,dd,uu,hh,ss)
  call spheat(tt,dd,cpp,cvv,soo)
  di=diel(pp,tt,dd)
  if(ifl.ne.1) then
    call visthe(dd,tt,vv,thh)
  else
    vv=0.d0
              thh=0.d0
  endif
else if(iop.eq.4) then
  si=ss
       isig=0
  do while(isig.lt.4)
    dd=findd(pp,tt,icount)
    call energy(pp,tt,dd,uu,hh,ss)
    call nuiter(tt,(ss-si),1.d-3,1.d-7,50,np,isig,2)
  enddo
  if(isig.eq.4) write(11,*) 'FPROP - Option 4 Failed to Converge'
  dd=findd(pp,tt,icount)
  call limits(pp,tt,icount)
  call energy(pp,tt,dd,uu,hh,ss)
  call spheat(tt,dd,cpp,cvv,soo)
  di=diel(pp,tt,dd)
  if(ifl.ne.1) then
    call visthe(dd,tt,vv,thh)
  else
    vv=0.d0
         thh=0.d0
  endif
else if(iop.eq.5) then
  ui=uu
       isig=0
  do while(isig.lt.4)
    pp=findp(tt,dd,icount)
    call energy(pp,tt,dd,uu,hh,ss)
```

```
      call nuiter(tt,(uu-ui),1.d-3,1.d-7,50,np,isig,2)
    enddo
    if(isig.eq.4) write(11,*) 'FPROP - Option 5 Failed to Converge'
    pp=findp(tt,dd,icount)
    call limits(pp,tt,icount)
    call energy(pp,tt,dd,uu,hh,ss)
    call spheat(tt,dd,cpp,cvv,soo)
    di=diel(pp,tt,dd)
    if(ifl.ne.1) then
      call visthe(dd,tt,vv,thh)
    else
      vv=0.d0
                  thh=0.d0
    endif
  else if(iop.eq.6) then
    hi=hh
          isig=0
    do while(isig.lt.4)
      dd=findd(pp,tt,icount)
      call energy(pp,tt,dd,uu,hh,ss)
      call nuiter(tt,(hh-hi),1.d-3,1.d-7,50,np,isig,2)
    enddo
    if(isig.eq.4) write(11,*) 'FPROP - Option 6 Failed to Converge'
    dd=findd(pp,tt,icount)
    call limits(pp,tt,icount)
    call energy(pp,tt,dd,uu,hh,ss)
    call spheat(tt,dd,cpp,cvv,soo)
    di=diel(pp,tt,dd)
    if(ifl.ne.1) then
      call visthe(dd,tt,vv,thh)
    else
      vv=0.d0
              thh=0.d0
    endif
  else if(iop.eq.7) then
    ui=uu
        isig=0
    do while(isig.lt.4)
      dd=findd(pp,tt,icount)
      call energy(pp,tt,dd,uu,hh,ss)
      call nuiter(tt,(uu-ui),1.d-3,1.d-7,50,np,isig,2)
    enddo
    if(isig.eq.4) write(*,*) 'FPROP - Option 7 Failed to Converge'
    dd=findd(pp,tt,icount)
    call limits(pp,tt,icount)
    call energy(pp,tt,dd,uu,hh,ss)
    call spheat(tt,dd,cpp,cvv,soo)
    di=diel(pp,tt,dd)
    if(ifl.ne.1) then
      call visthe(dd,tt,vv,thh)
    else
      vv=0.d0
              thh=0.d0
    endif
```

```
      else
        pp=0.d0
                  tt=0.d0
                  dd=0.d0
                  uu=0.d0
                  hh=0.d0
                  ss=0.d0
                  cpp=0.d0
        cvv=0.d0
                  vv=0.d0
                  thh=0.d0
                  soo=0.d0
                  di=0.d0
      endif
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++
      p=pp*pcon
      t=tt*tcon
      d=dd*dcon
      u=uu*ucon
      h=hh*ucon
      s=ss*scon
      cp=cpp*scon
      cv=cvv*scon
      so=soo*socon
      v=vv*vcon
      th=thh*thcon
      return
      end
*************************************************************************
      subroutine limits(p,t,icount)
      implicit real*8(a-h)
      implicit real*8(o-z)
      implicit integer*4(i-n)
      common/data/g(32),r,gam,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
      common/con/pcon,tcon,dcon,ucon,scon,vcon,thcon,socon
      common/isp/n,nw,nww
      common/diel/bx(6),px(6)
      if(p.gt.pul) then
        p=pul
        write(11,*)
            write(11,*) 'Data Point',icount
        write(11,*) 'The Input Pressure is Beyond the Limit'
        write(11,*) 'The Pressure returned is',p*pcon,'psia'
        write(11,*)
      endif
      if(t.gt.tul.or.t.lt.tll) then
        t=dmax1(tll,dmin1(tul,t))
        write(11,*)
            write(11,*) 'Data Point',icount
        write(11,*) 'The Input Temperature is Beyond the Limit'
        write(11,*) 'The Temperature returned is',t*tcon,'deg R'
        write(11,*)
      endif
```

```
      if(n.eq.1.and.p.lt.31.64d0) then
        tmelt=((p-px(4))/px(5))**(1.d0/px(6))
      else
        tmelt=((p-px(1))/px(2))**(1.d0/px(3))
      endif
      if(n.eq.1.and.t.ge.22.d0) then
        pmelt=px(4)+px(5)*t**px(6)
      else
        pmelt=px(1)+px(2)*t**px(3)
      endif
      if(p.gt.pmelt) then
        t=tmelt
        write(11,*)
            write(11,*) 'Data Point',icount
        write(11,*) 'Solid Phase Detected for this Pressure'
        write(11,*) 'The Temperature returned is',t*tcon,'deg R'
        write(11,*)
      endif
      return
      end
*************************************************************************
      real*8 function findd(p,t,icount)
      implicit real*8(a-h)
      implicit real*8(o-z)
      implicit integer*4(i-n)
      common/data/g(32),r,gam,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
      common/con/pcon,tcon,dcon,ucon,scon,vcon,thcon,socon
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
      x(z)=(1.d0-vp(7)/z)/(1.d0-vp(7)/vp(8))
      vpn(y)=vp(9)*dexp(vp(1)*x(y)+vp(2)*x(y)**2.d0+vp(3)*x(y)**3.d0
     & +vp(4)*x(y)**4.d0+vp(5)*x(y)*(1.d0-x(y))**vp(6))
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
      if(t.ge.tcc) then
        dd=1.0/(r*t*(1.1/pcc+0.7/p)/9.0)
        if((p/pcc).gt.2.d0.and.t/tcc.lt.2.5d0) dd=dtp
      else if(p.gt.vpn(t)) then
        dd=satl(t)
      else
        dd=satv(t)
      endif
   10 din=dd
      isig=0
      do while(isig.lt.4)
        call props(pp,dd,t,1)
        call nuiter(dd,(p-pp),1.d-3,1.d-7,50,np,isig,1)
      enddo
      if(isig.eq.4) then
        write(11,*)
            write(11,*) 'Data Point',icount
        write(11,*) 'FINDD - Failed to converge'
        write(11,*) '    Press =',p*pcon,'psia'
        write(11,*) '    Temp  =',t*tcon,'deg R'
```

```
      write(11,*) '    Dens =',dd*dcon,'lbm/ft^3'
          write(11,*) '              Error =',(p-pp)*pcon

    endif
    if(t.lt.tcc) then
      if(din/dd.lt.0.85d0) then
        dd=din+1
        go to 10
      endif
    else
      if((dd.gt.50.d0.and.din/dd.lt.0.85d0).or.dd.lt.0.d0) then
        dd=din+1
        go to 10
      endif
    endif
    findd=dd
    return
    end
*************************************************************************
    real*8 function findp(t,d,icount)
    implicit real*8(a-h)
    implicit real*8(o-z)
    implicit integer*4(i-n)
    common/data/g(32),r,gam,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
    x(z)=(1.d0-vp(7)/z)/(1.d0-vp(7)/vp(8))
    vpn(y)=vp(9)*dexp(vp(1)*x(y)+vp(2)*x(y)**2.d0+vp(3)*x(y)**3.d0
   & +vp(4)*x(y)**4.d0+vp(5)*x(y)*(1.d0-x(y))**vp(6))
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
    if(t.lt.tcc) then
      p=vpn(t)
      dv=findd(p-1.d-4,t,icount)
      dl=findd(p+1.d-4,t,icount)
      if(d.gt.dv.and.d.lt.dl) then
        d=dv
        write(11,*)
              write(11,*) 'Data Point',icount
        write(11,*) 'FINDP - The State Point Specified Corresponds'
        write(11,*) '       to a Density in the 2-Phase Region'
        write(11,*) '       SATURATED VAPOR IS ASSUMED'
        write(11,*)
      endif
    endif
    call props(pp,d,t,1)
    findp=pp
    return
    end
*************************************************************************
    real*8 function findt(p,d,icount)
    implicit real*8(a-h)
    implicit real*8(o-z)
    implicit integer*4(i-n)
```

```
      common/data/g(32),r,gam,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
      common/con/pcon,tcon,dcon,ucon,scon,vcon,thcon,socon
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
      x(z)=(1.d0-vp(7)/z)/(1.d0-vp(7)/vp(8))
      vpn(y)=vp(9)*dexp(vp(1)*x(y)+vp(2)*x(y)**2.d0+vp(3)*x(y)**3.d0
     & +vp(4)*x(y)**4.d0+vp(5)*x(y)*(dabs(1.d0-x(y)))**vp(6))
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
C  RETURNS TEMPERATURE(K), FROM THE 32-TERM MBWR EQN OF STATE.
C  INPUT IS PRESSURE(MPA) AND DENSITY(MOL/L).
      if(p.ge.pcc) then
        tt=tcc
      else
        tsat=tcc
        isig=0
        do while(isig.lt.4)
          pp=vpn(tsat)
          call nuiter(tsat,(p-pp),1.d-3,1.d-7,50,np,isig,1)
        enddo
        if(isig.eq.4) then
          write(11,*)
                write(11,*) 'Data Point',icount
          write(11,*) 'FINDT - Vapor Pressure Equ. Failed to Converge'
          write(11,*) '    Press =',p*pcon,'psia'
                write(11,*)      '    Tsat =',tsat*tcon,'deg R'
          write(11,*) '    Error =',(p-pp)*pcon
          write(11,*)
        endif
        dv=findd(p-1.d-4,tsat,icount)
        dl=findd(p+1.d-4,tsat,icount)
        if(d.gt.dv.and.d.lt.dl) then
          write(11,*)
                write(11,*) 'Data Point',icount
          write(11,*) 'FINDT - The State Point Specified Corresponds'
          write(11,*) '        to a Density in the 2-Phase Region'
          write(11,*)
          findt=tsat
          return
        endif
        tt=tsat
      endif
      isig=O
      do while(isig.lt.4)
        call props(pp,d,tt,1)
        call nuiter(tt,(p-pp),1.d-3,1.d-7,50,np,isig,1)
      enddo
      if(isig.eq.4) then
        write(11,*)
            write(11,*) 'Data Point',icount
        write(11,*) 'FINDT - Equation of State Failed to Converge'
        write(11,*) '    Press =',p*pcon,'psia'
            write(11,*) '    Temp =',t*tcon,'deg R'
        write(11,*) '    Error =',(p-pp)*pcon
```

```
      write(11,*)
    endif
    findt=tt
    return
    end
*************************************************************************
    real*8 function satv(tt)
    implicit real*8(a-h)
    implicit real*8(o-z)
    implicit integer*4(I-N)
    common/data/g(32),r,gam,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
    common/satc/a(20),dtpv,eg
    common/isp/n,nw,nww
    t=tt
    if(nw.eq.1) then
      if(t.lt.tcc) then
        xv=(tcc-t)/(tcc-ttp)
        yv=a(1)*tcc*xv/t+a(2)*xv**eg+a(3)*xv+a(4)*xv**(4.d0/3.d0)
     &      +a(5)*xv**(5.d0/3.d0)+a(6)*xv**2.d0
        satv=dcc*((dtpv/dcc)**yv)
      else
        satv=dcc
      endif
    else
      if(t.ge.tcc) then
        satv=dcc
      else
        itt=idint(tcc)
        if((itt+1-idint(t)).lt.1) t=dble(itt)
        xr=(tcc-t)/(tcc-ttp)
        dv=a(1)*dlog(xr)
        do 10 i=2,13
          mm=i+max0(0,isign(1,(i-5)))
          dv=dv+a(i)*(1.d0-xr**(dble(mm-5)/3.d0))
 10       continue
        dv=dcc+dexp(dv)*(dtpv-dcc)
        satv=dv-(dv-dcc)*(tt-t)
      endif
    endif
    return
    end
*************************************************************************
    real*8 function satl(tt)
    implicit real*8(a-h)
    implicit real*8(o-z)
    implicit integer*4(I-N)
    common/data/g(32),r,gam,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
    common/satc/a(20),dtpv,eg
    common/isp/n,nw,nww
    t=tt
    if(nw.eq.1) then
      if(t.lt.tcc) then
        x1=(tcc-t)/(tcc-ttp)
        yl=a(7)*dlog(x1)+a(8)*(1.d0-1.d0/x1)
```

```
     &       +a(9)*(1.d0-x1**(-2.d0/3.d0))
     &       +a(10)*(1.d0-x1**(-1.d0/3.d0))
     &       +a(11)*(1.d0-x1**(1.d0/3.d0))
     &       +a(12)*(1.d0-x1**(2.d0/3.d0))
     &       +a(13)*(1.d0-x1)
           satl=dcc+(dtp-dcc)*dexp(yl)
         else
           satl=dcc
         endif
       else
         if(t.ge.tcc) then
           satl=dcc
         else
           itt=idint(tcc)
           if((itt+1-idint(t)).lt.1) t=dble(itt)
           xr=(tcc-t)/(tcc-ttp)
           dl=a(14)*dlog(xr)
           do 10 i=2,7
             mm=i+max0(0,isign(1,(i-5)))
             dl=dl+a(i+13)*(1.d0-xr**(dble(mm-5)/3.d0))
    10     continue
           dl=dcc+dexp(dl)*(dtp-dcc)
           satl=dl-(dl-dcc)*(tt-t)
         endif
       endif
       return
       end
*************************************************************************
       subroutine satur(p,t,iop,icount)
       implicit real*8(a-h)
       implicit real*8(o-z)
       implicit integer*4(i-n)
       common/data/g(32),r,gam,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
       common/con/pcon,tcon,dcon,ucon,scon,vcon,thcon,socon
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
       x(z)=(1.d0-vp(7)/z)/(1.d0-vp(7)/vp(8))
       vpn(y)=vp(9)*dexp(vp(1)*x(y)+vp(2)*x(y)**2.d0+vp(3)*x(y)**3.d0
      & +vp(4)*x(y)**4.d0+vp(5)*x(y)*(dabs(1.d0-x(y)))**vp(6))
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
       if(iop.eq.0) then
         if(t.gt.tcc.or.t.lt.ttp) then
           write(11,*)
                   write(11,*) 'Data Point',icount
           write(11,*) 'SATUR - The Input Temperature is outside the'
           write(11,*) '        Range of Saturation Temperatures'
           write(11,*) '        The Range is ',ttp*tcon,'deg R','to',tcc*tcon,'deg R'
           write(11,*)
           return
         endif
         p=vpn(t)
       else
         if(p.gt.pcc.or.p.lt.ptp) then
```

```
        write(11,*)
              write(11,*) 'Data Point',icount
        write(11,*) 'SATUR - The Input Pressure is outside the'
        write(11,*) '       Range of Saturation Pressures'
        write(11,*) '       The Range is ',ptp*pcon,'psia','to',pcc*pcon,'psia'
        write(11,*)
        return
      endif
      t=tcc
      isig=0
      do while(isig.lt.4)
        psat=vpn(t)
        call nuiter(t,(p-psat),1.d-3,1.d-7,50,np,isig,1)
      enddo
      if(isig.eq.4) then
        write(11,*)
              write(11,*) 'Data Point',icount
        write(11,*) 'SATUR - Vapor Press Equation Failed to converge'
        write(11,*) '    Press =',psat*pcon,'psia'
              write(11,*)     '    Temp =',t*tcon,'deg R'
        write(11,*) '    Error =',(p-psat)*pcon
        write(11,*)
      endif
    endif
    return
    end
*************************************************************************
    subroutine energy(p,t,d,u,h,s)
    implicit real*8(a-h)
    implicit real*8(o-z)
    implicit integer*4(i-n)
    common/data/g(32),r,gam,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
    p0=0.101325d0
    call props(sd,d,t,4)
    call props(ud,d,t,5)
    call props(s0,0.d0,t,4)
    call props(u0,0.d0,t,5)
    s=((sd-s0)-r*dlog(d*r*t/p0))*1.d+3+cpi(t,2)
    u=(t*(sd-s0)+(ud-u0)-r*t)*1.d+3+cpi(t,3)
    h=u+p/d*1.d+3
    return
    end
*************************************************************************
    subroutine spheat(t,d,cp,cv,so)
    implicit real*8(a-h)
    implicit real*8(o-z)
    implicit integer*4(i-n)
    common/data/g(32),r,gam,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
    common/crit/em,eok,rm,tc,dc,x,pc,sig
    call props(cd,d,t,6)
    call props(c0,0.d0,t,6)
    cv=cpi(t,1)+(c0-cd-r)*1.d+3
    call props(dpt,d,t,3)
    call props(dpd,d,t,2)
```

```
      cp=cv+(t/dpd*((dpt/d)**2.d0))*1.d+3
      so=dsqrt(cp/cv/em*dpd*1.d+6)
      return
      end
*****************************************************************************
      real*8 function diel(p,t,d)
      implicit real*8(a-h)
      implicit real*8(o-z)
      implicit integer*4(I-N)
C          DIELECTRIC CONSTANT.  INPUT P(MPA), D(MOL/L) AND T(K).
      common/diel/bx(6),px(6)
      common/data/g(32),r,gam,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
      common/isp/n,nw,nww
      if(nw.eq.1) then
        cm=bx(1)+bx(2)*d+bx(3)*d**2.d0+bx(4)*dlog(1.d0+tcc/t)+bx(5)*p
      else
        cm=bx(1)+bx(2)*d+bx(3)*d**2.d0+bx(4)*d**3+bx(5)*p+bx(6)*t
      endif
      diel=(1.d0+2.d0*d*cm)/(1.d0-d*cm)
      return
      end
*****************************************************************************
      subroutine visthe(dd,t,vis,ther)
      implicit real*8(a-h)
      implicit real*8(o-z)
      implicit integer*4(i-n)
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++
c
c    calculates viscosity in (micro pa - sec)
c         and thermal conductivity in (mega watt/m - k)
c
c    input - Density in (mol/l) and temperature in (k)
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++
      common/data/g(32),r,gam,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
      common/data1/gv(9),gt(9),fv(4),ft(4),ev(8),et(8)
      common/crit/em,eok,rm,tc,dc,x,pc,sig
      common/isp/n,nw,nww
      common/b/gmw,seok,ssig
      common/sen/beta,xo,delta,e1,e2,agam
      common/a/se1,g1,b1,de,bk,d1,xz,zz,x1,x2,x3,x4
      dimension c0(9)
      data c0/-3.0328138281d+00, 1.6918880086d+01,-3.7189364917d+01,
     *       4.1288861858d+01,-2.4615921140d+01, 8.9488430959d+00,
     *      -1.8739245042d+00, 2.0966101390d-01,-9.6570437074d-03/
      if(nw.eq.1) then
        e0=0.d0
        do 10 j=1,9
          e0=e0+c0(j)*(t/seok)**(dble(j-4)/3.d0)
   10   continue
        eta0=2.6693d0*dsqrt(gmw*t)/(ssig*ssig/e0)
        eta1=fv(1)+fv(2)*(fv(3)-dlog(t/fv(4)))**2.d0
```

```
      eta=eta0+eta1*dd
      trv1=ev(3)+ev(4)*t**(-1.5d0)
      trv2=ev(5)+ev(6)/t+ev(7)*t**(-2.d0)
      trvx=dexp(ev(1)+ev(2)/t)
      r1=dd**0.1d0
      r2=((dd-dcc)/dcc)*dsqrt(dd)
      vis=trvx*(dexp(trv1*r1+trv2*r2)-1.d0)+eta
c
      trt0=et(1)+et(2)/t+et(3)/(t*t)
      trt1=et(4)+et(5)/t+et(6)/(t*t)
      trt2=et(7)+et(8)/t
      con1=8.31434d0*15.d0/4.d0
      con2=con1*2.d0/3.d0
      cp0=cpi(t,1)
      yc=(gt(1)+gt(2)*seok/t)*(cp0-con2)
      backg=eta0*1.d-3*(con1+yc)/gmw+(trt0+trt1*dd)*dd/(1.d0+trt2*dd)
      deld=dabs(dd-dcc)/dcc
      delt=dabs(t-tcc)/tcc
      fact=dmin1(100.d0,(x1*delt**4.d0+x2*deld**4.d0))
      call props(dpd,dd,t,2)
      call props(dpt,dd,t,3)
      dpd=dpd*1.d+6
        dpt=dpt*1.d+6
        pc1=pcc*1.d+6
      if(deld.eq.0) then
        bgam=xz**g1/d1*((1.d0+e1)/e1)**((g1-1.d0)/(2.d0*b1))
        chistar=bgam*delt**(-g1)
      else if(deld.le.0.25d0.and.delt.lt.0.03d0) then
        y=((delt/deld**(1.d0/b1))+xz)/xz
        top=deld**(-g1/b1)*((1.d0+e1)/(1.d0+e1*y**(2.d0*b1)))**(
     &        (g1-1.d0)/(2.d0*b1))
        div=d1*(de+(y-1.d0)*(de-1.d0/b1+e1*y**(2.d0*b1))/(1.d0+e1*
     &        y**(2.d0*b1)))
        chistar=top/div
      else
        chistar=pc1*dd/(dcc**2.d0*dpd)
      endif
      upper=x4*bk/pc1*(t*dpt*dcc/dd)**2.d0*(chistar**x3)*dexp(-fact)
      tcrit=upper/(zz*6.d0*3.1415926535898d0*vis*1.d-6)
      ther=backg+tcrit
    else
      d=dd*em*1.d-3
      dilv=0.d0
      do 15 i=1,9
        dilv=dilv+gv(i)*t**((dble(i)-4.d0)/3.d0)
15    continue
      fdcv=(fv(1)+fv(2)*(dabs(fv(3)-dlog(t/fv(4))))**2.d0)*d
      ra=d**0.1d0
      rb=dsqrt(d)*((d-ev(8))/ev(8))
      xv1=ev(1)+ev(6)/t
      xv2=xv1+ev(2)*rb+ev(3)*ra+ev(4)*rb/(t*t)+ev(5)*ra/t**1.5d0+
     &      ev(7)*rb/t
      excesv=dexp(xv2)-dexp(xv1)
      vis=dilv+fdcv+excesv
```

```
c
      av=6.0225d+23
      bkc=1.38054d-16
      deld=dabs(d-dc)/dc
      delt=dabs(t-tc)/tc
      rc=dsqrt(rm**5.d0*d*av/em*eok*x**2.d0/t)
      call props(dpt,dd,t,3)
      call props(dpd,dd,t,2)
      dpt=dpt*1.d+7
      dpd=dpd/em*1.d+10
      if(dpd.lt.0.d0) dpd=1.d0
      visc=vis*1.d-5
      if(deld.gt.0.25d0.or.delt.gt.0.025d0) then
        compres=(d*dpd)**(-0.5d0)
      else
        xx=delt/deld**(1.d0/beta)
        agb=(agam-1.d0)/(2.d0*beta)
        agbb=agb-1.d0
        xxo=(xx+xo)/xo
        h=e1*xxo*(1.d0+e2*xxo**(2.d0*beta))**agb
        hpr=h/(xx+xo)+(agam-1.d0)/xo*e1*e2*xxo**(2.d0*beta)*(1.d0+
     &       e2*xxo**(2.d0*beta))**agbb
        rcom=deld**(delta-1.d0)*(delta*h-xx/beta*hpr)*(d/dc)**2.d0
        compres=(rcom*pc*1.d+7)**(-0.5d0)
      endif
      ex=bkc*(t*dpt)**2.d0*compres
      exb=rc*dsqrt(bkc*t*d*av/em)
      crit=ex/(exb*6.d0*3.1415926535898d0*visc)
      bdd=deld**4.d0
      btt=delt**2.d0
      bxx=dmax1(-1.d+2,(-18.66d0*btt-4.25d0*bdd))
      critc=crit*dexp(bxx)*1.d-5
      dilt=0.d0
      do 20 i=1,9
        dilt=dilt+gt(i)*t**((dble(i)-4.d0)/3.d0)
 20   continue
      fdct=(ft(1)+ft(2)*(dabs(ft(3)-dlog(t/ft(4))))**2.d0)*d
      rb=dsqrt(d)*((d-et(8))/et(8))
      xt1=et(1)+et(6)/t
      xt2=xt1+et(2)*rb+et(3)*ra+et(4)*rb/(t*t)+et(5)*ra/t**1.5d0+
     &     et(7)*rb/t
      excest=dexp(xt2)-dexp(xt1)
      ther=dilt+fdct+excest+critc
    endif
    return
    end
*************************************************************************
    subroutine nuiter(x,y,xtol,ytol,mxp,np,is,il)
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++
c    This Routine is an iteration technique known as
c        the Newton Method
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++
```

```
      implicit real*8(a-h)
      implicit real*8(o-z)
      implicit integer*4(i-n)
      common/save/xn(9),yn(9),ya(9),yb(9),xmin(9),ymin(9)
      save /save/
      if(is.gt.3) return
      if(dabs(y).le.ytol) then
        is=5
        return
      endif
      if(is.eq.0) np=1
      if(is.le.1) then
        if(dabs(y).lt.dabs(yn(il)).or.is.eq.0) then
          xmin(il)=x
          ymin(il)=y
        endif
        xn(il)=x
        yn(il)=y
        is=2
        x=xn(il)*(1.d0+xtol)
        return
      endif
      if(is.eq.2) then
        yb(il)=y
        is=3
        x=xn(il)*(1.d0-xtol)
        return
      endif
      ya(il)=y
      if((yb(il)-ya(il)).eq.0.d0.or.np.gt.mxp) then
        is=4
        x=xmin(il)
        y=ymin(il)
        return
      endif
      x=xn(il)*(yb(il)-ya(il)-2.d0*xtol*yn(il))/(yb(il)-ya(il))
      is=1
      np=np+1
      return
      end
*****************************************************************************
      subroutine props(pp,dd,tt,k)
      implicit real*8(a-h)
      implicit real*8(o-z)
      implicit integer*4(i-n)
C THE 32 TERM EQUATION OF STATE, INPUT IS DENSITY(MOLES/L),
C TEMPERATURE(K), OUTPUT (PP) IS PRESSURE(MPA),OR DP/DD IN
C LITER-MPA/MOLE OR DP/DT MPA/K OR S,H,OR CV AT ONE LIMIT OF
C INTEGRATION
      dimension b(33)
      common/data/g(32),r,gm,vp(9),dtp,pcc,ptp,tcc,ttp,tul,tll,pul,dcc
      d=dd
      p=pp
      t=tt
```

```
      d2=d*d
      d3=d2*d
      d4=d3*d
      d5=d4*d
      d6=d5*d
      d7=d6*d
      d8=d7*d
      d9=d8*d
      d10=d9*d
      d11=d10*d
      d12=d11*d
      d13=d12*d
      tS=dsqrt(t)
      t2=t*t
      t3=t2*t
      t4=t3*t
      t5=t4*t
      f=dexp(gm*d2)
C=====================================================================
=========
      if(k.eq.1) then
C     ENTRY PRESS
       b(1)=d2*t
                  b(2)=d2*tS
                  b(3)=d2
                  b(4)=d2/t
                  b(5)=d2/t2
                  b(6)=d3*t
       b(7)=d3
                  b(8)=d3/t
                  b(9)=d3/t2
                  b(10)=d4*t
                  b(11)=d4
                  b(12)=d4/t
       b(13)=d5
                  b(14)=d6/t
                  b(15)=d6/t2
                  b(16)=d7/t
                  b(17)=d8/t
       b(18)=d8/t2
                  b(19)=d9/t2
                  b(20)=d3*f/t2
                  b(21)=d3*f/t3
       b(22)=d5*f/t2
                  b(23)=d5*f/t4
                  b(24)=d7*f/t2
                  b(25)=d7*f/t3
       b(26)=d9*f/t2
                  b(27)=d9*f/t4
                  b(28)=d11*f/t2
                  b(29)=d11*f/t3
       b(30)=d13*f/t2
                  b(31)=d13*f/t3
                  b(32)=d13*f/t4
```

```
      pc=r*t*d
C=============================================================
=========
    else if(k.eq.2) then
C     ENTRY DPDD
      f1=2.d0*f*gm*d
                  f21=3.d0*f*d2 +f1*d3
                  f22=5.d0*f*d4 +f1*d5
      f23=7.d0*f*d6 +f1*d7
                  f24=9.d0*f*d8 +f1*d9
      f25=11.d0*f*d10+f1*d11
                  f26=13.d0*f*d12+f1*d13
      b(1)=2.d0*d*t
                  b(2)=2.d0*d*tS
                  b(3)=2.d0*d
                  b(4)=2.d0*d/t
      b(5)=2.d0*d/t2
                  b(6)=3.d0*d2*t
                  b(7)=3.d0*d2
                  b(8)=3.d0*d2/t
      b(9)=3.d0*d2/t2
                  b(10)=4.d0*d3*t
                  b(11)=4.d0*d3
                  b(12)=4.d0*d3/t
      b(13)=5.d0*d4
                  b(14)=6.d0*d5/t
                  b(15)=6.d0*d5/t2
                  b(16)=7.d0*d6/t
      b(17)=8.d0*d7/t
                  b(18)=8.d0*d7/t2
                  b(19)=9.d0*d8/t2
                  b(20)=f21/t2
      b(21)=f21/t3
                  b(22)=f22/t2
                  b(23)=f22/t4
                  b(24)=f23/t2
      b(25)=f23/t3
                  b(26)=f24/t2
                  b(27)=f24/t4
                  b(28)=f25/t2
      b(29)=f25/t3
                  b(30)=f26/t2
                  b(31)=f26/t3
                  b(32)=f26/t4
      pc=r*t
C=============================================================
=========
    else if(k.eq.3) then
C     ENTRY DPDT
      b(1)=d2
                  b(2)=d2/(2.d0*tS)
                  b(3)=0.d0
                  b(4)=-d2/t2
      b(5)=-2.d0*d2/t3
```

```
                b(6)=d3
                b(7)=0.d0
                b(8)=-d3/t2
                b(9)=-2.d0*d3/t3
      b(10)=d4
                b(11)=0.d0
                b(12)=-d4/t2
                b(13)=0.d0
                b(14)=-d6/t2
      b(15)=-2.d0*d6/t3
                b(16)=-d7/t2
                b(17)=-d8/t2
                b(18)=-2.d0*d8/t3
      b(19)=-2.d0*d9/t3
                b(20)=-2.d0*d3*f/t3
                b(21)=-3.d0*d3*f/t4
      b(22)=-2.d0*d5*f/t3
                b(23)=-4.d0*d5*f/t5
                b(24)=-2.d0*d7*f/t3
      b(25)=-3.d0*d7*f/t4
                b(26)=-2.d0*d9*f/t3
                b(27)=-4.d0*d9*f/t5
      b(28)=-2.d0*d11*f/t3
                b(29)=-3.d0*d11*f/t4
                b(30)=-2.d0*d13*f/t3
      b(31)=-3.d0*d13*f/t4
                b(32)=-4.d0*d13*f/t5
      pc=r*d
C=============================================================
=========
    else if(k.eq.4) then
C     ENTRY DSDN
C     PARTIAL OF ENTROPY WITH
C     RESPECT TO THE G COEFFICIENTS
C     S=S0-R*LOGF(D*R*T/P0)+(DSDN(D)-DSDN(0))*1000.DO +CPOS(T)
      g1=f/(2.d0*gm)
                g2=(f*d2-2.d0*g1)/(2.d0*gm)
      g3=(f*d4-4.d0*g2)/(2.d0*gm)
                g4=(f*d6-6.d0*g3)/(2.d0*gm)
      g5=(f*d8-8.d0*g4)/(2.d0*gm)
                g6=(f*d10-10.d0*g5)/(2.d0*gm)
      b(1)=-d
                b(2)=-d/(2.d0*tS)
                b(3)=0.d0
                b(4)=+d/t2
                b(5)=2.d0*d/t3
      b(6)=-d2/2.d0
                b(7)=0.d0
                b(8)=d2/(2.d0*t2)
                b(9)=d2/t3
      b(10)=-d3/3.d0
                b(11)=0.d0
                b(12)=d3/(3.d0*t2)
                b(13)=0.d0
```

```
      b(14)=d5/(5.d0*t2)
                  b(15)= 2.d0*d5/(5.d0*t3)
                  b(16)=d6/(6.d0*t2)
      b(17)=d7/(7.d0*t2)
                  b(18)=2.d0*d7/(7.d0*t3)
                  b(19)=d8/(4.d0*t3)
      b(20)=2.d0*g1/t3
                  b(21)=3.d0*g1/t4
                  b(22)=2.d0*g2/t3
      b(23)=4.d0*g2/t5
                  b(24)=2.d0*g3/t3
                  b(25)=3.d0*g3/t4
      b(26)=2.d0*g4/t3
                  b(27)=4.d0*g4/t5
                  b(28)=2.d0*g5/t3
      b(29)=3.d0*g5/t4
                  b(30)=2.d0*g6/t3
                  b(31)=3.d0*g6/t4
      b(32)=4.d0*g6/t5
      pc=0.d0
C=============================================================
========
    else if(k.eq.5) then
C     ENTRY DUDN
C     TERMS NEEDED FOR ENTHALPY CALCULATION
C     H=H0+(T*dSDN(D)-DSDN(0))*1000.+(DUDN(D-DUDN(0))*1000.+CPOH(T)
C     +(P/D-R*T)*1000.
      g1=f/(2.d0*gm)
                  g2=(f*d2-2.d0*g1)/(2.d0*gm)
      g3=(f*d4-4.d0*g2)/(2.d0*gm)
                  g4=(f*d6-6.d0*g3)/(2.d0*gm)
      g5=(f*d8-8.d0*g4)/(2.d0*gm)
                  g6=(f*d10-10.d0*g5)/(2.d0*gm)
      b(1)=d*t
                  b(2)=d*tS
                  b(3)=d
                  b(4)=d/t
                  b(5)=d/t2
                  b(6)=d2*t/2.d0
      b(7)=d2/2.d0
                  b(8)=d2/(2.d0*t)
                  b(9)=d2/(2.d0*t2)
      b(10)=d3*t/3.d0
                  b(11)=d3/3.d0
                  b(12)=d3/(3.d0*t)
                  b(13)=d4/4.d0
      b(14)=d5/(5.d0*t)
                  b(15)=d5/(5.d0*t2)
                  b(16)=d6/(6.d0*t)
      b(17)=d7/(7.d0*t)
                  b(18)=d7/(7.d0*t2)
                  b(19)=d8/(8.d0*t2)
      b(20)=g1/t2
                  b(21)=g1/t3
```

```
                b(22)=g2/t2
                b(23)=g2/t4
                b(24)=g3/t2
        b(25)=g3/t3
                b(26)=g4/t2
                b(27)=g4/t4
                b(28)=g5/t2
                b(29)=g5/t3
        b(30)=g6/t2
                b(31)=g6/t3
                b(32)=g6/t4
        pc=0.d0
C=============================================================
=========
      else if(k.eq.6) then
C     ENTRY TDSDT
C     TEMP. TIMES THE PARTIAL OF
C     ENTROPY WITH RESPECT TO TEMP.
C     CV=CV0+(tdSdN(/)-tdSdN(d))*1000.
        g1=f/(2.d0*gm)
                g2=(f*d2-2.d0*g1)/(2.d0*gm)
        g3=(f*d4-4.d0*g2)/(2.d0*gm)
                g4=(f*d6-6.d0*g3)/(2.d0*gm)
        g5=(f*d8-8.d0*g4)/(2.d0*gm)
                g6=(f*d10-10.d0*g5)/(2.d0*gm)
        b(1)=0.d0
                b(2)=-d/(4.d0*tS)
                b(3)=0.d0
                b(4)=2.d0*d/t2
        b(5)=6.d0*d/t3
                b(6)=0.d0
                b(7)=0.d0
                b(8)=d2/t2
        b(9)=3.d0*d2/t3
                b(10)=0.d0
                b(11)=0.d0
        b(12)=(2.d0*d3)/(3.d0*t2)
                b(13)=0.d0
                b(14)=(2.d0*d5)/(5.d0*t2)
        b(15)=(6.d0*d5)/(5.d0*t3)
                b(16)=d6/(3.d0*t2)
        b(17)=(2.d0*d7)/(7.d0*t2)
                b(18)=(6.d0*d7)/(7.d0*t3)
        b(19)=(3.d0*d8)/(4.d0*t3)
                b(20)=6.d0*g1/t3
                b(21)=12.d0*g1/t4
        b(22)=6.d0*g2/t3
                b(23)=20.d0*g2/t5
                b(24)=6.d0*g3/t3
        b(25)=12.d0*g3/t4
                b(26)=6.d0*g4/t3
                b(27)=20.d0*g4/t5
        b(28)=6.d0*g5/t3
                b(29)=12.d0*g5/t4
```

```
                b(30)=6.d0*g6/t3
      b(31)=12.d0*g6/t4
                b(32)=20.d0*g6/t5
      pc=0.d0
    endif
C=============================================================
=========
    p=0.d0
    do 10 i=1,32
      p=p+g(i)*b(i)
  10 continue
    pp=p+pc
    return
    end
*****************************************************************************
    real*8 function cpi(tt,k)
    implicit real*8(a-h)
    implicit real*8(o-z)
    implicit integer*4(i-n)
C  CALCULATES SPECIFIC HEAT, ENTROPY, AND ENTHALPY FOR THE
IDEAL GAS.
C  OUTPUT IS IN J/(MOL*K), FOR CP AND S,  AND J/MOL FOR H.
C  HYDROGEN, (N=1), IS TREATED AS A SPECIAL CASE AS THE COEFF. FOR
C  CP ARE IN THREE TEMPERATURE RANGES.  T < 40 K,  40 < T < 140 K,
C  AND T > 140 K.
    common/cpid/gi(11),gh(11),gl(11)
    common/isp/n,nw,nww
    dimension g0(11)
    data g0/3*0.d0,2.5000315d0,7*0.d0/,r/8.31434d0/
    if(n.eq.1) then
      if(tt.lt.140.d0) then
        g0(9)=gh(9)
        gi(9)=gh(9)
        gi(10)=gh(10)+(eq3(140.d0,gh)-eq3(140.d0,gl))*r
        gi(11)=gh(11)+(eq2(140.d0,gh)-eq2(140.d0,gl))*r
        if(tt.lt.40.d0) then
          gi(10)=gi(10)+(eq3(40.d0,gl)-eq3(40.d0,g0))*r
          gi(11)=gi(11)+(eq2(40.d0,gl)-eq2(40.d0,g0))*r
          do 10 i=1,8
            gi(i)=g0(i)
  10        continue
        else
          do 15 i=1,8
            gi(i)=gl(i)
  15        continue
        endif
      else
        do 20 i=1,11
          gi(i)=gh(i)
  20      continue
      endif
    endif
    if(k.eq.1) then
      cpi=eq1(tt,gi)*r
```

```
      else if(k.eq.2) then
        cpi=eq2(tt,gi)*r+gi(11)
      else if(k.eq.3) then
        cpi=eq3(tt,gi)*r+gi(10)
      endif
      return
      end
*************************************************************************
      real*8 function eq1(t,g)
      implicit real*8(a-h)
      implicit real*8(o-z)
      implicit integer*4(i-n)
      dimension g(11)
      u=g(9)/t
      eu=dexp(u)
      eq1=g(1)/(t**3.d0)+g(2)/(t**2.d0)+g(3)/t+g(4)+g(5)*t+g(6)*t**2.d0
     & +g(7)*t**3.d0+g(8)*eu*(u/(eu-1.d0))**2.d0
      return
      end
*************************************************************************
      real*8 function eq2(t,g)
      implicit real*8(a-h)
      implicit real*8(o-z)
      implicit integer*4(i-n)
      dimension g(11)
      u=g(9)/t
      eu=dexp(u)
      eq2=g(1)/(-3.d0*t**3.d0)+g(2)/(-2.d0*t**2.d0)+g(3)/(-t)
     & +g(4)*dlog(t)+g(5)*t+g(6)/2.d0*t**2.d0+g(7)/3.d0*t**3.d0
     & +g(8)*(u/(eu-1.d0)-dlog(1.d0-1.d0/eu))
      return
      end
*************************************************************************
      real*8 function eq3(t,g)
      implicit real*8(a-h)
      implicit real*8(o-z)
      implicit integer*4(i-n)
      dimension g(11)
      u=g(9)/t
      eu=dexp(u)
      eq3=g(1)/(-2.d0*t**2.d0)+g(2)/(-t)+g(3)*dlog(t)+g(4)*t
     & +g(5)/2.d0*t**2.d0+g(6)/3.d0*t**3.d0+g(7)/4.d0*t**4.d0
     & +g(8)*g(9)/(eu-1.d0)
      return
      end
```

APPENDIX B

Run Program Scripts

Run Program Script from the Pump Performance Program

```
on mouseUp
  global check1,check2,check3,check4,selection,selection2
  global efficiency,pressureratio,flowrate,pumpspeed,frictionloss,elevation
  put 1 into proceed
  if check1 = 0 then
    show card field id 19
    put 0 into proceed
  end if
  if check2 = 0 then
    show card field id 20
    put 0 into proceed
  end if
  if check3 = 0 then
    show card field id 21
    put 0 into proceed
  end if
  if check4 = 0 then
    show card field id 22
    put 0 into proceed
  end if
  if proceed = 1 then
    put line 1 of card field id 12 of card id 7230 into number
    put 1 into pt
    if selection2 = 0 then
      put line 1 of card field id 12 of card id 7230 into line 6 of background field id 2 of
card id 7925
      put 7 into dataf
      repeat for number times
        if line pt of card field id 14 of card id 7230 is empty then
          put line (pt-1) of card field id 14 of card id 7230 into line dataf of background field
id 2 of card id 7925
        else
          put line pt of card field id 14 of card id 7230 into line dataf of background field id 2
of card id 7925
        end if
        add 1 to pt
        add 1 to dataf
      end repeat
    else
      put line 1 of card field id 12 of card id 7230 into line 4 of background field id 2 of
card id 7925
      put 5 into dataf
```

```
    repeat for number times
      if line pt of card field id 14 of card id 7230 is empty or line pt of card field id 17 of
card id 7230 is empty then
        if line pt of card field id 14 of card id 7230 is empty then
          put line (pt-1) of card field id 14 of card id 7230 into our
        else
          put line pt of card field id 14 of card id 7230 into our
        end if
        if line pt of card field id 17 of card id 7230 is empty then
          put line (pt-1) of card field id 17 of card id 7230 into you
        else
          put line pt of card field id 17 of card id 7230 into you
        end if
        put our && you into line dataf of background field id 2 of card id 7925
        add 1 to pt
        add 1 to dataf
      else
        put line pt of card field id 14 of card id 7230 into our
        put line pt of card field id 17 of card id 7230 into you
        put our && you into line dataf of background field id 2 of card id 7925
        add 1 to pt
        add 1 to dataf
      end if
    end repeat
  end if
  put selection2 into line 3 of background field id 2 of card id 7925
  hide card field id 28
  show card field id 29
  put "::LRE-CP^2:data in" into datain
  open file datain
  write background field id 2 of card id 7925 to file datain
  close file datain
  open MIPROP
  put "::LRE-CP^2:data out" into dataout
  wait 5 seconds
  open file dataout
  put 1 into number1
  put empty into card field id 18 of card id 8886
  put empty into card field id 19 of card id 8886
  put empty into card field id 20 of card id 8886
  put empty into card field id 21 of card id 8886
  put empty into card field id 22 of card id 8886
  put empty into card field id 23 of card id 8886
  put empty into card field id 24 of card id 8886
  put empty into card field id 25 of card id 8886
  put empty into card field id 30 of card id 8886
  put "System Messages" into line 1 of card field id 30 of card id 8886
  repeat for number times
    put number1 into line number1 of card field id 18 of card id 8886
    read from file dataout until return
    put It into ptemp
    put ptemp into line number1 of card field id 19 of card id 8886
    read from file dataout until return
    put It into ttemp
```

```
 put ttemp into line number1 of card field id 20 of card id 8886
 read from file dataout until return
 put It into dtemp
 put dtemp into line number1 of card field id 21 of card id 8886
 read from file dataout until return
 put It into utemp
 put utemp into line number1 of card field id 22 of card id 8886
 read from file dataout until return
 put It into htemp
 put htemp into line number1 of card field id 23 of card id 8886
 read from file dataout until return
 put It into stemp
 put stemp into line number1 of card field id 24 of card id 8886
 read from file dataout until return
 put It into rtotemp
 put rtotemp into line number1 of card field id 25 of card id 8886
 add 1 to number1
end repeat
close file dataout
put "::LRE-CP^2:SDH:messages" into messages
open file messages
read from file messages for 16834
put It after line 2 of card field id 30 of card id 8886
close file messages
put card field id 40 of card id 4374 into pressureratio
put 1 into number1
put 5 into dataf
repeat for number times
 put line number1 of card field id 19 of card id 8886 into ptemp
 multiply ptemp by pressureratio
 put line number1 of card field id 24 of card id 8886 into stemp
 put ptemp && stemp into line dataf of background field id 2 of card id 7925
 add 1 to number1
 add 1 to dataf
end repeat
put 4 into line 3 of background field id 2 of card id 7925
open file datain
write background field id 2 of card id 7925 to file datain
close file datain
open MIPROP
wait 5 seconds
open file dataout
put 1 into number1
put empty into card field id 22 of card id 8979
put empty into card field id 23 of card id 8979
put empty into card field id 24 of card id 8979
put empty into card field id 25 of card id 8979
put empty into card field id 26 of card id 8979
put empty into card field id 27 of card id 8979
put empty into card field id 28 of card id 8979
repeat for number times
 read from file dataout until return
 put It into ptemp
 put ptemp into line number1 of card field id 22 of card id 8979
```

```
 read from file dataout until return
 put It into ttemp
 put ttemp into line number1 of card field id 23 of card id 8979
 read from file dataout until return
 put It into dtemp
 put dtemp into line number1 of card field id 24 of card id 8979
 read from file dataout until return
 put It into utemp
 put utemp into line number1 of card field id 25 of card id 8979
 read from file dataout until return
 put It into htemp
 put htemp into line number1 of card field id 26 of card id 8979
 read from file dataout until return
 put It into stemp
 put stemp into line number1 of card field id 27 of card id 8979
 read from file dataout until return
 put It into rtotemp
 put rtotemp into line number1 of card field id 28 of card id 8979
 add 1 to number1
end repeat
close file dataout
put card field id 38 of card id 4374 into efficiency
put card field id 40 of card id 4374 into pressureratio
put 1 into number1
put 5 into dataf
repeat for number times
  put line number1 of card field id 23 of card id 8886 into htemp
  put line number1 of card field id 26 of card id 8979 into htemp2
  put (htemp + (htemp2-htemp)/efficiency) into htemp3
  put line number1 of card field id 19 of card id 8886 into ptemp
  put (ptemp * pressureratio) into ptemp3
  put ptemp3 && htemp3 into line dataf of background field id 2 of card id 7925
  add 1 to number1
  add 1 to dataf
end repeat
put 6 into line 3 of background field id 2 of card id 7925
open file datain
write background field id 2 of card id 7925 to file datain
close file datain
open MIPROP
wait 5 seconds
open file dataout
put 1 into number1
put empty into card field id 10 of card id 8979
put empty into card field id 11 of card id 8979
put empty into card field id 12 of card id 8979
put empty into card field id 13 of card id 8979
put empty into card field id 14 of card id 8979
put empty into card field id 15 of card id 8979
put empty into card field id 16 of card id 8979
put empty into card field id 17 of card id 8979
put empty into card field id 29 of card id 8979
put "System Messages" into line 1 of card field id 29 of card id 8979
repeat for number times
```

```
    put number1 into line number1 of card field id 10 of card id 8979
    read from file dataout until return
    put It into ptemp
    put ptemp into line number1 of card field id 11 of card id 8979
    read from file dataout until return
    put It into ttemp
    put ttemp into line number1 of card field id 12 of card id 8979
    read from file dataout until return
    put It into dtemp
    put dtemp into line number1 of card field id 13 of card id 8979
    read from file dataout until return
    put It into utemp
    put utemp into line number1 of card field id 14 of card id 8979
    read from file dataout until return
    put It into htemp
    put htemp into line number1 of card field id 15 of card id 8979
    read from file dataout until return
    put It into stemp
    put stemp into line number1 of card field id 16 of card id 8979
    read from file dataout until return
    put It into rtotemp
    put rtotemp into line number1 of card field id 17 of card id 8979
    add 1 to number1
  end repeat
  close file dataout
  open file messages
  read from file messages for 16834
  put It after line 2 of card field id 29 of card id 8979
  close file messages
  hide card field id 29
  show card field id 30
  play "harpsichord"
  wait 2 seconds
  hide card field id 30
  hide card field id 31
 end if
 put card field id 6 of card id 7230 into efficiency
 put card field id 40 of card id 4374 into pressureratio
 put card field id 42 of card id 4374 into flowrate
 put card field id 4 of card id 6365 into pumpspeed
 put card field id 10 of card id 7230 into frictionloss
 put card field id 11 of card id 7230 into elevation
end mouseUp
```

Run Program Script from the Valve Performance Program

```
on mouseUp
 global check1,check2,check3,selection,selection2
 put 1 into proceed
 if check1 = 0 then
```

```
 play "harpsichord" "a"
 show card field id 26
 put 0 into proceed
end if
if check2 = 0 then
 play "harpsichord" "a"
 show card field id 28
 put 0 into proceed
end if
if card field id 53 is empty then
 put 0 into card field id 53
end if
if selection2 <> 0 then
 if card field id 33 is empty then
  put 0 into check3
 else
  if card field id 35 is empty then
   put 0 into check3
  else
   if card field id 37 is empty then
    put 0 into check3
   else
    if card field id 39 is empty and card field id 53 = 0 then
     put 1 into check3
    else
     if card field id 39 is not empty and card field id 53 = 0 then
      put 1 into check3
     else
      if card field id 39 is not empty and card field id 53 <> 0 then
       put 1 into check3
      else
       put 0 into check3
      end if
     end if
    end if
   end if
  end if
 end if
else
 if card field id 33 is empty then
  put 0 into check3
 else
  if card field id 35 is empty then
   put 0 into check3
  else
   if card field id 39 is empty and card field id 53 = 0 then
    put 1 into check3
   else
    if card field id 39 is not empty and card field id 53 = 0 then
     put 1 into check3
    else
     if card field id 39 is not empty and card field id 53 > 0 then
      put 1 into check3
     else
```

```
        put 0 into check3
       end if
      end if
     end if
    end if
   end if
  end if
 if check3 = 0 then
  play "harpsichord" "a"
  show card field id 63
  put 0 into proceed
 end if
 if proceed = 1 then
  hide card field id 26
  hide card field id 28
  hide card field id 63
  put line 1 of card field id 43 into number
  put 1 into pt
  if selection2 = 0 then
   put number into line 6 of background field id 1 of card id 4448
   put 7 into dataf
   repeat for number times
    if line pt of card field id 35 is empty then
     put line (pt-1) of card field id 35 into line dataf of background field id 1 of card id
4448
     add 1 to pt
     add 1 to dataf
    else
     put line pt of card field id 35 into line dataf of background field id 1 of card id 4448
     add 1 to pt
     add 1 to dataf
    end if
   end repeat
  else
   put number into line 4 of background field id 1 of card id 4448
   put 5 into dataf
   repeat for number times
    if line pt of card field id 35 is empty or line pt of card field id 37 is empty then
     if line pt of card field id 35 is empty then
      put line (pt-1) of card field id 35 into our
     else
      put line pt of card field id 35 into our
     end if
     if line pt of card field id 37 is empty then
      put line (pt-1) of card field id 37 into you
     else
      put line pt of card field id 37 into you
     end if
     put our && you into line dataf of background field id 1 of card id 4448
     add 1 to pt
     add 1 to dataf
    else
     put line pt of card field id 35 into our
     put line pt of card field id 37 into you
```

```
      put our && you into line dataf of background field id 1 of card id 4448
      add 1 to pt
      add 1 to dataf
    end if
  end repeat
end if
show card field id 30
put selection2 into line 3 of background field id 1 of card id 4448
put "::LRE-CP^2:SDH:data in" into datain
open file datain
write background field id 1 of card id 4448 to file datain
close file datain
open MIPROP
put "::LRE-CP^2:data out" into dataout
wait 5 seconds
open file dataout
put 1 into number1
repeat with increment = 3 to 11
  put empty into card field id increment of card id 5547
end repeat
put empty into card field id 42 of card id 5547
put "Inlet System Messages" into line 1 of card field id 42 of card id 5547
repeat for number times
  put number1 into line number1 of card field id 3 of card id 5547
  read from file dataout until return
  put It into ptemp
  put ptemp into line number1 of card field id 5 of card id 5547
  read from file dataout until return
  put It into ttemp
  put ttemp into line number1 of card field id 6 of card id 5547
  read from file dataout until return
  put It into dtemp
  put dtemp into line number1 of card field id 7 of card id 5547
  read from file dataout until return
  put It into utemp
  put utemp into line number1 of card field id 8 of card id 5547
  read from file dataout until return
  put It into htemp
  put htemp into line number1 of card field id 9 of card id 5547
  read from file dataout until return
  put It into stemp
  put stemp into line number1 of card field id 10 of card id 5547
  read from file dataout until return
  put It into rtotemp
  add 1 to number1
end repeat
close file dataout
put 1 into number1
repeat for number times
  put line number1 of card field id 41 into line number1 of card field id 4 of card id
5547
  put line number1 of card field id 39 into line number1 of card field id 11 of card id
5547
  add 1 to number1
```

```
end repeat
put 1 into number1
repeat for number times
 if line number1 of card field id 39 is empty then
  put "N/A" into line number1 of card field id 11 of card id 5547
  add 1 to number1
 end if
end repeat
put 1 into number1
repeat for number times
 if line number1 of card field id 41 is empty then
  put "N/A" into line number1 of card field id 4 of card id 5547
  add 1 to number1
 end if
end repeat
put "::LRE-CP^2:messages" into messages
open file messages
read from file messages for 16834
put It after line 2 of card field id 42 of card id 5547
close file messages
put card field id 53 into heatrate
put 1 into number1
put 5 into dataf
repeat for number times
 if line number1 of card field id 39 is empty then
  put 1 into flowrate
 else
  put line number1 of card field id 39 into flowrate
 end if
 put line number1 of card field id 9 of card id 5547 into htemp
 put (heatrate/flowrate + htemp) into htemp2
 put line number1 of card field id 5 of card id 5547 into ptemp
 if line number1 of card field id 33 is empty then
  put line (number1-1) of card field id 33 into pdelta
 else
  put line number1 of card field id 33 into pdelta
 end if
 put (ptemp-pdelta) into ptemp2
 put ptemp2 && htemp2 into line dataf of background field id 1 of card id 4448
 add 1 to number1
 add 1 to dataf
end repeat
put 6 into line 3 of background field id 1 of card id 4448
open file datain
write background field id 1 of card id 4448 to file datain
close file datain
open MIPROP
wait 5 seconds
open file dataout
put 1 into number1
repeat with increment = 21 to 29
 put empty into card field id increment of card id 5547
end repeat
repeat for number times
```

```
    put number1 into line number1 of card field id 21 of card id 5547
    read from file dataout until return
    put It into ptemp
    put ptemp into line number1 of card field id 23 of card id 5547
    read from file dataout until return
    put It into ttemp
    put ttemp into line number1 of card field id 24 of card id 5547
    read from file dataout until return
    put It into dtemp
    put dtemp into line number1 of card field id 25 of card id 5547
    read from file dataout until return
    put It into utemp
    put utemp into line number1 of card field id 26 of card id 5547
    read from file dataout until return
    put It into htemp
    put htemp into line number1 of card field id 27 of card id 5547
    read from file dataout until return
    put It into stemp
    put stemp into line number1 of card field id 28 of card id 5547
    read from file dataout until return
    put It into rtotemp
    add 1 to number1
  end repeat
  close file dataout
  put 1 into number1
  repeat for number times
    put line number1 of card field id 41 into line number1 of card field id 22 of card id
5547
    put line number1 of card field id 39 into line number1 of card field id 29 of card id
5547
    add 1 to number1
  end repeat
  put 1 into number1
  repeat for number times
    if line number1 of card field id 39 is empty then
      put "N/A" into line number1 of card field id 29 of card id 5547
      add 1 to number1
    end if
  end repeat
  put 1 into number1
  repeat for number times
    if line number1 of card field id 41 is empty then
      put "N/A" into line number1 of card field id 22 of card id 5547
      add 1 to number1
    end if
  end repeat
  put empty into card field id 43 of card id 5547
  put "Outlet System Messages" into line 1 of card field id 43 of card id 5547
  open file messages
  read from file messages for 16834
  put It after line 2 of card field id 43 of card id 5547
  close file messages
  --Compute valve flow coefficient--
  put empty into card field id 2 of card id 7084
```

```
  put empty into card field id 13 of card id 7084
  put empty into card field id 6 of card id 7084
  put 1 into number1
  repeat for number times
    put number1 into line number1 of card field id 2 of card id 7084
    add 1 to number1
  end repeat
  set numberformat to "0.00"
  put 1 into number1
  repeat for number times
    if card field id 56 is empty or line number1 of card field id 39 is empty then
      put "N/A" into line number1 of card field id 13 of card id 7084
    else
      put card field id 56 into diam
      put line number1 of card field id 7 of card id 5547 into dtemp
      put (flowrate/dtemp*7.48) into volrate
      put line number1 of card field id 5 of card id 5547 into ptemp1
      put line number1 of card field id 23 of card id 5547 into ptemp2
      put (volrate*sqrt(dtemp/62.367/abs(ptemp1-ptemp2))) into vfc1
      put (1/(890*diam^4/vfc1^2 + 1)^0.5) into vfc2
      put vfc2 into line number1 of card field id 13 of card id 7084
    end if
    add 1 to number1
  end repeat
  --Compute valve loss coefficient--
  set numberformat to "0.00"
  put 1 into number1
  repeat for number times
    if card field id 56 is empty or line number1 of card field id 39 is empty then
      put "N/A" into line number1 of card field id 6 of card id 7084
    else
      put card field id 56 into diam
      put line number1 of card field id 7 of card id 5547 into dtemp
      put (flowrate/dtemp*7.48) into volrate
      put line number1 of card field id 5 of card id 5547 into ptemp1
      put line number1 of card field id 23 of card id 5547 into ptemp2
      put (volrate*sqrt(dtemp/62.367/abs(ptemp1-ptemp2))) into vfc1
      put (890.3958*diam^4/vfc1^2) into vlc
      put vlc into line number1 of card field id 6 of card id 7084
    end if
    add 1 to number1
  end repeat
  hide card field id 30
  play "harpsichord"
  show card field id 60
  wait 2 seconds
  hide card field id 60
  hide card field id 64
 end if

end mouseUp
```

Run Program Script from the Propellant Tank Performance Program

```
on mouseUp
  global check1,check2,check3,check4,selection,selection2
  hide card field id 47 of card id 7812
  put 1 into proceed
  if check1 = 0 then
    play "harpsichord" "a"
    show card field id 23
    put 0 into proceed
  end if
  if check2 = 0 then
    play "harpsichord" "a"
    show card field id 24
    put 0 into proceed
  end if
  put 1 into check3
  if card field id 12 is empty then
    put 0 into check3
  end if
  if card field id 15 is empty then
    put 0 into check3
  end if
  if check3 = 0 then
    play "harpsichord" "a"
    show card field id 25
    put 0 into proceed
  end if
  if proceed = 1 then
    hide card field id 23
    hide card field id 24
    hide card field id 25
    put line 1 of card field id 3 into number
    if hilite of card button id 13 of card id 4122 is true then
      put selection2 into line 3 of background field id 1 of card id 4662
      put 0 into line 4 of background field id 1 of card id 4662
      put 0 into line 5 of background field id 1 of card id 4662
      put number into line 6 of background field id 1 of card id 4662
      put 7 into dataf
      put 1 into pt
      repeat for number times
        put line pt of card field id 12 into line dataf of background field id 1 of card id 4662
        add 1 to pt
        add 1 to dataf
      end repeat
      show card field id 26
      put "::LRE-CP^2:data in" into datain
      open file datain
      write background field id 1 of card id 4662 to file datain
      close file datain
      open MIPROP
      put "::LRE-CP^2:data out" into dataout
```

```
wait 5 seconds
open file dataout
put 1 into number1
repeat with increment = 14 to 19
  put empty into card field id increment of card id 7812
end repeat
repeat with increment = 21 to 23
  put empty into card field id increment of card id 7812
end repeat
put empty into card field id 36 of card id 7812
repeat with increment = 41 to 46
  put empty into card field id increment of card id 7812
end repeat
put empty into card field id 24 of card id 7812
put "Saturated Liquid Messages" into line 1 of card field id 24 of card id 7812
repeat for number times
  put number1 into line number1 of card field id 14 of card id 7812
  read from file dataout until return
  put It into ptemp
  put ptemp into line number1 of card field id 15 of card id 7812
  read from file dataout until return
  put It into ttemp
  put ttemp into line number1 of card field id 16 of card id 7812
  read from file dataout until return
  put It into dtemp
  put dtemp into line number1 of card field id 17 of card id 7812
  read from file dataout until return
  put It into utemp
  read from file dataout until return
  put It into htemp
  read from file dataout until return
  put It into stemp
  read from file dataout until return
  put It into rtotemp
  add 1 to number1
end repeat
close file dataout
put 1 into number1
repeat for number times
  put line number1 of card field id 15 into line number1 of card field id 23 of card id
7812
  add 1 to number1
end repeat
put 1 into number1
repeat for number times
  put line number1 of card field id 17 of card id 7812 into dtemp
  put line number1 of card field id 23 of card id 7812 into vtemp
  put line number1 of card field id 34 of card id 4662 into alpha1
  put (dtemp*vtemp*alpha1) into mtemp
  put mtemp into line number1 of card field id 19 of card id 7812
  add 1 to number1
end repeat
put "::LRE-CP^2:messages" into messages
open file messages
```

```
read from file messages for 16834
put It after line 2 of card field id 24 of card id 7812
close file messages
put 1 into line 4 of background field id 1 of card id 4662
put 7 into dataf
put 1 into pt
repeat for number times
  put line pt of card field id 12 into line dataf of background field id 1 of card id 4662
  add 1 to pt
  add 1 to dataf
end repeat
put "::LRE-CP^2:data in" into datain
open file datain
write background field id 1 of card id 4662 to file datain
close file datain
open MIPROP
put "::LRE-CP^2:data out" into dataout
wait 5 seconds
open file dataout
put 1 into number1
put empty into card field id 25 of card id 7812
put "Saturated Vapor Messages" into line 1 of card field id 25 of card id 7812
repeat for number times
  read from file dataout until return
  put It into ptemp
  read from file dataout until return
  put It into ttemp
  read from file dataout until return
  put It into dtemp
  put dtemp into line number1 of card field id 18 of card id 7812
  read from file dataout until return
  put It into utemp
  read from file dataout until return
  put It into htemp
  read from file dataout until return
  put It into stemp
  read from file dataout until return
  put It into rtotemp
  add 1 to number1
end repeat
close file dataout
put 1 into number1
repeat for number times
  put line number1 of card field id 18 of card id 7812 into dtemp
  put line number1 of card field id 23 of card id 7812 into vtemp
  put line number1 of card field id 34 of card id 4662 into alpha1
  put (dtemp*vtemp*(1-alpha1)) into mtemp
  put mtemp into line number1 of card field id 21 of card id 7812
  add 1 to number1
end repeat
put 1 into number1
repeat for number times
  put line number1 of card field id 19 of card id 7812 into mtempl
  put line number1 of card field id 21 of card id 7812 into mtempg
```

```
 put (mtempl + mtempg) into line number1 of card field id 22 of card id 7812
 add 1 to number1
end repeat
put "::LRE-CP^2:messages" into messages
open file messages
read from file messages for 16834
put It after line 2 of card field id 25 of card id 7812
close file messages
if hilite of card button id 8 of card id 4122 is true then
 put "Before tank emptying:" into card field id 4 of card id 7812
 put "After tank emptying:" into card field id 26 of card id 7812
 show card field id 47 of card id 7812
 put 1 into number1
 repeat for number times
  put number1 into line number1 of card field id 36 of card id 7812
  put line number1 of card field id 17 of card id 7812 into dtemp
  put line number1 of card field id 23 of card id 7812 into vtemp
  put line number1 of card field id 49 into alpha2
  put (dtemp*vtemp*alpha2) into line number1 of card field id 45 of card id 7812
  put line number1 of card field id 18 of card id 7812 into dtemp
  put (dtemp*vtemp*(1-alpha2)) into line number1 of card field id 46 of card id 7812
  add 1 to number1
 end repeat
 put 1 into number1
 repeat for number times
  put line number1 of card field id 45 of card id 7812 into mtempl
  put line number1 of card field id 46 of card id 7812 into mtempg
  put (mtempl + mtempg) into line number1 of card field id 41 of card id 7812
  add 1 to number1
 end repeat
 put 1 into number1
 repeat for number times
  put line number1 of card field id 45 of card id 7812 into ml2
  put line number1 of card field id 19 of card id 7812 into ml1
  put (ml2-ml1) into line number1 of card field id 42 of card id 7812
  put line number1 of card field id 46 of card id 7812 into mg2
  put line number1 of card field id 21 of card id 7812 into mg1
  put (mg2-mg1) into line number1 of card field id 43 of card id 7812
  put line number1 of card field id 41 of card id 7812 into mt2
  put line number1 of card field id 22 of card id 7812 into mt1
  put (mt2-mt1) into line number1 of card field id 44 of card id 7812
  add 1 to number1
 end repeat
 end if
 if hilite of card button id 53 of card id 4122 is true then
  put empty into card field id 4 of card id 7812
  put empty into card field id 26 of card id 7812
 end if
end if
if hilite of card button id 14 of card id 4122 is true then
 send "mouseup" to card button id 61
end if
if hilite of card button id 16 of card id 4122 is true then
 send "mouseup" to card button id 60
```

```
    end if
    hide card field id 26
    show card field id 27
    play "harpsichord"
    wait 2 seconds
    hide card field id 27
    hide card field id 22
  end if

end mouseUp
```

VITA

Stephen Douglas Howard was born in Silver Spring, Maryland on April 24, 1967. His family moved to Clearwater, Florida in 1976. In June 1985 he graduated from Pinellas Park High School in Clearwater. He attended the University of South Florida after high school, where he received an Associate's degree in August 1987. He entered the Georgia Institute of Technology in September 1987 and received a Bachelor of Aerospace Engineering degree in June 1990. He then enrolled at the University of Tennessee Space Institute the following August and received the Master of Science degree in aerospace engineering in August 1993.