

University of Alabama in Huntsville

**LOUIS**

---

Theses

UAH Electronic Theses and Dissertations

---

2008

## Experimental and analytical development program of the sequential feed system

Christopher Morton

Follow this and additional works at: <https://louis.uah.edu/uah-theses>

---

### Recommended Citation

Morton, Christopher, "Experimental and analytical development program of the sequential feed system" (2008). *Theses*. 431.  
<https://louis.uah.edu/uah-theses/431>

This Thesis is brought to you for free and open access by the UAH Electronic Theses and Dissertations at LOUIS. It has been accepted for inclusion in Theses by an authorized administrator of LOUIS.

**EXPERIMENTAL AND ANALYTICAL DEVELOPMENT PROGRAM OF THE  
SEQUENTIAL FEED SYSTEM**

**by**

**CHRISTOPHER MORTON**

**A THESIS**

**Submitted in partial fulfillment of the requirements  
for the degree of Master of Science in Engineering  
in  
The Department of Mechanical and Aerospace Engineering  
to  
The School of Graduate Studies  
of  
The University of Alabama in Huntsville**

**HUNTSVILLE, ALABAMA**

**2008**

In presenting this thesis in partial fulfillment of the requirements for the master's degree from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Christopher Morton  
(student signature)

3-4-08  
(date)

## THESIS APPROVAL FORM

Submitted by Christopher Morton in partial fulfillment of the requirements for the degree of Master of Science in Engineering and accepted on behalf of the Faculty of the School of Graduate Studies by the thesis committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate on the work described in this thesis. We further certify that we have reviewed the thesis manuscript and approve it in partial fulfillment of the requirements for the degree of Master of Science in Engineering.

James B. Blackmon 3/4/08 Committee Chair  
(Date)

Marlow D. Moore

Robert A. Dineen 3/4/08

\_\_\_\_\_

Kirk Frank Department Chair

Joe C. College Dean

Debra M. Moriarty 5/2/08 Graduate Dean

## ABSTRACT

The School of Graduate Studies  
The University of Alabama in Huntsville

Degree Master of Science in Engineering College/Dept. Engineering / Mechanical  
and Aerospace Engineering

Name of Candidate: Christopher Morton

Title: Experimental and Analytical Development Program of the Sequential Feed System

A novel liquid propellant feed system, termed the Sequential Feed System (SFS), delivers pressurized liquid through the sequential operation of a valve and tank system. Three relatively small high pressure tanks are sequentially filled from a larger low pressure supply tank, pressurized to high pressure, and drained. Inherent in this approach is a fail-operational mode; a three-tank system can continue to operate as a two-tank system if any of the valves should fail. The test program to evaluate the fail-operational aspect of the SFS is described; test program results proved the fail-operational capability. Additional analytical work on the fail-operational SFS and the SFS concept in general are also included. A sizing code developed previously for in space propulsion systems was extended to include launch vehicles with and without drag. Reliability analyses of the SFS in comparison to turbopump and conventional pressure fed systems show that it offers higher reliability for the same mean-time-between-failures of the valves.

Abstract Approval: Committee Chair

James B. Bleckman

Department Chair

Kadir Frenk

Graduate Dean

Debra M. Maricarity

## **ACKNOWLEDGMENTS**

I would like to thank Dr. James Blackmon for the immeasurable help he has given me during the course of my graduate studies. Additionally, I would like to thank the faculty and staff of the Propulsion Research Center for the great opportunities I have had to do research and further my education. The test program was also made possible with the contribution of Tony Hall, Dave Lineberry, as well as Dr. Marlow Moser and the other students, staff, and faculty working at the UAH Johnson Research Center. Many thanks go to David Eddleman and Kevin Pedersen for their prior work on the RFS project and their continued support for the ongoing research into the SFS. Lastly, I would like to thank my family for their years of support (and my mother's proofreading), and the many people in my life I have been fortunate to count as friends.

In memory of Dr. Clark Hawk and Dr. Mark Bower.

# TABLE OF CONTENTS

	Page
List of Figures .....	x
List of Tables .....	xiii
List of Abbreviations, Acronyms, and Terms.....	xvii
List of Symbols .....	xix
Chapter	
1 TECHNOLOGY BACKGROUND.....	1
1.1 Propellant Feed Systems.....	2
1.1.1 Conventional Pressure Fed Systems.....	3
1.1.2 Turbopump Systems.....	5
1.1.3 Other Propellant Feed System Types .....	9
1.2 Sequential Feed System Concept.....	11
1.2.1 Previous Experimental Work on the SFS .....	15
1.2.2 Previous Analytical Work on the SFS.....	20
2 ANALYTICAL WORK.....	26
2.1 Engine Mass Estimation .....	26
2.1.1 Model and Derivation of Equations .....	27
2.1.2 Implementation and Results .....	31
2.1.3 Alternate Formulation .....	35
2.2 Launch Vehicle Simulation .....	36
2.2.1 Equations and Algorithm.....	37
2.2.2 Verification and Validation .....	43

2.2.3	Alternate Version: Payload Analysis.....	45
2.3	SFS Reliability Analysis.....	48
2.3.1	Model .....	49
2.3.2	Derivation of Equations.....	57
2.3.3	Application.....	62
2.3.4	Numerical Implementation.....	66
2.3.5	Previous Work.....	68
2.3.6	Results .....	70
2.4	SFS Vent Stage Analysis .....	74
2.4.1	Iterative Solution .....	76
2.4.2	Analytical Solution.....	78
2.4.3	Implementation and Results .....	84
2.5	SFS Refill Stage Analysis.....	88
2.5.1	Derivation of Equations.....	89
2.5.2	Implementation and Algorithm .....	92
2.5.3	Results .....	94
3	SOFTWARE: INTEGRATED CONTROL AND MONITORING PROGRAM ...	97
3.1	Design.....	97
3.1.1	Program Features.....	98
3.1.2	Interface.....	103
3.1.3	Algorithms.....	109
3.1.4	States .....	113
3.2	Implementation.....	117
3.2.1	Hardware Control.....	117
3.2.2	Data Acquisition.....	119
3.2.3	Failure Simulation .....	121



3.2.4	Auto Sequence Generation.....	130
3.2.5	State Functions .....	134
3.2.6	Sub-VIs .....	137
4	EXPERIMENTAL WORK.....	139
4.1	Experimental Goals .....	139
4.2	Test Bed Configuration.....	142
4.2.1	Test Bed Subsystems – Run Tanks .....	143
4.2.2	Test Bed Subsystems – Water .....	150
4.2.3	Test Bed Subsystems – Air .....	158
4.2.4	Test Bed Subsystems – Data Acquisition.....	163
4.2.5	Test Bed Subsystems – Control.....	166
4.3	Test Program.....	169
5	EXPERIMENTAL RESULTS.....	173
5.1	Calibration Tests.....	173
5.2	Baseline Tests .....	177
5.3	Fail-Operational Tests .....	180
6	CONCLUSIONS AND FUTURE WORK .....	183
6.1	Conclusions .....	183
6.2	Future Work.....	187
	APPENDIX A: Excel and Visual Basic for Applications Code.....	191
A.1	Custom VBA I/O Methods .....	191
A.1.1	Introductory Comments.....	191
A.1.2	read Function.....	194

A.1.3 writeTo Method.....	195
A.1.4 getCell Function .....	195
A.1.5 getAddress Method .....	195
A.1.6 getCol Function .....	196
A.2 Analysis Spreadsheets.....	198
A.2.1 Mass Project .....	198
A.2.2 DeltaV .....	205
A.2.3 Reliability.....	211
A.2.4 Blowdown .....	222
A.2.5 Refill.....	235
A.3 Data Analysis Spreadsheets .....	241
A.3.1 Fluctuation.....	241
A.3.2 Variations .....	248
APPENDIX B: Test Procedure .....	258
B.1 Test Operation.....	258
B.1.1 Safety Concerns.....	258
B.1.2 Test Procedures .....	261
B.1.2.1 Pre and Post-Test Procedures .....	261
B.1.2.2 Calibration Tests .....	264
B.1.2.3 Baseline Tests .....	267
B.1.2.4 Fail-Operational Tests.....	268
B.2 Test Procedure Document.....	270
APPENDIX C: Analysis Results of Test Data .....	300
C.1 Baseline Test Analysis .....	300
C.2 Fail-Operational Test Analysis .....	308
REFERENCES .....	322

## LIST OF FIGURES

Figure	Page
1.1 Turbopump Cycle Diagrams.....	6
1.2 SFS Test Apparatus.....	16
1.3 SFS Test Bed at NASA MSFC North Test Area .....	17
1.4 Screenshot of Propellant Feed System Analysis Spreadsheet .....	21
2.1 Engine Mass Estimation Model.....	27
2.2 Mass Project Spreadsheet Calculation Section .....	32
2.3 Real vs. Calculated Engine Masses .....	34
2.4 Block Diagram for Conventional Pressure Fed System .....	49
2.5 Conventional Pressure Fed Block Diagram.....	51
2.6 Schematic Diagram of Example Turbopump System.....	53
2.7 Turbopump Block Diagram .....	55
2.8 SFS Block Diagram .....	56
2.9 SFS Tank Subsystem Block Diagram.....	56
2.10 Block Diagram for N-Feed Systems with Allowed Failure Modes.....	64
2.11 Reliability vs. Time Chart.....	71
2.12 Reliability vs. Time Chart, Reduced Scale .....	72
2.13 Vent Analysis Results Chart .....	87
3.1 ICM Settings Tab.....	104
3.2 ICM Data Tab .....	106
3.3 ICM Sequence Tab .....	108

3.4 Flow Chart of ICM program operation.....	110
3.5 Flow Chart of Idle State.....	114
3.6 Flow Chart of File and Auto States.....	115
3.7 Flow Chart of Response State.....	116
3.8 Sub-VI failureSetup Front Panel.....	123
3.9 ICM Error Code Conventions.....	126
3.10 Sub-VI basicDetect Program Flow Chart .....	127
4.1 Test Bed Schematic.....	142
4.2 SFS Run Tanks and Frame .....	144
4.3 Drain and Fill Manifolds.....	145
4.4 Gas Valve Assembly.....	147
4.5 Liquid Valve Assembly .....	149
4.6 Water Subsystem Schematic.....	150
4.7 Main Pump and Water Supply Tank.....	151
4.8 Pump Outlet and Accumulator Tank .....	153
4.9 Fill Line into Run Tank Subsystem .....	154
4.10 Run Tank Subsystem with Drain Line Installed .....	156
4.11 Drain Line to Catch Tank .....	157
4.12 Return Pump and Filter.....	158
4.13 Air Subsystem Schematic .....	159
4.14 Dome Regulator Mounted on Support Bracket .....	160
4.15 Hand Regulator Mounted on SFS Frame.....	161
4.16 Relief Valve Installed above Pressurization Manifold .....	162

4.17 Gas Valve Assembly with DAQ Hardware Installed .....	165
4.18 Ross Valve Controller.....	167
4.19 Control Subsystem Relay Panel.....	168
5.1 Baseline Test Water Outlet Pressure.....	179
5.2 Baseline Test Drain Transition .....	180
A.1 Mass Project Spreadsheet.....	198
A.2 DeltaV Spreadsheet.....	205
A.3 Reliability Spreadsheet .....	211
A.4 Blowdown Spreadsheet, Sim Sheet Iterative Solution.....	223
A.5 Blowdown Spreadsheet, Sim Sheet Analytical Solution .....	223
A.6 Blowdown Spreadsheet, Sim Sheet Comparison of Solutions .....	224
A.7 Blowdown Spreadsheet, Data Sheet .....	224
A.8 Refill Spreadsheet .....	235
A.9 Fluctuation Spreadsheet First Revision.....	241
A.10 Fluctuation Spreadsheet Second Revision .....	243
A.11 Variations Spreadsheet.....	249

## LIST OF TABLES

Table	Page
1.1 Test Matrix Used During Previous SFS Testing.....	18
1.2 Experimental Results of SFS Testing .....	19
1.3 Analysis Spreadsheet Validation Results: Shuttle OMS .....	22
1.4 Analytical Results for CEV Feed System Comparison .....	23
2.1 Engine Mass Input Parameters.....	29
2.2 DeltaV Input Parameters.....	38
2.3 DeltaV Verification and Validation Inputs/Outputs .....	44
2.4 Comparison of DeltaV Results .....	45
2.5 DeltaV Revision 6 Code Inputs and Outputs.....	47
2.6 Reliability Analysis Nomenclature .....	58
2.7 Input Section of Reliability spreadsheet .....	66
2.8 Reliability Spreadsheet Results.....	73
2.9 Vent Stage Analysis Nomenclature .....	75
2.10 Blowdown Spreadsheet Inputs.....	85
2.11 Blowdown Spreadsheet Outputs .....	86
2.12 Refill Analysis Symbols .....	90
2.13 Refill Program Inputs and Outputs .....	95
2.14 Refill Program Simulation Results .....	96
3.1 Failure Detection Tiers .....	112
3.2 ICM Program States.....	114

3.3 SFS Cycle Start Times .....	131
3.4 SFS Cycle Stage Start Time Equations.....	132
3.5 Sub-VI Descriptions.....	137
4.1 Test Bed Subsystem Descriptions.....	143
4.2 GVA Components.....	147
4.3 LVA Components .....	149
4.4 Water Subsystem Components .....	153
4.5 Water Subsystem Components Continued.....	154
4.6 Water Subsystem Components Continued.....	156
4.7 Water Subsystem Components Continued.....	157
4.8 DAQ Changes .....	163
4.9 DAQ Subsystem Components.....	165
4.10 2007 Experiment Test Matrix .....	170
4.11 Fail-Operational Test Matrix .....	171
5.1 Standard Auto Sequence Timing .....	176
5.2 Fail-Operational Test Results .....	182
A.1 Mass Project Cell Formulas .....	198
A.2 Mass Projects Inputs .....	199
A.3 Mass Project Inputs.....	200
A.4 Mass Project Outputs .....	201
A.5 Mass Project Outputs .....	202
A.6 Blowdown Spreadsheet Cell Formulas .....	225
C.1 First Analysis Results, Zero Overlap Test .....	300

C.2 First Analysis Results, 0.1 Second Overlap Test .....	301
C.3 First Analysis Results, 0.2 Second Overlap Test .....	301
C.4 First Analysis Results, 0.3 Second Overlap Test .....	301
C.5 First Analysis Results, 0.4 Second Overlap Test .....	302
C.6 First Analysis Results, 0.5 Second Overlap Test .....	302
C.7 Second Analysis Results, Zero Overlap and 1% Threshold.....	303
C.8 Second Analysis Results, 0.1 Second Overlap and 1% Threshold .....	303
C.9 Second Analysis Results, 0.2 Second Overlap and 1% Threshold .....	303
C.10 Second Analysis Results, 0.3 Second Overlap and 1% Threshold .....	304
C.11 Second Analysis Results, 0.4 Second Overlap and 1% Threshold .....	304
C.12 Second Analysis Results, 0.5 Second Overlap and 1% Threshold .....	305
C.13 Second Analysis Results, Zero Overlap and 2% Threshold.....	306
C.14 Second Analysis Results, 0.1 Second Overlap and 2% Threshold .....	306
C.15 Second Analysis Results, 0.2 Second Overlap and 2% Threshold .....	307
C.16 Second Analysis Results, 0.3 Second Overlap and 2% Threshold .....	307
C.17 Second Analysis Results, 0.4 Second Overlap and 2% Threshold .....	307
C.18 Second Analysis Results, 0.5 Second Overlap and 2% Threshold .....	307
C.19 Start Failure in Vent Valve of Tank 1 During Vent Stage .....	308
C.20 Start Failure in Drain Valve of Tank 1 During Drain Stage .....	309
C.21 Start Failure in Press Valve of Tank 1 During Drain Stage .....	309
C.22 Start Failure in Press Valve of Tank 1 During Press Stage.....	310
C.23 Start Failure in Fill Valve of Tank 1 During Fill Stage .....	310
C.24 Start Failure in Vent Valve of Tank 1 During Fill Stage .....	311



C.25 Stop Failure in Vent Valve of Tank 1 During Vent Stage .....	311
C.26 Stop Failure in Drain Valve of Tank 1 During Drain Stage.....	312
C.27 Stop Failure in Press Valve of Tank 1 During Drain Stage .....	312
C.28 Stop Failure in Press Valve of Tank 1 During Press Stage.....	313
C.29 Stop Failure in Fill Valve of Tank 1 During Fill Stage.....	313
C.30 Stop Failure in Vent Valve of Tank 1 During Fill Stage .....	314
C.31 Start Failure in Vent Valve of Tank 2 During Vent Stage .....	314
C.32 Start Failure in Drain Valve of Tank 2 During Drain Stage .....	315
C.33 Start Failure in Press Valve of Tank 2 During Drain Stage .....	315
C.34 Start Failure in Press Valve of Tank 2 During Press Stage.....	316
C.35 Start Failure in Fill Valve of Tank 2 During Fill Stage .....	316
C.36 Start Failure in Vent Valve of Tank 2 During Fill Stage .....	317
C.37 Start Failure in Vent Valve of Tank 3 During Vent Stage .....	317
C.38 Start Failure in Drain Valve of Tank 3 During Drain Stage .....	318
C.39 Start Failure in Press Valve of Tank 3 During Drain Stage .....	318
C.40 Stop Failure in Vent Valve of Tank 2 During Vent Stage .....	319
C.41 Stop Failure in Drain Valve of Tank 2 During Drain Stage.....	319
C.42 Stop Failure in Press Valve of Tank 2 During Drain Stage .....	320
C.43 Stop Failure in Press Valve of Tank 2 During Press Stage.....	320
C.44 Stop Failure in Fill Valve of Tank 2 During Fill Stage.....	321
C.45 Stop Failure in Vent Valve of Tank 2 During Fill Stage .....	321

## LIST OF ABBREVIATIONS, ACRONYMS, AND TERMS

<u>Term</u>	<u>Description</u>
	<u>General</u>
APTA	Autogenous Pressurization Thrust Augmentation
CEV	Crew Exploration Vehicle
CN	Converging Nozzle
COPV	Composite Over-Wrapped Pressure Vessel
CPF	Conventional Pressure Fed
DIO	Digital Input/Output
DN	Diverging Nozzle
FS	Feed System
GVA	Gas Valve Assembly
HOV	Hand Operated Valve
ICM	Integrated Control and Monitoring
JRC	Johnson Research Center
lbf	Pound Force
lbm	Pound Mass
LED	Light Emitting Diode
LH <sub>2</sub>	Liquid Hydrogen
LOX	Liquid Oxygen
LVA	Liquid Valve Assembly
MSFC	Marshall Space Flight Center
MTBF	Mean Time Between Failures
NASA	National Aeronautics and Space Administration
NI	National Instruments
OMS	Orbital Maneuvering System
PID	Partial Integration and Differentiation
PRC	Propulsion Research Center
psi	Pounds per Square Inch
psia	Pounds per Square Inch Absolute
psig	Pounds per Square Inch Gauge
RFS	Reciprocating Feed System
ROV	Remotely Operated Valve
SBB-100	Data Acquisition Box
scfm	Standard Cubic Feet per Minute
SFS	Sequential Feed System
SOV	Solenoid Operated Valve
SSME	Space Shuttle Main Engine
TP	Turbopump
UAH	University of Alabama in Huntsville
VBA	Visual Basic for Applications
VI	Virtual Instrument

<u>Computer Software</u>	
Auto Sequence	A computer-generated Sequence
Condition Sequence	A Sequence generated through response to sensor data
Cycle	SFS Cycle
Cycle Stage	Specific state of a SFS Run Tank during operation, e.g., Drain, Vent, Fill, Press, Idle
Drain Stage	SFS Cycle Stage: Liquid Flows out of COPVs into engine
File Sequence	A Sequence read from data on a file
Fill Stage	Fill1 and Fill2 SFS Cycle Stages, Liquid Flows into COPVs
Fill-1 Stage	SFS Cycle Stage: Fill with Vent Closed
Fill-2 Stage	SFS Cycle Stage: Fill with Vent Open
Idle Stage	SFS Cycle Stage: COPV is filled, pressurized, and awaiting the next Drain Stage
Leak Failure	A valve that does not close completely
Power Off Failure	A valve that receives an unexpected Power Off signal
Power On Failure	A valve that receives an unexpected Power On signal
Press	Pressurization
Press Stage	SFS Cycle Stage: Pressurant gas flows into COPVs to increase tank pressure
Run Tank	SFS COPV
Sequence	A collection of valve commands for the operation of the SFS
SFS Cycle	Sequence that brings the entire SFS back to its original state
Stage	Cycle Stage
Start Failure	A valve that fails to actuate correctly at the start of a Stage
Stop Failure	A valve that fails to actuate correctly at the end of a Stage
Tank Cycle	Series of SFS Stages that brings a Run Tank back to its original state
Vent Stage	SFS Cycle Stage: Expulsion of pressurant gas from COPV after a Drain Stage

## LIST OF SYMBOLS

<b><u>Term</u></b>	<b><u>Description</u></b>
<u>Engine Mass Estimation</u>	
$A_e$	Nozzle Exit Area in ft <sup>2</sup>
$A^*$	Nozzle Throat Area in ft <sup>2</sup>
$A_e / A^*$	Nozzle Expansion Ratio
$D_c$	Chamber Diameter in ft
$D^*$	Nozzle Throat Diameter in ft
$g_e$	Gravitational Acceleration on Earth
$I_{sp}$	Specific Impulse in seconds
$L_c$	Chamber Length in ft
$m_x$	Mass of Component X (e.g., $M_{barrel}$ ) in lbm
$\dot{m}$	Mass Flow Rate in lbm/s
$P_0$	Chamber Pressure in psia
$R$	Specific Gas Constant in ft-lbf/lbm-°R
$s_f$	Wall Material Safety Factor
$t$	Wall Thickness in ft
$T$	Thrust in lbf
$T_0$	Chamber Temperature in °R
$\gamma$	Ratio of Specific Heats
$\rho$	Wall Material Density in lbm/ft <sup>3</sup>
$\sigma$	Wall Material Tensile Strength in psia
<u>Launch Vehicle Simulation</u>	
$A$	Cross-Sectional Area (largest cross section of the rocket) in ft <sup>2</sup>
$C_D$	Coefficient of Drag
$dx$	Incremental change in property x (e.g., $dt$ or $d\theta$ )
$D$	Drag Force in lbf
$h$	Altitude in ft
$h_0$	Initial Altitude in feet
$m$	Rocket mass in lbm
$M$	Vertical Rocket Analysis: Mach Number
$P$	Pressure in psia
$PayloadGuess$	Vertical Rocket Analysis: Computer variable used to compute possible payload mass of the rocket, in lbm
$R$	Vertical Rocket Analysis: Mass fraction

Subscript i	Iteration Counter
$t$	Vertical Rocket Analysis: Time in seconds
$t_b$	Burn Time in seconds
$t_{endturn}$	Time to End Turn (when the simulation completes the turn) in seconds
$t_{startturn}$	Time to Start Turn (when the simulation begins calculating the turn from the Initial Flight Angle to the Final Flight Angle) in seconds
$t_{T2}$	Time to Start Second Phase in seconds
$t_{T3}$	Time to Start Third Phase in seconds
$T_0$	Initial Thrust in lbf
$T_2$	Second Phase Thrust in lbf
$T_3$	Third Phase Thrust in lbf
$\Delta u$	Change in velocity
$v$	Velocity in ft/s
$v_0$	Initial Velocity in ft/s
$v_{target}$	Target Velocity (desired final payload velocity) in ft/s
$v_x$	Velocity in the x direction, ft/s
$v_y$	Velocity in the y direction, ft/s
$w_e$	Empty Weight (rocket without propellant) in lbm
$w_i$	Inert Weight (non-propulsion structural weight) in lbm
$w_{misc}$	Propulsion System Weight (engine, fuel tanks, etc.) in lbm
$w_p$	Propellant Weight in lbm
$\theta_z$	Flight Angle in degrees
$\theta_{end}$	Final Flight Angle in degrees
$\theta_0$	Initial Flight Angle (measured from horizontal axis) in degrees
<u>SFS Reliability Analysis</u>	
M	Number of Feed Systems needed for overall success
MTBF	Mean Time Between Failures
N	Total Number of Feed Systems
$P(A \cap B)$	Probability of A and B, expression for reliability of equipments in series
$P(A \cup B)$	Probability of A or B, expression for reliability of equipments in parallel

$P_F$	Probability of failure
$P_S$	Probability of success
$P_X$	Probability of success of X (e.g., $P_{CPF}$ )
T#	Equipment Label for an SFS tank on a reliability diagram (e.g., T1, T2 or T3)
V#	Equipment Label for a valve on a reliability diagram (e.g., V1 or V2)
Z	Number of combinations of successful Feed Systems needed for overall success
$\sigma$	Standard Deviation
$\tau$	Normalized Probability
$Pr ob(\tau)$	Probability of a measurement falling in the range of plus or minus $\tau$ on the normalized Gaussian distribution
<u>SFS Vent Stage Analysis</u>	
A	Line Area in ft <sup>2</sup>
a	Speed of sound in ft/s
D	Line Diameter in ft
m	Pressurant mass in lbm
$m_i$	Initial mass of pressurant in SFS tank, in lbm
P	Pressure in psia
$P_{ext}$	External Pressure in psia
$P_f$	Final Pressure in psia
$P_i$	Initial Pressure in psia
Subscript 1	Initial or Total Conditions
Subscript 2	Final Conditions
Subscript n	Iteration counter
T	Temperature in °R
$T_i$	Initial Temperature in °R
$T^*$	Sonic Temperature
V	SFS Tank Volume in ft <sup>3</sup>
$\Delta x$	Change in property x (e.g., $\Delta P$ )
$\alpha$	Equation Constant
<u>SFS Refill Stage Analysis</u>	
Re	Reynolds number
D	Pipe inner diameter in ft
f	Friction factor
g	Acceleration in ft/s <sup>2</sup>
$\Sigma K$	Total Resistance
L	Pipe Length in ft

Subscript main	Property of main propellant tank
Subscript SFS	Property of SFS tank
$v_e$	Exit velocity in ft/s
$z$	Height in ft
$\mu$	Dynamic Viscosity in lbf-s/ft <sup>2</sup>
$\eta$	Viscosity in ft <sup>2</sup> /s
$\rho$	Propellant density
<u>SFS Cycle Timing Equations</u>	
D	Drain Time
F	Fill Time
FV	Fill/Vent Time
N	Number of Drains
O	Drain Overlap
P	Pressurization Time
V	Vent Time
<u>Miscellaneous</u>	
$C_v$	Valve Coefficient

# CHAPTER 1

## TECHNOLOGY BACKGROUND

Chemical rockets derive thrust through the expansion of hot gases through a nozzle. The two most basic types of chemical rockets are those using a solid propellant, and those using liquid propellants. Solid propellant rockets are driven by igniting a solid state compound that contains a mixture of fuel and oxidizer. The entire block of solid propellant that is used by the rocket is referred to as the grain. The thrust generated from this type of propellant depends primarily on the propellant chemical composition (how much chemical energy exists in the propellant to be released), the operating pressure (which determines the rate at which the propellant burns) and the grain geometry (which influences the amount of gas that is evolved).

Liquid propellant rockets differ from solid propellant rockets in that the propellants are stored in liquid form, and as a result there are many more options as to how liquid rockets are constructed. In a monopropellant system, a self-oxidizing compound is burned (such as  $\text{H}_2\text{O}_2$ , which decomposes exothermically). In a bipropellant system, the fuel and oxidizer are separately stored compounds that are burnt when injected together into the combustion chamber. Additionally, liquid rocket engines offer a great deal more control over how thrust is generated. Solid rocket engines typically must burn to completion, while liquid engines may be shut down, restarted and



throttled as needed. As such, liquid rocket engines are often used for missions that require maneuvers, engine throttling, and repeated engine restarts. This feature of liquid rocket engines also influences the design and construction of components used to transport the propellant to the engine.

Lastly, a hybrid rocket is a type of chemical rocket engine that combines solid and liquid propellants. These types of rockets function by using a solid state compound for one of the propellants (typically the fuel) and a liquid compound for the other. The solid grain will only burn when the liquid propellant is present. This allows the hybrid rocket to be controlled more readily than a traditional solid rocket, as the thrust can be throttled or shut off completely by controlling the flow of the liquid propellant.

## 1.1 Propellant Feed Systems

The components of a liquid rocket engine used to transport the propellants from the storage tank to the engine itself are referred to as the propellant feed system. This system is used to maintain the pressure in the propellant storage tanks to allow for the propellant to flow out of the tanks and into the engine, as well as pressurize the propellant (through the use of a pump or the internal tank pressure) to the required operating pressure for the engine.

Propellant feed systems have a number of basic designs and modifications to those designs; each system has its own advantages and drawbacks for specific missions. The two most general categories of propellant feed system are known as conventional pressure fed systems and turbopump systems.

### 1.1.1 Conventional Pressure Fed Systems

A conventional pressure fed system is typically characterized as a system that stores the propellants in high pressure tanks prior to flow into the engine. The storage pressure must be sufficiently high to ensure the desired engine operating pressure, or additional pressurant must be added to the storage tanks to maintain that pressure. Once the propellant storage tanks are pressurized, the valves between tanks and engine can be opened to allow propellant flow into the engine.

The variations in the conventional pressure fed concept are classified by how the pressurizing gas is generated. Some common variations include the following<sup>1</sup>:

**Polytropic Expansion or Blowdown System:** The propellant tanks are pressurized prior to engine operation, and no additional pressurant is added. As a result, the tank pressure will decay over time as the pressurant gas pushes more and more propellant out into the engine.

**Stored Gas System:** Pressurant gas (typically gaseous helium stored in high pressure bottles) flows through a regulator to keep the ullage pressure in the storage tanks constant. An optional heat exchanger may be used to control the pressurant temperature.

**Evaporated Propellant System:** The liquid propellant is heated (through use as a coolant on the engine nozzle or through other means) until it evaporates. The gaseous propellant is then sent back into the storage tanks and used as pressurant gas.

**Stored Liquid System:** The pressurant gas is stored as a liquid in high pressure bottles. The liquid is evaporated, and the resulting pressurant gas is sent into the propellant storage tanks to pressurize the propellant. Such a pressurant would have to be non-reactive with both oxidizer and fuel (for bipropellant systems).

Main Tank Injection System: A chemical is injected into the propellant tank that causes a small reaction. Gas produced from this reaction is used to pressurize the tank. An example of this is the injection of fluorine into a tank of liquid hydrogen. The result is a small amount of hydrogen fluoride and a large amount of gaseous hydrogen.

Solid Propellant Gas Generator System: A solid propellant cartridge is burned to generate the pressurant gas.

Of these variations to the conventional pressure fed concept, the Stored Gas system is the most used,<sup>2</sup> typically with helium as the pressurant. This configuration offers a great deal of simplicity to the feed system, and keeps the pressurant gas as light as possible while still remaining inert.

The disadvantage of the Stored Gas system, and indeed with all high pressure conventional pressure fed systems, is that operating the engine at high pressure requires the use of very massive propellant storage tanks. These tanks must be heavier than lower pressure tanks to maintain structural integrity during rocket operation. As a result, more mass of the rocket is tied up in the structure and the tanks. Typically this means the feed system is run at a lower pressure to decrease weight, but that in turn lowers the specific impulse and delivered payload. At higher chamber pressure, the specific impulse increases and the throat area decreases, allowing for higher expansion ratio, higher thrust, and a smaller, lighter engine. Real systems aim to have the highest possible pressure, and thus highest specific impulse, while still maintaining the performance for the required amount of delivered payload.

### 1.1.2 Turbopump Systems

A turbopump system is a propellant feed system that uses a hot gas turbine to drive a pump. This pump pressurizes the propellants prior to the engine inlet. In addition to the pressure needed to operate the engine, turbopump systems also include some features found in conventional pressure fed systems to maintain tank pressure. The reason for this is that the system must be designed such that the pressure of the propellant is kept high enough so that it remains in the liquid state as it passes through all valves and fittings. If that does not happen and cavitation occurs in the feed system lines upstream of the pump inlet, then the efficiency of the system will go down and some components may be damaged. Avoiding cavitation problems also sets a limit on what the maximum speed of the turbine shaft may be,<sup>3</sup> and that in turn affects how the pump will operate.

Turbopump cycles are classified into either open or closed cycles depending on how the propellant is moved through the engine components. In an open cycle, the working fluid is dumped overboard after it passes through the turbine. This can be done through a separate nozzle, or by routing the fluid into the main combustion nozzle downstream of the combustion chamber. In closed cycles, the working fluid flows directly into the combustion cycle with the rest of the propellant. In general, closed cycles give 1-5% better specific impulse because the turbine exhaust gas is expanded through the full pressure ratio of the main nozzle.<sup>3</sup> Expanding the gas through another nozzle, as is done in the open cycles, would still produce thrust, but would not be as efficient since the pressure ratio is not as high as is found in the main nozzle. The RS-68 (the largest liquid hydrogen/liquid oxygen rocket engine currently built)<sup>3</sup> is an example of

a closed cycle, while the RL10<sup>3</sup> is an example of an open cycle. Typically the more complex cycles are used for larger engines.

The most common types of turbopump cycles are illustrated in Figure 1.1,<sup>3</sup> and described below.

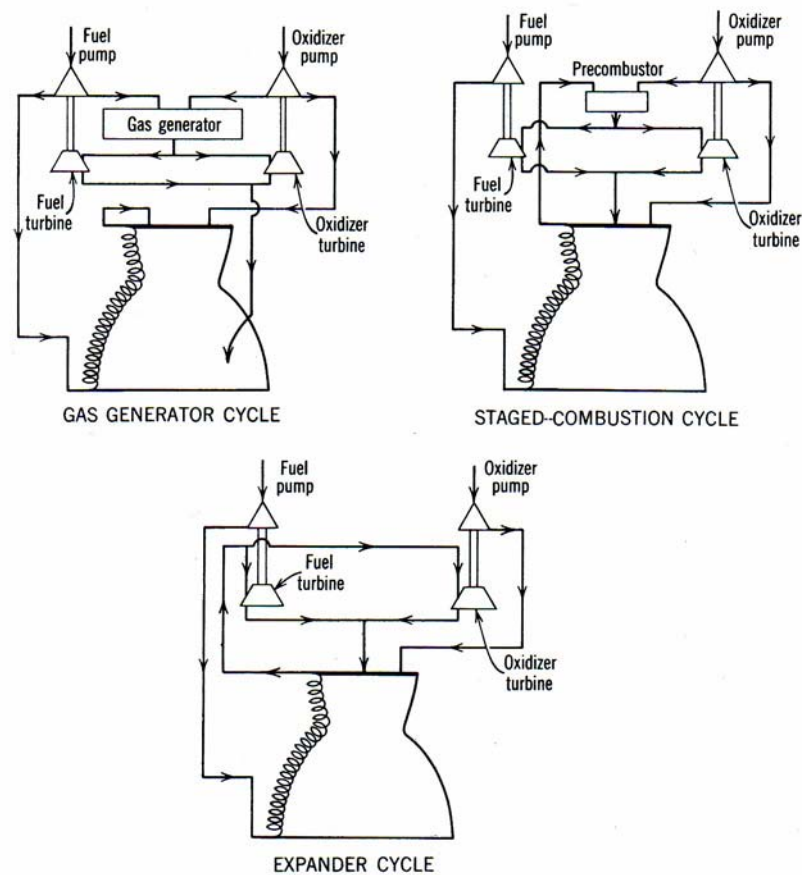


Figure 1.1 Turbopump Cycle Diagrams

**Gas Generator Cycle:** The gas used to operate the turbines which drive the fuel and oxidizer pumps comes from a gas generator. The gas generator may be supplied by bleeding propellant from the lines to the engine (as shown in Figure 1.1), or with

chemicals stored in separate storage tanks. In the latter case, the chemicals used for gas generator operation do not have to be the same as those used for propellant. Some systems use a separate monopropellant to generate gas for the turbine. Since in most cases this cycle is open, the turbine exhaust is vented overboard and will not have to react with the propellants in the main nozzle. The RS-68<sup>3</sup> uses this type of engine cycle.

**Expander Cycle:** One of the propellants (typically the fuel in a bipropellant system) is used as coolant for the main nozzle. After the coolant has been heated, it is passed through the turbine to drive the pumps. The turbine then exhausts into the combustion chamber, and the propellant is burned. The RL10<sup>3</sup> uses this type of engine cycle.

**Staged Combustion Cycle:** Like the expander cycle, this cycle requires one of the propellants to be used as coolant for the main nozzle. Before entering the turbine, the fuel and some of the oxidizer is combusted in a precombuster to generate high energy gas. The combustion allows for the gas going into the turbines to have higher energy than what could be achieved by simply using the energy transferred to the coolant. The turbine exhaust the flows into the main combustion chamber and is burned again with the rest of the oxidizer. The SSME<sup>3</sup> uses this type of engine cycle.

Each of these systems has advantages and disadvantages, compared to each other and compared to conventional pressure fed systems. In general, turbopumps are lighter than other propellant feed systems for high thrust applications. This is due to the fact that turbopumps can deliver high pressure propellant flow without requiring the main propellant storage tanks to be pressurized to high pressure. The disadvantage in this is

that turbopump systems are far more complex and costly than conventional pressure fed systems, and they pose operational and reliability challenges.

The specific advantages and drawbacks of the three common cycles<sup>3</sup> used in turbopump systems are as follows. The gas generator cycle is fairly simple (as turbopump systems go) and allows for lower tank pressure because the lines and pumps do not require as much pressure as in other cycles. This in turn allows for lower mass in the pump, turbine, and fluid lines. The disadvantage of this cycle is that it has a lower specific impulse than both the expander and the staged combustion cycles due to the propellant that is dumped overboard. The gas generator exhaust is typically vented through a sonic nozzle at low pressure, which means lower specific impulse.

The expander cycle uses coolant to directly drive the turbine. Specific impulse is not as high as the staged combustion cycle, but the mass of the system is lower because of lower pressures required to operate.

The staged combustion cycle has the highest specific impulse of all of the common turbopump cycles, but at some significant costs and increases to complexity. The precombuster adds an additional pressure drop into the system, and thus the pumps must pressurize the propellant slightly more than they would have to otherwise. As a result, the pump, turbines, and the fluid lines must be slightly heavier. This system is also more complex than the others, since both the fuel and oxidizer are used in driving the turbines, requiring more valves and a more complex flow path. Despite these drawbacks the high specific impulse of the system may lead to better vehicle performance under the right design conditions.

### 1.1.3 Other Propellant Feed System Types

For many applications of rocket engines, high specific impulse is desired. The specific impulse is a measure of how efficiently the rocket uses its propellant, as thrust is equal to the mass flow rate times the specific impulse. By increasing specific impulse, a rocket system can have higher thrust for the same mass flow rate, which would increase the amount of delivered payload. In terms of real systems and actual missions, delivered payload is the most important performance criterion. In order to achieve high specific impulse, the combustion chamber pressure is optimized such that it is as high as possible without the weight of the structure impairing the thrust generated. Achieving high pressure is one of the factors that increases the mass flow rate of propellant through the rocket engine, which increases the thrust produced. Higher pressure also requires more massive tanks or complex pumps. The propellant feed system is the component of the rocket that is responsible for determining what the chamber pressure will be, thus care must be taken in the design phase to select the system that will offer the highest pressure, and thus highest specific impulse, while still allowing the necessary payload to be delivered.

Tradeoffs between selecting a conventional pressure fed system and a turbopump for use in a propellant feed system include issues of simplicity of operation and maintenance, reliability, cost, and system performance as measured by delivered payload. Conventional pressure fed systems are inherently simple. They all operate under the principle of using pressurized gas to force the liquid propellants into the combustion chamber. The difficulty with this is that the storage tanks needed in such a feed system become heavier as the required pressure increases.



Turbopumps, on the other hand, eliminate the need for heavy, thick-walled propellant storage tanks because the pump delivers the propellant at the desired pressure. However, turbopumps are inherently complex because they require at least one turbine, large enough pumps to pressurize the propellants, a fluid line system to operate both the turbine and pump the propellant into the combustion chamber, and a means of driving the turbine. They also involve more complex operations, such as restart, thermal conditioning, and chill down. Turbopump maintenance is complex and costly in multiple use vehicles such as the SSME. For example, it takes over sixty days<sup>4</sup> to refurbish the entire Space Shuttle, including the SSME turbopumps, on the ground. In space, maintenance of turbopumps would be even more challenging.

Currently there are two other propellant feed systems under development that seek to minimize the current complexity of these trade-offs in choosing a propellant feed system: the Pistonless Pump<sup>5</sup> and the Sequential Feed System<sup>6</sup> (SFS, previously termed the Reciprocating Feed System or RFS). The Pistonless Pump<sup>7</sup> uses two differently sized high pressure tanks that are significantly smaller than the main propellant tanks. Propellant is pressurized to high pressure in the two tanks, and then expelled from either one or the other. The smaller of the two tanks is used to maintain propellant flow while the larger refills from the main storage tank. The SFS is the subject of this thesis, and the SFS concept is described in the next section. Both of these new propellant feed systems operate by using small high pressure tanks and gas pressurization to deliver high pressure liquid propellant to an engine without the need of large high pressure main propellant tanks or complex turbomachinery.

The Pistonless Pump and the SFS are in several regards fundamentally the same, but with several differences. The SFS can be used with three tanks, and in such a configuration if any valve or tank fails it can continue to operate on the remaining two tanks. The three tanks used in the SFS are identical volume, and as such the flow times to fill, drain, vent, and pressurize are equal, whereas the Pistonless Pump has two tanks, one inside the other. This configuration leads to pressure and flowrate variations when propellant flow switches between the two tanks and also poses more complexity and access issues to the system.

## 1.2 Sequential Feed System Concept

The Sequential Feed System<sup>6,8,9</sup> is a propellant feed system that delivers high pressure liquid flow to a rocket engine by means of sequentially draining a number of high pressure liquid propellant tanks, referred to as run tanks. In the typical SFS arrangement, three high pressure run tanks are located between the main propellant storage tank and the engine. The main storage tank is kept at low pressure, and each run tank is filled from the main tank. Once filled, the run tanks are pressurized from high pressure gas storage bottles. The pressurizing gas can be helium or gaseous hydrogen or oxygen, as discussed in more detail later.

During operation of the engine, only one run tank is draining at any given time. When a run tank is emptied, the valve between it and the engine is closed and the valve between the next run tank and the engine is opened, allowing it to drain. The switch between run tanks is designed such that the decrease in drain rate from the first tank and the increase in drain rate from the second tank produce a virtually continuous drain rate

into the engine. This process repeats for the third run tank when the second tank runs to completion. At the end of the burn time, the system will close all drain valves to the engine and refill all three tanks so that they are ready to begin again.

The run tanks are small enough that they cannot contain all of the propellant necessary for the engine burn, so the system will begin refilling the run tanks after they empty and the next tank in the sequence takes over. The entire refill process is designed to be completed before the next run tank is drained completely. For example, when run tank 1 completes its drain, run tank 2 will begin draining. Run tank 1 will vent the excess pressurant gas, and then refill with liquid from the main tank. After run tank 1 is full of liquid, the valve to the pressurant bottles will open, and the propellant in run tank 1 will be pressurized. This should all occur before run tank 2 finishes draining.

The SFS has an important advantage in that it has an inherent fail-operational mode. If a valve on any of the three run tanks fails, then the system can continue to operate by alternating between the remaining two tanks. This is the key feature that separates the SFS concept from the Pistonless Pump.<sup>5</sup> In the Pistonless Pump design, the second tank exists only to continue the propellant flow while the first tank is refilled. If the valves controlling either tank fail, then the system cannot continue to operate. With the SFS, the run tanks are symmetrical in volume so that any two tanks can take over for the three tank system should a failure occur. The Pistonless Pump system may then be considered a simplified version of the SFS without the fail-operational mode. However, in principle two or more sets of the Pistonless Pump system could be used to create a fail-operational mode.

Additional modifications to the SFS concept that have been considered include the use of more than three run tanks, the use of other pressurization schemes (other than stored inert gas) for the run tanks, and the Autogenous Pressurization Thrust Augmentation (APTA) SFS. With more than three run tanks, an additional fail-operational mode is added with each extra run tank. For example, an SFS with four tanks could fail over to three tank operation if one tank has a failure, and then fail over again to two tank operation if one of the remaining tanks suffered a failure. Adding additional tanks of course increases the mass and complexity of the system. The typical system that has been considered and tested only has three tanks because ideally the valves should be constructed well enough to not fail under normal operating conditions, but any single failure in a valve would not be catastrophic. This meets the standard design guideline for rocket systems in that no single credible failure should result in a mission failure.

The method used to pressurize the SFS run tanks would be determined based on overall mission requirements. Using stored helium gas as a pressurant is convenient in many cases because the pressurant will not react with any propellant compound and is the lightest pressurant gas that is also non-reactive. Depending on the application, other pressurization methods may be preferable, and there is nothing in the SFS concept that requires a specific method. The means of maintaining ullage pressure in a pressure tank (as is done with conventional pressure fed systems, see Section 1.1.1) can easily be modified to pressurize the small SFS run tanks.

The Autogenous Pressurization Thrust Augmentation SFS concept uses gaseous propellants as the pressurant gas, as opposed to an inert gas like helium. This kind of pressurant may be obtained in a variety of ways, but the design currently considered has

been that the pressurant would be stored as a liquid in separate tanks. These pressurant bottles would be pressurized to approximately 20 psi above operating pressure (to allow for pressure drops in the system) and the liquid would be vaporized before flowing into pressurize the run tanks. The advantage with this concept is that when a run tank finishes draining, it is not necessary to simply vent the excess pressurant overboard. Fuel and oxidizer propellant may be vented into a combustion chamber and burnt in order to add to the thrust of the vehicle or act as a reaction control system. At this time  $\text{LH}_2/\text{LO}_2$  is the only propellant combination that has been analyzed using the feed system sizing code developed for the SFS.<sup>10</sup> However the basic principle could be used with other propellant combinations. Due to the nature of the APTA SFS, propellant combinations have to be chosen with a mind to how the pressurant will react with the propellant. Analysis of this concept<sup>10</sup> for the propellant combination of  $\text{LH}_2/\text{LOX}$  shows that there is a significant advantage in delivered payload when the pressurant gas can be used as a source of thrust.

The main advantages of the SFS as a propellant feed system are that it allows for low mass, low pressure propellant storage tanks and high pressure propellant flow to the engine, with the resulting increase in specific impulse and decrease in engine mass. In addition, it is a very simple system that is competitive with those currently available in terms of delivered payload<sup>11</sup> and system reliability. Pressurizing the propellants in the smaller run tanks eliminates the need to keep those propellants at high pressure at all times. This means there is no need for massive high pressure propellant storage tanks, and it allows the engine to be run at high pressure (with higher specific impulse than other types of feed systems). The system itself consists only of very basic and proven technologies: tanks and valves. Combined with an inherent fail-operational mode and the

APTA modification to enhance system performance, the SFS offers a substantial improvement over current propellant feed systems.

#### 1.2.1 Previous Experimental Work on the SFS

The previous experimental work on the SFS was conducted at the University of Alabama in Huntsville<sup>10,12</sup> (UAH). This previous project conducted proof-of-concept testing for the SFS, showing that the pressure and flowrate variations during run tank switch over were minimal, on the order of plus or minus 2-5 percent. These tests were conducted at the NASA Marshall Space Flight Center's (MSFC) North Test Area.

Figure 1.2 depicts the SFS test apparatus that was used during testing. A triangular frame was used to support the three Composite Over-Wrapped Pressure Vessels (COPVs) that were originally rated at approximately 2,800 psi, together with as the three drain, fill, and pressurization manifolds and the four valves on each tank. The frame and the COPVs are shown during initial construction in Figure 1.2.



Figure 1.2 SFS Test Apparatus

The North Test Area at NASA MSFC was selected as the testing site because of the available 10,000 gallon tank of deionized water (seen in the background of Figure 1.3) that served as the low pressure supply tank. Missile-grade air was also available and was used as the pressurant gas after passing through a pressure regulating panel to maintain the pressure at 500psi. The work conducted at NASA MSFC was

partly supported by a Space Act Agreement with UAH. The SFS test bed, as set up at the North Test Area, is shown in Figure 1.3.



Figure 1.3 SFS Test Bed at NASA MSFC North Test Area

The test matrix that was used is shown in Table 1.1.<sup>10</sup> A key issue that was investigated during the course of the proof-of-concept testing was the effect changing run tanks had on the common SFS outflow pressure and flowrate. The timing of the fill, drain, and vent stages of the SFS cycle were determined through calibration tests. In an



operational SFS, liquid level sensors would be used, but such equipment was unavailable for the proof-of-concept test program. After the calibration tests, complete SFS cycle tests were conducted to examine how the pressure and flowrate varied whenever a switch from one run tank to another occurred. Alterations in cycle timing and some changes to the apparatus hardware were made to minimize the observed pressure and flowrate changes, both in the duration of the change and how extreme it was.

Table 1.1 Test Matrix Used During Previous SFS Testing

<b><u>Step</u></b>	<b><u>Process</u></b>	<b><u>Desired Result</u></b>
1. Fill Cycle Calibration	Fill tanks for prescribed periods of time. Manually drain, capture, and record water volume	Characterization of the fill rate of each of the run tanks
2. Drain Cycle Calibration	Fill tanks to desired volume. Pressurize and drain for prescribed periods of time. Vent. Manually drain, capture, and record water volume	Characterization of the drain rate of each of the run tanks
3. Vent Cycle Calibration	Obtain pressure versus time data from the drain calibration data files	Characterization of the vent rate of each of the run tanks
4. Single SFS Cycle	Use calibration data to alternately drain pressurized flow from all run tanks. Make timing/hardware changes as necessary	Steady, pressurized outflow during run tank drain transition
5. Two SFS Cycles	Use data from previous step to alternately perform two RFS sequences of each run tank	Steady, pressurized outflow during run tank drain transition
6. Several SFS Cycles	Use data from previous steps to alternately perform multiple RFS sequences of each run tank	Steady, pressurized outflow during run tank drain transition

The results the SFS testing are shown in Table 1.2.<sup>10</sup> After several test runs, it was determined that it was necessary to install check valves on the drain lines to prevent the pressure differentials between run tanks from decreasing the outflow pressure and causing flowrate variations. The ability of the test bed to supply pressurant flow to the run tanks was also determined to be a limiting factor in how steady the outflow pressure could be. Another modification that was done to reduce pressure variations was the overlapping of the drain and pressurization cycles. The end result was a test run with a flow rate fluctuation that was only 1.5% and lasted 1.2 seconds. This was a substantial improvement over earlier test runs that had fluctuations of 5-7%. It was concluded that the SFS system could be configured to minimize outflow fluctuations by valve overlap and that the concept was adequately proven. With the addition of a small accumulator tank, even lower pressure and flowrate variations would occur.

Table 1.2 Experimental Results of SFS Testing

<b>Time tank was pressurized prior to drain</b>	<b>Time drain cycles were overlapped</b>	<b>Time pressurization cycles were overlapped</b>	<b>Check valves installed?</b>	<b>Flow rate fluctuation</b>	<b>Flow rate fluctuation duration</b>	<b>Set pressure</b>
1.00 sec	0.50 sec	1.50 sec	No	5%	2.40 sec	500 psig
1.00 sec	0.50 sec	0 sec	No	7%	1.71 sec	505 psig
0.40 sec	0.50 sec	0 sec	No	5%	1.12 sec	500 psig
0.60 sec	0 sec	0 sec	No	5%	0.95 sec	445 psig
0.60 sec	0 sec	0.60 sec	Yes	5%	1.4 sec	485 psig
0.60 sec	0.15 sec	0.75 sec	Yes	3%	1.3 sec	485 psig
0.60 sec	0.25 sec	0.85 sec	Yes	2%	1.1 sec	485 psig
0.60 sec	0.50 sec	1.10 sec	Yes	1.5%	1.2 sec	485 psig

### 1.2.2 Previous Analytical Work on the SFS

The previous analytical work is in an Excel spreadsheet used for mass analysis of the SFS,<sup>11</sup> conventional pressured fed, and turbopump feed systems. The overall goal of this spreadsheet was to create a tool that could be used to compare the total system mass, inert mass, and delivered payload for the different propellant feed systems. The primary comparison metric was that of delivered payload, and the rocket systems modeled by the analysis were for in-space propulsion. This spreadsheet was unique in that it was designed to allow comparisons of various feed system types (conventional pressure fed, turbopump, SFS, and modifications of those basic designs). There are many computer codes in existence used to calculate performance of specific feed systems, but little has been done in the way of one code that can be used to compare feed systems in a trade study. The SFS analysis spreadsheet was intended to be used for trade studies and demonstrate the advantages of the SFS over other feed systems with similar mission requirements.

The spreadsheet used several common inputs for all systems, the basic mission parameters such as propellant combination, mixture ratio, specific impulse, burn time, and change in velocity. Using these general mission parameters and more specific user-supplied data, such as material properties, the program calculated the masses and dimensions of the various components, such as tanks and lines, for the feed systems. The mass of the propellant required was determined by the mission parameters, and with the structural mass accounted for by the feed system calculation, the program was then able to make an estimate of what the deliverable payload for an actual rocket of the calculated dimensions would be. The mission parameters were kept constant for all feed systems, as

was the propellant mass. This allowed a more reasonable comparison of feed system performance: which system using the same amount of propellant and achieving the same change in velocity would deliver the most payload.

Each feed system type would have different structural masses and specific details, but by keeping the propellant mass the same for all systems, the final result of delivered payload could be accurately compared for different system architectures with the same change in velocity. A representative screenshot of the spreadsheet interface is shown below in Figure 1.4; a SFS propulsion system is shown.

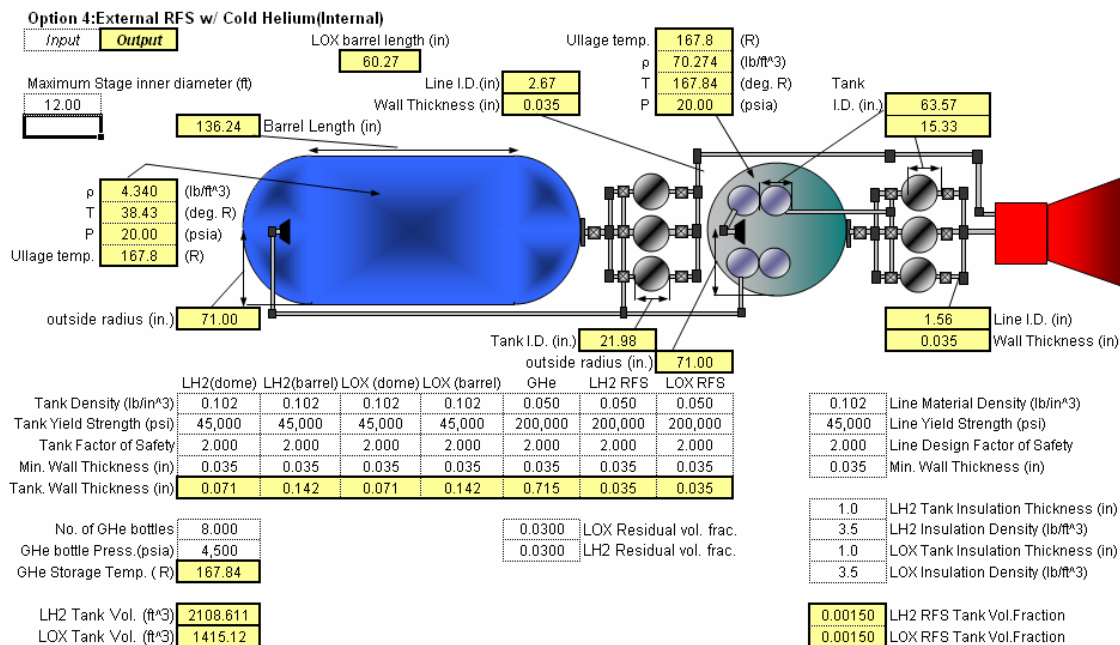


Figure 1.4 Screenshot of Propellant Feed System Analysis Spreadsheet

The spreadsheet was not formally validated, but accuracy was checked by inputting data for the Space Shuttle OMS, Apollo Lunar Ascent, and Apollo Lunar Descent systems into the spreadsheet. The equations used by the program were obtained from the available literature, as well as curve fits of past system data. Using these equations and the data from the aforementioned systems, the spreadsheet was able to predict the delivered payload results for all three systems with significant accuracy, some outputs were within a few units of the actual numbers, as shown with the summary of results from the Shuttle OMS simulation in Table 1-3.<sup>10</sup>

Table 1.3 Analysis Spreadsheet Validation Results: Shuttle OMS

Major Inputs for OMS Pod Simulation		Major Outputs	Simulation	Actual OMS
6,000 lbf	Avg. Thrust	N2O4 Tank mass	249 lb	250 lb
313 sec	Avg. Specific Impulse	MMH Tank mass	251 lb	250 lb
630 sec	Burn Time	GHe Tank mass	261 lb	272 lb
1.65	Mixture ratio	N2O4 propellant mass	7,519 lb	~ 8,000 lb
4,800 psia	GHe storage Pressure	MMH propellant mass	4,557 lb	~ 5,000 lb
125 psia	Chamber Pressure	N2O4 Tank Volume	89.85 ft <sup>3</sup>	89.89 ft <sup>3</sup>
0.102 lb/in <sup>3</sup>	MMH Tank Density	MMH Tank Volume	90.43 ft <sup>3</sup>	89.89 ft <sup>3</sup>
45,000 psi	MMH Tank Yield Strength	GHe Tank Volume	16.87 ft <sup>3</sup>	17.03 ft <sup>3</sup>
2.00	MMH Tank Factor of Safety	Engine mass	41 lb	260 lb
0.102 lb/in <sup>3</sup>	N2O4 Tank Density			
45,000 psi	N2O4 Tank Yield Strength			
2.00	N2O4 Tank Factor of Safety			
0.075 lb/in <sup>3</sup>	GHe Tank Density			
125,000	GHe Tank Yield Strength			
2.00	GHe Tank Factor of Safety			

With these results being in close agreement with the actual data, the code was concluded to be sufficiently valid for trade study comparisons. The main analysis work

comparing the SFS with other propellant feed systems was done by doing a comparison of possible propulsion systems using the Crew Exploration Vehicle (CEV)<sup>13</sup> configuration as a baseline. The systems that were considered were conventional pressure fed systems, turbopumps, a standard SFS, and an APTA SFS. The results of these analyses are summarized in Table 1.4.<sup>10</sup> The CEV baseline was a LOX/Methane conventional pressure fed system at 250psi.

Table 1.4 Analytical Results for CEV Feed System Comparison

Propellants	Conventional Pressure fed		RFS		Conventional Turbopump		Autogeneous	
	Chamber Pressure (psia)	Payload (lbm)	Chamber Pressure (psia)	Payload (lbm)	Chamber Pressure (psia)	Payload (lbm)	Chamber Pressure (psia)	Payload (lbm)
LO <sub>2</sub> /CH <sub>4</sub>	250	2,919	250	5,830	250	5,822	-	-
	500	570	500	6,193	500	6,217	-	-
	1,000	0	1,000	5,867	1,000	6,141	-	-
LO <sub>2</sub> /LH <sub>2</sub>	250	0	250	8,726	250	9,277	250	10,311
	500	0	500	8,593	500	9,583	500	10,807
	1,000	0	1,000	7,242	1,000	9,460	1,000	10,873
LO <sub>2</sub> /RP-1	250	1,841	250	4,656	250	4,587	-	-
	500	0	500	5,018	500	4,938	-	-
	1,000	0	1,000	5,503	1,000	5,452	-	-
MMH/N <sub>2</sub> O <sub>4</sub>	250	2,283	250	4,249	250	4,450	-	-
	500	488	500	4,351	500	4,699	-	-
	1,000	0	1,000	3,662	1,000	4,502	-	-

As can be seen from the results, the calculated delivered payload for the SFS is greater than the conventional pressure fed system for all propellant combinations and all pressures (the column labeled “RFS” is the SFS data, the old name for the system was used for this analysis). Turbopumps are also better than the conventional pressure fed systems, but it is important to note that the SFS is competitive with turbopump systems as

well. In addition, the APTA SFS offers the highest payload of any system considered in this analysis. The maximum APTA SFS payload of 10,873 lbs is 13% higher than the maximum delivered payload of the turbopump, 9,583 lbs. These results demonstrate the advantages of the SFS in one case. The results are limited to feed systems analyzed with the sizing code and using the CEV baseline assumptions.

It should be noted that the previous analytical work relied on the basic assumption that the propellant feed systems were all part of a spacecraft, which assumed that the engines were operating without drag or gravity. Other limitations include problems with using curve fits of existing data (such as engine mass) giving inaccurate results, and the fact that the spreadsheet does not simulate launch vehicles. Expanding upon this spreadsheet has been the key focus of the current analytical work. Chapter 2 describes the analysis projects completed to expand the spreadsheet and to investigate other SFS features.

Current feed system technologies used for rocket engines have trade-offs that must be evaluated when deciding which system to use for a specific mission. Conventional pressure fed systems benefit from low complexity and reduced cost, but cannot provide high pressure propellant flow to an engine without incurring a mass penalty. Turbopump systems do not require as much structural mass when operating at high pressure, but they are complex and costly pieces of equipment.

The Sequential Feed System is a new feed system concept that mitigates some of the drawbacks observed in other feed system types. The SFS is simple, low cost, and can provide high pressure flow without significant structural mass. To demonstrate the capabilities of the SFS, analytical and experimental work was conducted at UAH.<sup>10</sup> The

main analytical project was the development of the feed system sizing code to allow performance comparisons of the SFS to other feed systems. This code was validated with data from real rocket systems, and was then applied to CEV baseline specifications. The results of this analysis showed that the SFS provided superior performance to the conventional pressure fed system, and was better or competitive with turbopump systems in many of the cases examined. Even in the cases where the SFS did not provide a performance benefit over a turbopump system, the low-cost and increased reliability aspects of the SFS are points of consideration for the selection of a feed system for a real mission. To further demonstrate the abilities of the SFS, experimental testing was conducted to prove the concept with a full-scale test bed constructed and run at MSFC. This testing was done using air and water as simulated pressurant and propellant. The testing showed that the SFS concept would work, and that steady high pressure outflow of propellant could be achieved.

The topic of this thesis is the continuation of the SFS development program at UAH. The analytical projects that were done are discussed in Chapter 2, and include expansions to the SFS sizing code (improved mass estimation and addition to the code to allow for vertical launch simulations), analysis of SFS reliability as compared to other feed system types, and first-order simulations of SFS operation. The experimental aspect of this thesis continues from the reliability analysis, in that the fail-operational capability of the SFS was experimentally verified. To this end, a new test bed control program was written (see Chapter 3), the SFS test bed was reconstructed at UAH (see Chapter 4), and testing was done to examine the effects various valve failure modes had on the system's ability to provide steady outflow. The results of these tests are discussed in Chapter 5.



## CHAPTER 2

### ANALYTICAL WORK

Analysis conducted as part of this effort include, engine mass estimation (Section 2.1), launch vehicle performance (Section 2.2) used with the feed system sizing code,<sup>10</sup> SFS system reliability (Section 2.3), and the times necessary for venting (Section 2.4) and refilling (Section 2.5).

#### 2.1 Engine Mass Estimation

The system sizing mass and performance analysis spreadsheet<sup>10</sup> compares the performance of different feed systems for the same basic parameters of thrust, mission velocity change and the mass of propellant. To improve this analysis, the engine mass estimating relationship was refined. Estimating engine mass can be difficult because what counts towards engine mass varies between manufacturers. For the purpose of making a simplified and consistent mass estimation, a model was developed by which the engine mass of a rocket could be calculated based on the rocket performance parameters and the material properties of the engine.

### 2.1.1 Model and Derivation of Equations

The model used for the engine mass calculations was made intentionally simple, representing the most ideal kind of rocket engine. The calculation model assumes the mass of the engine comprises a combustion chamber with a cylindrical body and flat head, and a converging-diverging nozzle, both parts of which are truncated cones. A schematic of this model is shown in Figure 2.1.

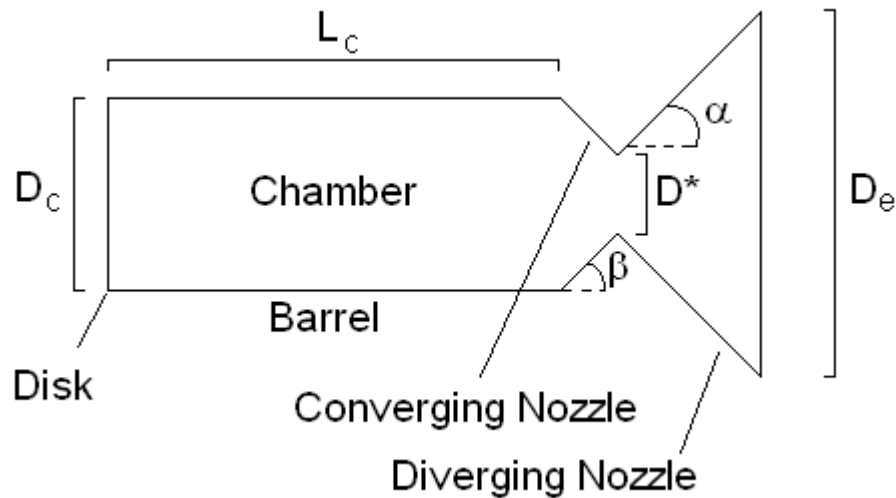


Figure 2.1 Engine Mass Estimation Model

It was decided to use this oversimplified model of the engine for two reasons: ease of calculation and validity of comparisons. Further analyses are required to determine the masses of auxillary hardware such as main engine cutoff valves, thrust vector control actuators, mounting structures, and, as appropriate, the turbopump assembly. All of thses components are included in real engines, and make up part of the

engine mass. The problem as far as the analysis spreadsheet is concerned is making a comparison between different feed system types. For example, an engine using a conventional pressure fed or SFS would not have some of its mass tied up in a turbopump assembly. The old engine mass estimating relationship used in the analysis spreadsheet<sup>10</sup> was derived from actual engine data, but application of that relationship to a general engine resulted in poor accuracy. The simplified engine model only includes the components which can reasonably be assumed to be on all engine types. Thus a comparison of engine mass using this model is valid no matter the feed system used.

Evaluating the mass of an engine with the assumed geometry can be further simplified if certain assumptions are made. First, it is assumed that the same material is used to make the walls of all parts of the engine. Second, the wall thickness in the barrel section of the combustion chamber is assumed to be the highest (because that is where the highest pressure is), and that thickness is used in all of the other sections of the engine. And third, the diameter and length of the combustion chamber are multiples of the nozzle throat diameter,  $D^*$ .

With these assumptions, the analysis is a matter of finding  $D^*$  from the rocket operation parameters (primarily thrust, chamber pressure, and expansion ratio), putting chamber diameter ( $D_c$ ) and length ( $L_c$ ) in terms of  $D^*$ , evaluating the wall thickness from the chamber pressure and the wall material properties, and then calculating the mass from the geometry of the engine. The necessary parameters needed for this calculation are summarized in Table 2.1.

Table 2.1 Engine Mass Input Parameters

<b>Parameter</b>	<b>Symbol</b>
Chamber Pressure	$P_0$
Thrust	$T$
Specific Impulse	$I_{sp}$
Chamber Temperature	$T_0$
Ratio of Specific Heats	$\gamma$
Specific Gas Constant	$R$
Expansion Ratio	$A_e / A^*$
Wall Material Density	$\rho$
Wall Material Tensile Strength	$\sigma$
Wall Material Safety Factor	$s_f$

The calculation of engine mass can be derived as follows. First, the mass flow rate,  $\dot{m}$ , is calculated as

$$\dot{m} = \frac{T}{I_{sp} g_e} . \quad (2.1)$$

Next the nozzle throat area,  $A^*$ , is determined from the mass flow rate using the expression<sup>18</sup>

$$A^* = \frac{\dot{m}}{P_0} \cdot \sqrt{\frac{RT_0}{\gamma} \cdot \left( \frac{2}{\gamma+1} \right)^{\frac{-(\gamma+1)}{\gamma-1}}} . \quad (2.2)$$

The expansion ratio and  $A^*$  are used to calculate nozzle exit area

$$A_e = A^* \frac{A_e}{A^*} . \quad (2.3)$$

Next the throat diameter,  $D^*$ , can be calculated if a circular geometry is assumed. At this point, a diameter of the chamber,  $D_c$ , is also assumed based on the nozzle diameter. The equations for these quantities are

$$D^* = \sqrt{\frac{4A^*}{\pi}} \quad (2.4)$$

and

$$D_c = 5D^*. \quad (2.5)$$

The length of the cylindrical chamber,  $L_c$  (ft), is then assumed to be

$$L_c = 1.5D_c = 7.5D^*. \quad (2.6)$$

Note that the constants in Equations (2.5) and (2.6) are somewhat arbitrary, suggested by the literature.<sup>14</sup> For different applications the user of the spreadsheet may modify these constants as needed, but the default values used in the analysis are as shown in the Equations.

Wall thickness of the cylindrical combustion chamber,  $t$ , is given as

$$t = \frac{2P_0 D_c}{\sigma} s_f. \quad (2.7)$$

Taking the thickness to be constant, an initial mass calculation is made based on the specified geometry. The equation for this is simply the sum of the component masses: disk (at the end of the chamber), barrel, converging nozzle (CN), and diverging nozzle (DN). These equations are

$$M_{disk} = \rho s_f t \cdot \frac{25\pi \cdot D^{*2}}{4}, \quad (2.8)$$

$$M_{barrel} = \rho s_f t \cdot 5\pi \cdot L_c D^*, \quad (2.9)$$

$$M_{CN} = \rho s_f t \cdot 12\pi \cdot D^{*2}, \quad (2.10)$$

and

$$M_{DN} = \rho \cdot s_f t \cdot \pi D^{*2} \left( 2.868 \frac{A_e}{A^*} - 2.868 \right). \quad (2.11)$$

The constants in Equation (2.10) and Equation (2.11) arise from the assumption that the nozzle convergence half angle ( $\beta$  in Figure 2.1) is  $60^\circ$  and the divergence half angle ( $\alpha$ ) is  $15^\circ$ . These assumptions, along with the assumptions for Equations (2.5) and (2.6), were based on the guidelines of Krzycki.<sup>14</sup>

Adding the terms of Equation (2.8) through Equation (2.11) and putting in the definition of  $A^*$  from Equation (2.2) and Equation (2.4) results in the final expression for engine mass:

$$M_{calc} = \frac{80 \cdot s_f}{\sqrt{\pi P_0}} \cdot \frac{\rho}{\sigma} \cdot \left( \frac{T}{I_{sp} g_e} \right)^{3/2} \cdot \left[ \frac{RT_0}{\gamma} \left( \frac{2}{\gamma+1} \right)^{\frac{-(\gamma+1)}{\gamma-1}} \right]^{3/4} \cdot \left[ 52882 + 2.868 \frac{A_e}{A^*} \right]. \quad (2.12)$$

### 2.1.2 Implementation and Results

To evaluate this model against data from real engines, the equations were coded into an Excel spreadsheet named Mass Project. A screenshot of the calculation section of this spreadsheet is shown in Figure 2.2. Most of the calculations are handled in the cell

formulas, and the Auto Calculate button is used to import engine data from later in the spreadsheet.

	A	B	C	D	E	F	G	H	I
1	Inputs			Outputs			Masses		
2									
3	Chamber Pressure (psia)	1015		Mass Flow Rate (lbm/sec)	177.7707		Top Disk	679.7058301	
4									
5	Thrust (lbf)	1740162		Throat Area (ft <sup>2</sup> )	0.416682		Barrel	4078.23498	
6									
7	Isp (sec)	304		Exit Area (ft <sup>2</sup> )	6.666915		Converging Nozzle	1305.035194	
8									
9	Chamber Temperature (deg R)	6606		Throat Diameter (ft)	0.728564		Diverging Nozzle	4678.551169	
10									
11	Gamma	1.24		Barrel Diameter (ft)	3.64282		Total	10741.52717	
12									
13	Specific Gas Constant (ft-lbf/lbm deg R)	7.65		Barrel Length (ft)	5.464229				
14							Percent Error	-41.93455228	
15	Expansion Ratio	16		Wall Thickness (ft)	0.130499				
16									
17	Wall Material Density (lbm/ft <sup>3</sup> )	500					Calculated Mass	10741.52717	
18									
19	Wall Material Tensile Strength (psia)	85000							
20									
21	Wall Material Safety Factor	1.5		Auto Calculate					
22									
23	Target Mass (lbm)	18499							
24									

Figure 2.2 Mass Project Spreadsheet Calculation Section

The input parameters for the Excel calculations use the following assumptions.

First, chamber temperature ( $T_0$ ) is the combustion temperature of the propellants.

Second, the specific gas constant ( $R$ ) and ratio of specific heats ( $\gamma$ ) are both constant and depend on the initial composition of the propellants. Lastly, the following numerical assumptions were made in evaluating the equations. Wall material density ( $\rho$ ) is 500 lbm/ft<sup>3</sup>, wall material tensile strength ( $\sigma$ ) is 85,000 psia (both of these values based on CRES 304 Stainless Steel<sup>15</sup>), and the wall material safety factor ( $s_f$ ) is 1.5.

While the spreadsheet is structured to evaluate the mass of a rocket engine with arbitrary input parameters, comparisons with real systems can be made if the necessary input parameters can be found. The data<sup>16</sup> shown in Table A.2 and Table A.3, are taken from the Mass Project spreadsheet. When the Auto Calculate button is pressed, an Excel macro is activated that imports the input parameter data into the cells shown in Figure 2.2. These cells then calculate the mass, and the macro copies the results from those cells into the output columns of the same row that the input data came from. This method was developed to automate the data entry for the program.

Using these data a comparison could be made between the analytical model and the reported mass of each engine. The inherent inaccuracies of the model are very significant. This was expected, as the model was intentionally simple and what constitutes engine mass varies from engine to engine and manufacturer to manufacturer. Since the overall goal was not to get an exact mass prediction, it was decided to plot the calculated masses for each engine against the real engine mass data. This plot, along with a linear curve fit, is shown in Figure 2.3.

The individual data points on Figure 2.3 represent engines. The real mass, as reported in the literature<sup>16</sup> for that engine is represented on the y-axis, and the calculated mass (using the data from that engine in the engine mass model) is on the x-axis. The “ $y=x$ ” curve fit line is shown to represent the line that engines would fall on if their real mass was accurately modeled by this estimation. As can be seen from the data, the real engines did not fall on this line, as was expected.



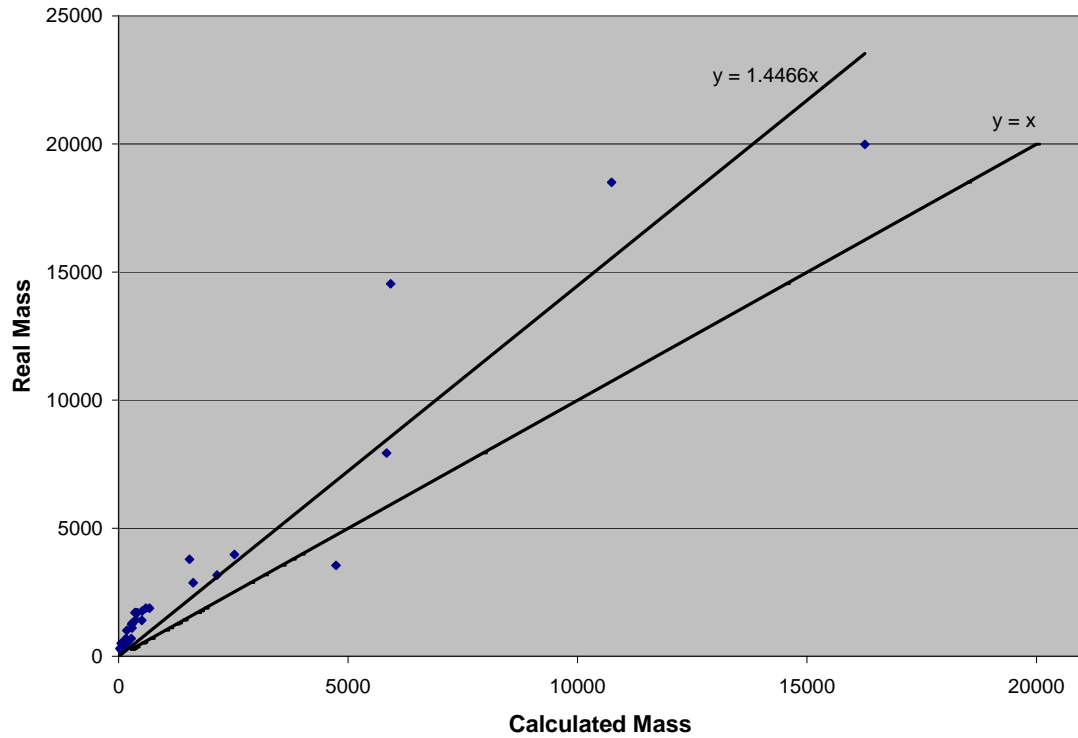


Figure 2.3 Real vs. Calculated Engine Masses

By running a linear curve fit on the real mass versus calculated mass a correction factor of 1.4466 can be obtained. This correction factor can be viewed as what, on average, the calculated mass needs to be multiplied by to arrive at a engine mass value close to what would be observed on a real system. The mass estimation relations previously derived can then be modified with this correction factor to get the final mass estimate of

$$M_{est} = 1.4466M_{calc} . \quad (2.13)$$

While the model and equations developed in this analysis do have their limitations and are not exceptionally accurate, the results of this project are useful in generating a first-order estimate of engine mass. This approach is also superior to the way engine mass was guessed at on the feed system sizing spreadsheet<sup>10</sup> in that it depends on the rocket performance parameters already in the spreadsheet and can be integrated into the existing code by modifying the cell formulas.

### 2.1.3 Alternate Formulation

One drawback of the engine mass model as it was originally developed was that it required the expansion ratio as an input parameter. For a launch vehicle, the theoretically optimal expansion ratio changes with altitude as the external pressure changes, even though the physical nozzle's expansion ratio is fixed. In order to find the engine mass for the optimal case, a few assumptions can be made and changed before the mass results are calculated.

First, instead of requiring the expansion ratio as an input parameter, the outer stage diameter of the rocket is needed. It can be assumed that the rocket only has a single engine, and that the exit diameter of the engine will not exceed the diameter of the vehicle. As such, the cross-sectional area of the rocket can be assumed to be the value for  $A_e$ . This assumption replaces Equation (2.3).

Next, it will be assumed that the  $I_{sp}$  input is the optimal  $I_{sp}$  found when the rocket is optimally expanded. The exact definition of what is “optimal” depends on the circumstances, and the optimal  $I_{sp}$  would have to be arrived at through the use of some other analysis tool, such as Cequel.<sup>17</sup>

With thrust, chamber pressure, and chamber temperature known, Equation (2.1) can be evaluated to find mass flow rate. Equation (2.2) can then be used to find throat area, as is done in the original version of the analysis.  $A_e$  was already found from the stage diameter assumption, so throat area can be used to find the other engine dimensions by evaluating Equation (2.4) and the rest of the analysis.

This change in the analysis is very minor, essentially swapping out the expansion ratio input for stage diameter and using that to calculate exit area, but this alternate formulation can be used to more realistically represent the dimensions of an optimally expanded engine without having to know the optimal expansion ratio ahead of time.

## 2.2 Launch Vehicle Simulation

The original version of the system sizing mass and performance spreadsheet<sup>10</sup> only considered feed systems used for in-space propulsion. As part of the continuing work to enhance the spreadsheet, a computer program was written to implement the vertical rocket equations with and without drag and simulate a launch vehicle. This program was called DeltaV, and was implemented in Excel using Visual Basic for Applications. Though developed as a separate spreadsheet, the DeltaV code was compatible with the system sizing code and could be integrated with little difficulty. This allows for different propulsion systems to be compared directly for launch vehicles (as opposed to just in-space vehicles, as was the case with the earlier system sizing code). This was an important improvement in the system sizing code's capabilities.

The main purpose of DeltaV is to provide a calculation tool that can determine the final velocity and altitude of a rocket given initial conditions, and determine how much

payload the rocket is able to deliver to a target velocity and altitude. DeltaV was developed as a separate program from the analysis spreadsheet, but implemented in such a way that it could be integrated into the spreadsheet at a later date. The current version of DeltaV uses a simplified vertical trajectory model and the driving inputs behind the simulation are vehicle and propellant mass.

### 2.2.1 Equations and Algorithm

Simulation of the launch trajectory can be done by evaluating the velocity equation for an ideal rocket. In differential form,<sup>3,18</sup> this equation is

$$dv = \frac{T}{\dot{m}} \frac{dm}{m} - \frac{D}{m} dt - g \sin \theta dt . \quad (2.14)$$

The angle  $\theta$  in Equation (2.14) is measured from the horizontal reference plane to the direction of the rocket's flight.

The DeltaV program uses this equation to run an iterative simulation of the rocket trajectory. For each time step,  $dt$ , Equation (2.14) is evaluated to determine how the rocket's velocity changed for that time step. The change in position of the rocket can be found by multiplying  $dv$  by the time step. The goal of the program is to calculate the final height and final velocity of a rocket that starts out at a certain altitude and velocity. The user can specify that the rocket will turn in flight from some initial angle to a specified final angle. Typically, for vertical flight, the initial angle is ninety degrees (as measured from the horizontal plane). The time that the turn occurs can be set to the entire burn time, or some subset of it. Additionally, a three stage thrust profile may be

included in the simulation. The second and third thrust values enter into the calculations at the times specified for them to start. The program inputs are listed below in

Table 2.2. Clarification of what the inputs refer to is included where necessary.

Table 2.2 DeltaV Input Parameters

<b><u>Input</u></b>	<b><u>Symbol</u></b>
Empty Weight (rocket without propellant)	$w_e$
Propellant Weight	$w_p$
Inert Weight (non-propulsion structural weight)	$w_i$
Propulsion System Weight (engine, fuel tanks, pressure bottles, etc.)	$w_{misc}$
Specific Impulse	$I_{sp}$
Burn Time	$t_b$
Initial Velocity	$v_0$
Initial Altitude	$h_0$
Cross Sectional Area (largest cross section of the rocket)	$A$
Target Velocity (desired final payload velocity)	$v_{target}$
Initial Flight Angle (measured from horizontal axis)	$\theta_0$
Final Flight Angle	$\theta_{end}$
Time to Start Turn (when the simulation begins calculating the turn from the Initial Flight Angle to the Final Flight Angle)	$t_{startturn}$
Time to End Turn (when the simulation completes the turn)	$t_{endturn}$
Initial Thrust	$T_0$
Second Phase Thrust (used to include changes in thrust over the burn time)	$T_2$
Third Phase Thrust	$T_3$
Time to Start Second Phase	$t_{T2}$
Time to Start Third Phase	$t_{T3}$

The simulation is iterative and operates under a time step approach to calculate the final results. The time step is determined by dividing the burn time by one thousand.

The program loops through the calculation simulating the change in velocity and flight angle until such time as the calculated mass loss is greater than the mass of the propellant. If the other thrust phase inputs are used, the simulation will also change the thrust as appropriate. The algorithm used by the simulation is as follows:

- 1) Define variables, clear old data, and read in the input data.
- 2) Error check that the initial thrust is greater than the initial weight of the rocket.
- 3) Set up the first iteration by calculating:

$$m = w_e + w_p , \quad (2.15)$$

$$\dot{m} = \frac{T_1}{I_{sp}} , \quad (2.16)$$

$$v_x = v_0 \cdot \cos \theta , \quad (2.17)$$

$$v_y = v_0 \cdot \sin \theta , \quad (2.18)$$

$$v = \sqrt{v_x^2 + v_y^2} , \quad (2.19)$$

$$\rho = 0.075 \cdot \exp(-h_0^{1.15} \cdot 7.4 \times 10^{-6}) , \quad (2.20)$$

and

$$\frac{d\theta}{dt} = \frac{-(\theta_0 - \theta_{end})}{t_{endturn} - t_{startturn}} . \quad (2.21)$$

Note that atmospheric density and pressure are based on the ICAO standard atmosphere formula.<sup>18</sup>

4) Evaluate the following equations until the mass goes below the empty weight. At this condition the program is set to recognize that there is no propellant left. In these equations the subscript ' $i$ ' denotes the iteration counter. An ' $i - 1$ ' subscript denotes the value of the variable from the previous iteration.

5) First calculate the pressure at the current altitude, the Mach number ( $M$ ), and the drag force. The equations for these quantities are

$$P = 2395.3 \exp(-h_i \cdot 5 \times 10^{-5}), \quad (2.22)$$

$$M = v_i \sqrt{\frac{\rho_i}{1.4 P_i}}, \quad (2.23)$$

and

$$D_i = C_D \rho_i v_i^2. \quad (2.24)$$

The value of the drag coefficient,  $C_D$ , is selected based on a typical<sup>18</sup> variation versus Mach number for an inclination to the flight direction of  $0^\circ$ . The program does not replicate the entire variation point for point, but breaks  $C_D$  values into the following ranges:

$$0.065 \text{ for } M < 0.5, \quad (2.25)$$

$$0.0825 \text{ for } 0.5 < M < 1,$$

0.125 for  $1 < M < 1.4$ ,

0.14 for  $1.4 < M < 2$ ,

(2.25)

0.12 for  $2 < M < 3$ ,

and

0.1 for  $M > 3$ .

(2.25)

6) With drag determined, the change in velocity for an ideal rocket may be calculated using the equation<sup>3,18</sup>

$$\frac{dv}{dt} = \frac{T}{M_i} - \frac{D_i A}{M_i} - g_e \sin \theta_i . \quad (2.26)$$

7) The change in velocity allows the program to advance the variables to the next iteration. The calculations are

$$v_{x,i+1} = v_{x,i} + \frac{dv}{dt} dt \cos \theta_i , \quad (2.27)$$

$$v_{y,i+1} = v_{y,i} + \frac{dv}{dt} dt \sin \theta_i , \quad (2.28)$$

$$v_{i+1} = \sqrt{v_{x,i+1}^2 + v_{y,i+1}^2} , \quad (2.29)$$

$$\theta_{i+1} = \theta_i + \frac{d\theta}{dt} dt \quad (2.30)$$



(If the current time ( $t$ ) value is between the Start Turn and End Turn input values),

$$h_{i+1} = h_i + (v_{y,i+1} dt), \quad (2.31)$$

$$\rho_{i+1} = 0.075 \cdot \exp(-h_{i+1}^{1.15} \cdot 7.4 \times 10^{-6}), \quad (2.32)$$

$$m_{i+1} = ms_i - \dot{m} dt, \quad (2.33)$$

and

$$t_{i+1} = t_i + dt. \quad (2.34)$$

8) Check to see if the simulation time is past the start time for either Phase Two or Phase Three thrust, and if so change the value of  $T$  accordingly.

9) With the iterative simulation complete, output the final results. If the final velocity is greater than or equal to the target velocity, then the payload mass is given by

$$m_{payload} = w_e - (w_i + w_{misc}). \quad (2.35)$$

Otherwise, the payload mass is zero. It should be noted that the output of zero payload mass does not mean there is no payload; instead, it is intended to show that the specified rocket conditions cannot get any payload to the target velocity. The rocket system may be perfectly suitable to deliver some amount of payload to some other target velocity, just not the one the user specified.

10) Output final velocity, altitude, mass, and flight angle.

11) Use Empty Weight, Propellant Weight, and final mass to calculate and output the percentage of propellant remaining at the end of the simulation.

The assumptions made in this program are as follows. There is constant mass flow rate through the engine and constant thrust for each of the three thrust phases.

The turn from the initial flight angle to the final flight angle is done uniformly through the time specified. Gravitational acceleration throughout the flight is  $32.2\text{ft/s}^2$ , sea level gravitational acceleration.

### 2.2.2 Verification and Validation

The DeltaV code was checked for accuracy by running the simulations using data from the Apollo program Saturn V rocket, specifically the first S-IC stage.<sup>19,20,21</sup> The program inputs and outputs used for code verification are shown in Table 2.3.

While this simulation ends with the rocket burning all of its propellant (which does not always occur in a real rocket) the final velocity and altitude results are in good agreement with the actual values of 7,838ft/s and 202,099ft, respectively.

An early version of the program did not include the rocket turning or thrust profile features that the final version had. The results of this version had very good agreement with velocity, but the altitude results were very high. It was realized that the straight vertical trajectory was not what physically happened to a rocket in flight, so the extra trajectory modifying features were added. By specifying how the rocket turns or how the thrust behaves with time, the simulation can be adjusted to bring it into closer agreement with the physical reality. With the kind of equations used in this simulation, based on ideal rockets, any result within 15% of the real value is going to be good enough for a

first-order estimate of what will actually happen, since the main purpose of this trajectory analysis was to provide an internally consistent means of comparing the different propulsion systems.

Table 2.3 DeltaV Verification and Validation Inputs/Outputs

<b>Input</b>	
Empty Weight (lbm)	1722000
Propellant Weight (lbm)	4393000
Inert Weight (lbm)	23400
Propulsion System Weight (lbm)	210600
Specific Impulse (s)	265
Burn Time (s)	150
Initial Velocity (ft/s)	0
Initial Altitude (ft)	0
Cross Section Area (ft <sup>2</sup> )	845.4
Target Velocity (ft/s)	7800
Initial Flight Angle (degrees)	90
Final Flight Angle (degrees)	20
Time to Start Turn (s)	0
Time to End Turn (s)	70
Initial Thrust (lbf)	7500000
Second Phase Thrust (lbf)	7800000
Third Phase Thrust (lbf)	8100000
Time to Start Second Phase (s)	50
Time to Start Third Phase (s)	100
<b>Output</b>	
Payload Weight to target velocity (lbm)	1488000
Final Velocity (ft/s)	7953
Final Altitude (ft)	191835
Final Mass (lbm)	1723000
Final Flight Angle (degrees)	20.1
Propellant Remaining (%)	0.023

The data in Table 2.4 summarizes the results of the code and how they compare with the actual results, using the input values above. The version that uses variable  $C_D$  is the current version.

Table 2.4 Comparison of DeltaV Results

	Actual Values	Constant $C_D=0.2$	Variable $C_D$ Value	Constant $C_D$ % Error	Variable $C_D$ % Error
Velocity	7837.6 ft/s	7912.1 ft/s	7953.5 ft/s	0.96%	1.46%
Altitude	202099.7 ft	190643.4 ft	191835.2 ft	5.67%	5.08%

In summary, the DeltaV code has been successfully tested and validated using the Saturn V inputs. It is reasonable to expect that this program will allow for first-order estimates of rocket trajectory and payload calculations for the different propulsion system options.

### 2.2.3 Alternate Version: Payload Analysis

An addition to the DeltaV project was made by modifying the input parameters required by the calculation. The version of DeltaV so far described is the fifth revision (previous revisions were code tests or incomplete versions). The revision 6 code is very similar to the revision 5 code, but has a slightly different intent. The key feature of revision 6 is that payload weight is determined by the program in such a way that the end

result is the maximum theoretical payload for the given rocket system. The problem with the way revision 5 is set up is that by requiring empty and structural weights as inputs, the payload weight is constrained to be the difference of those two inputs.

In revision 6, the empty weight is not included as an input and the inert and propulsion system weight inputs are combined into one structural weight input. Payload weight, and by extension empty weight, is guessed at by the program before the simulation is run. The algorithm for this is as follows:

1. Set first guess for payload as

$$PayloadGuess = (T_0 - 1) - w_p - w_i \quad (2.36)$$

2. Begin a loop, on each iteration evaluate the equation<sup>18</sup>

$$\begin{aligned} \Delta v &= v_e \ln(R) \\ \Delta v &= I_{sp} g_e \ln \left( \frac{w_p + w_e + PayloadGuess}{w_e + PayloadGuess} \right). \end{aligned} \quad (2.37)$$

3. Compare this  $\Delta v$  to the difference in target and initial velocity, if the equation result is less than that difference; decrease the payload guess by 0.1 times the current guess.
4. If  $\Delta v$  is greater than or equal to the input change in velocity, run the simulation (as outlined for revision 5, using the payload and structural weight to calculate empty weight).
5. If the simulation does not yield a final simulated velocity greater than the target, reduce the payload guess and reiterate.
6. Continue this loop until the program arrives at a payload guess that gives the desired target velocity when the simulation is run.

Using this code the user can evaluate a theoretical rocket system on the basis of what is the upper limit of the payload that can be delivered to the target velocity.

Verification was done using the same Saturn V data as for the revision 5 code and a summary of the inputs and outputs is shown below in Table 2.5.

Table 2.5 DeltaV Revision 6 Code Inputs and Outputs

<b>Input</b>	
Propellant Weight (lbm)	4393000
Structural Weight (lbm)	234000
Specific Impulse (s)	265
Burn Time (s)	150
Initial Velocity (ft/s)	0
Initial Altitude (ft)	0
Cross Section Area (ft <sup>2</sup> )	845.4
Target Velocity (ft/s)	7800
Initial Flight Angle (degrees)	90
Final Flight Angle (degrees)	20
Time to Start Turn (s)	0
Time to End Turn (s)	70
Initial Thrust (lbf)	7500000
Second Phase Thrust (lbf)	7800000
Third Phase Thrust (lbf)	8100000
Time to Start Second Phase (s)	50
Time to Start Third Phase (s)	100
<b>Output</b>	
Payload Weight to target velocity (lbm)	1884974
Final Velocity (ft/s)	7962
Final Altitude (ft)	556937
Final Mass (lbm)	2121163
Final Flight Angle (degrees)	20
Propellant Remaining (%)	0.049

Note that while the velocity prediction is fairly consistent (as it should be, the program requires that the target velocity be reached) the altitude is much higher in this

simulation. These data show the difficulty of validating the revision 6 code. It is unknown if the Saturn V was carrying the maximum amount of payload, and there are several other assumptions (thrust phases, timing, flight trajectory, constant  $I_{sp}$ ) that can alter the outcome. Though formal validation is difficult in this case, the purpose of the program is not to replicate historical data. Rather, the goal is to provide a first order estimate on what an upper limit on payload is, given that other mission parameters are constant. For that purpose, these results show that the program is working as intended.

### 2.3 SFS Reliability Analysis

One of the key features of the three-tank SFS concept is the fail-operational mode. In principle, the system may suffer a valve failure in one tank and continue to operate on the remaining tanks. The questions of interest then, are how is SFS reliability determined for a real system and how does that reliability compare with that of other feed systems? The purpose of this analysis was to develop a reliability model that could answer those questions.

Following previous work<sup>10</sup> done on this topic, the reliability model was developed according to the method used by the military standard<sup>22</sup> for mission reliability. A key part of this method is the development of block diagrams. These diagrams are used to show how the functions of the various equipments that make up an item are related to each other in terms of overall success of the item. A functional description of, and block diagrams for, conventional pressure fed systems, turbopumps, and the SFS are given in the next section.

### 2.3.1 Model

Three types of feed systems were considered for this analysis: conventional pressure fed (CPF), turbopump (TP), and SFS. While many variants of the CPF and turbopump systems exist, it was decided to develop a very general and simplified model of each system for analysis and comparison purposes.

A conventional pressure fed system is characterized by the use of pressurized gas in the ullage space of the propellant storage tank to supply the needed pressure to force propellants into the engine. Functionally, this system can be broken down into five components: the pressurant tank, the propellant tank, the engine, and the two valves that control the pressurant flow into the propellant tank and the propellant flow into the engine, respectively. The standard method<sup>22</sup> for analyzing the reliability of this system would be to draw a block diagram with five components in series, as shown below in Figure 2.4.

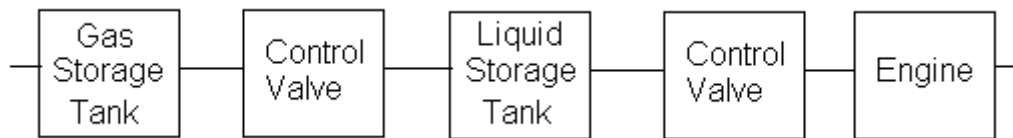


Figure 2.4 Block Diagram for Conventional Pressure Fed System

Block diagrams are read left to right, and indicate success if there is always a path going right, even with the failure of one piece of the equipment blocks. Equipment refers to any block on the diagram, and may represent a single component in the real system or



a more complicated subsystem. Each equipment block has its own reliability value associated with it. Equipment blocks that are drawn in series with each other (one after the other, connected with a line) indicate that both of those equipments must function in order for the path to be valid. Equipment blocks drawn in parallel (connected by a line with one or more branches) represent alternate modes of operation.

In the sample diagram for the CPF system, all five components are drawn in series. Examination of the block diagram shows that the storage tanks and the control valves and the engine must all work in order for the entire feed system to be successful. It should be noted that this is a very general and simplified model, and if it were to be used, then additional diagrams would have to be drawn if the actual components had any complexity to them. For example, the engine could contain a coolant system or thrust vector control mechanism. Thus, the block on the diagram in Figure 2.4 can be treated as representing a subsystem which could have its own reliability block diagram. For simplicity in this analysis, the block diagram for such a subsystem is omitted.

In order to evaluate the block diagrams and determine the reliability of the feed systems they represent, certain simplifying assumptions are made. The key assumption, for the purposes of this analysis, is that the only sources of failure are from the valves. All tanks and other hardware in the feed systems are assumed to be 100% reliable. The reasons for this assumption are to simplify the analysis, as well as allow a comparison between feed systems based on similar components. There are components that are not common to all feed systems, such as a gas generator in a turbopump, and so the goal is to only evaluate the reliability in terms of hardware that is common for all feed system

types. Performing the analysis with this assumption allows for the comparison of systematic reliability effects that are due only to the valves in each type of feed system.

Applying this assumption requires a revision in the block diagram for the CPF. The storage tank and the engine may be removed from the diagram since they are assumed to always work. This leaves only the control valves, which for many CPF systems are quad-valves. The block diagram for a CPF system modeled as two quad-valves is shown in Figure 2.5.

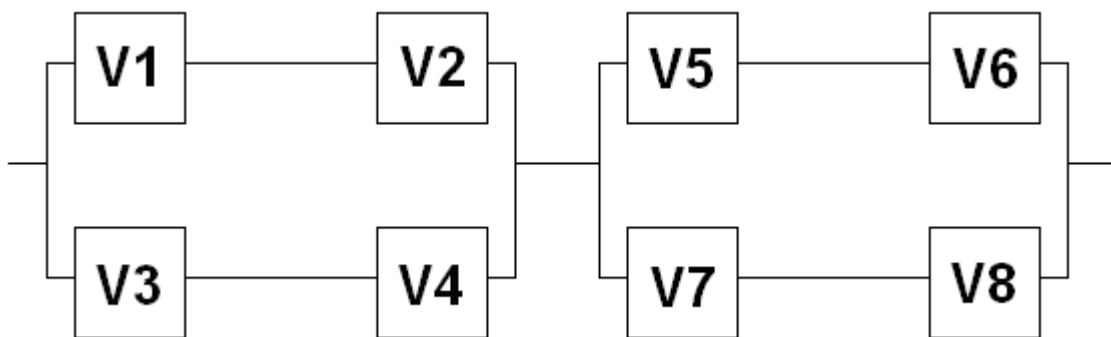


Figure 2.5 Conventional Pressure Fed Block Diagram

The quad-valve is made up of four separate valves, arranged as shown in the block diagram (one quad-valve on the left, and one on the right). All four valves are given the same open and close commands, and in the normal case all four valves actuate together. When open, fluid flows through both branches to the end, and when closed the leftmost pair of valves stop the fluid flow. The purpose of this arrangement is to allow for normal valve operation even if one of the component valves suffers a failure. This

can happen in one of two ways: the valve fails open, or it fails closed. In the fail open case, occurring for example in V1, the quad-valve will act normally when it is supposed to be open. When it is supposed to be closed, fluid will flow past the failed open V1, but will be stopped by V2. If the valve fails closed, then the fluid will always flow through V3 and V4 when the quad-valve is supposed to be open. When the quad-valve is supposed to be closed, it will behave normally. The same type of behavior occurs if the failure occurs in any of the four component valves.

Modeling the CPF control valves as a quad-valve highlights another important assumption in this analysis: similarity. It is assumed that all similar valves and subsystems have the same reliability. This means that for the CPF,  $V1=V2=\dots=V8$ , which is to say that V1 is as likely to succeed as any of the other valves.

The logic of the block diagram differs slightly from the functionality of the quad-valve. In the block diagram, the only concern is whether or not the equipment is functioning as intended. No consideration is given as to whether the quad-valve is supposed to be open or closed, or what exact kind of failure occurs in a specific valve. The equipment choices are binary: success or failure. Viewed in these terms, the block diagram can be read by going back to how a successful operation path is defined. In the CPF case, the way to read the block diagram for the first quad-valve is: V1 *and* V2 must function *or* V3 *and* V4 must function. Note that ‘and’ and ‘or’ in the last statement should be read as logical operators. If V1 fails, then it has a false value. V2 may be functioning correctly, but that is irrelevant to the success of the V1-V2 path, since V1 *and* V2 is equal to false *and* true, which is false. For the whole system, the first quad-valve (V1 through V4) *and* the second quad-valve (V5-V8) must function.

Regardless of how exactly a valve fails, the reliability of the entire system is determined by whether or not there is another path from the left side of the diagram to the right. In the case of the quad-valve for the CPF, such a path exists if the only failures occur in one branch of the quad-valve (V1-V2, V3-V4, V5-V6, or V7-V8).

A turbopump feed system is characterized by the use of a gas turbine driven pump to pressurize the propellants prior to injection into the engine. There are many variations on the turbopump concept, for this analysis a simplified version<sup>3</sup> was selected. This example turbopump is shown below in Figure 2.6, and is representative of the type of turbopump system used with relatively low thrust engines (e.g., RL10) and many larger engines. More complex turbopumps and engines, such as the SSME, are not modeled here.

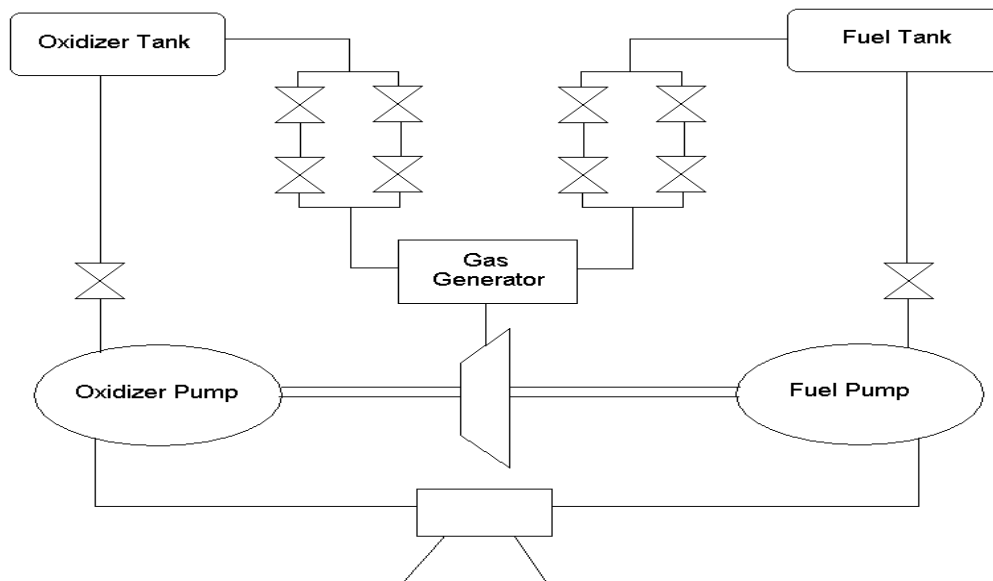


Figure 2.6 Schematic Diagram of Example Turbopump System

In this system, oxidizer and propellant are bled off from the main propellant storage tanks and combusted in the gas generator. The products of combustion are used to drive the turbine, which in turn drives the pumps that pressurize the remainder of the propellant. A quad-valve is used to control the flow of propellant into the gas generator, and a single valve is used to control the flow of propellant into the pump.

Applying the assumptions that only the valves can fail allows a great deal of simplification when drawing the turbopump block diagram. The tanks, pumps, engine, gas generator, and turbine can all be considered 100% reliable. This leaves only the quad-valve and the inlet valve. To be consistent with the CPF model, the block diagram will only consider one side, fuel or oxidizer, of the feed system.

Even though the quad-valve and the inlet valve are not in series physically, it is important to realize that both must function in order for the system to succeed. As such, the two valves are in series functionally. The block diagram to represent the turbopump is shown in Figure 2.7. It should be noted that valves in the turbopump pressurization system, needed to keep the liquid propellant at the right pressure throughout the pump assembly, are not included here. This makes the turbopump model a more conservative estimate of reliability, in comparison to the SFS which must include pressurization valves.

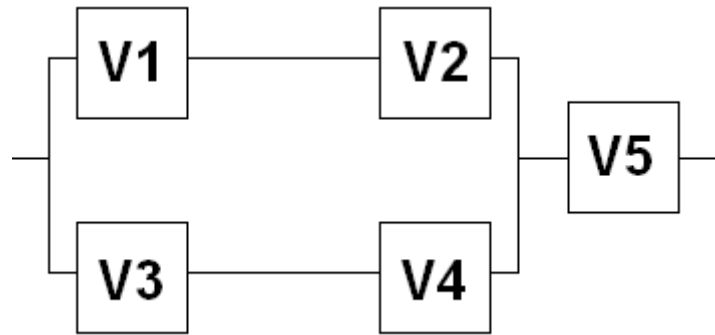


Figure 2.7 Turbopump Block Diagram

In this diagram, V1 through V4 represent the valves that make up the quad-valve, and V5 represents the inlet valve. Though the numbering is the same, for convenience, it is not intended that V1 of the turbopump is the same type of valve with the same reliability as V1 of the CPF. What is the same is the similarity assumption, so that we can assume that the four valves of the quad-valve all have the same reliability. The inlet valve (V5) cannot be considered part of the same subsystem; it has a different reliability associated with it.

The Sequential Feed System (SFS) is characterized by the use of several high pressure “run” tanks that are sequentially filled, pressurized and drained into the engine. Each tank has four valves that control venting of the pressurizing gas, refill of propellant from the main storage tank into the run tank, flow of pressurization gas into the run tank, and propellant flow to the engine. The standard SFS arrangement consists of three run tanks, in order to allow for a fail-operational mode. If any of the valves on any one of the tanks fails during operation, the other two tanks can continue to maintain flow to the engine. One tank will be refilling while the other is draining, while the third failed tank

is isolated from the others. Details of the various valve failure modes was investigated experimentally, and discussed in Chapters 4 and 5. From this model the following block diagrams can be drawn, shown in Figure 2.8 and Figure 2.9.

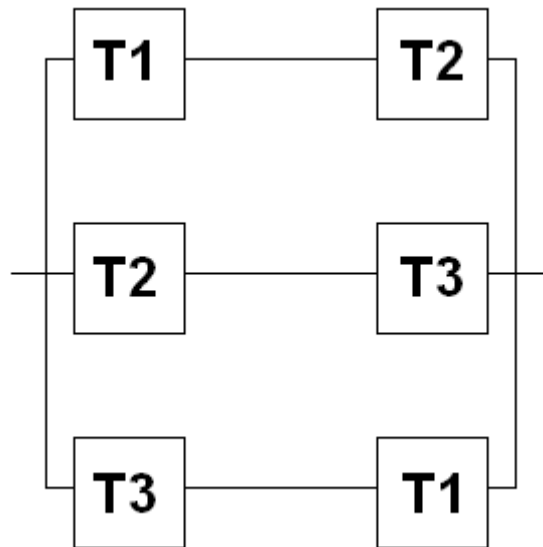


Figure 2.8 SFS Block Diagram



Figure 2.9 SFS Tank Subsystem Block Diagram

The SFS diagrams demonstrate how subsystems and a two-out-of-three condition can be represented on the block diagram. Each SFS tank subsystem (labeled “T” in

Figure 2.8) consists of the four valves associated with that tank (Figure 2.9). In this model all four valves are in series and must all function in order for the tank subsystem to function. To simplify the larger SFS diagram, it is not necessary to draw all four valves for each tank, as long as the subsystem is defined. T1 contains four valves, as does T2 and T3. The structure of the SFS diagram is intended to show the fail-operational capability of the SFS. Success occurs if either T1 *and* T2 work, *or* T2 *and* T3 work, *or* T3 *and* T1 work. In terms of actual operation, one expects that the third tank (whatever number that may be at the time) will also work after the first two, but in terms of reliability the only concern is what the possible modes of operation are. The case in which all tanks work is covered equally by all three branches of the diagram.

### 2.3.2 Derivation of Equations

With the models for the three feed systems designed, the next goal was to develop equations that would determine a numerical value for the reliability of a feed system. These equations were developed using method 1001 from the military standard.<sup>22</sup> The method is known as the conventional probability method, and works by evaluating the block diagram in terms of the probability of success of each equipment block. The block diagrams are related to equipment probabilities through two basic laws: for equipment blocks *A* and *B* in series

$$P(A \cap B) = P(A) \cdot P(B), \quad (2.38)$$

and for equipment blocks *A* and *B* in parallel



$$P(A \cup B) = P(A) + P(B) - P(A \cap B) . \quad (2.39)$$

In this example  $A$  and  $B$  are generalized events, and the equations may be applied recursively for more complex situations. The symbols and abbreviations used in this derivation are summarized in Table 2.6.

Table 2.6 Reliability Analysis Nomenclature

<b><u>Symbol</u></b>	<b><u>Meaning</u></b>
$V1$	Any of the valves in the CPF or Turbopump (TP) quad-valves, or any SFS valve.
$V5$	A non-quad-valve component for a Turbopump.
$T1$	SFS tank subsystem, consisting of four V1 components in series.
$P_F$	Probability of Failure.
$P_S$	Probability of Success.
$MTBF$	Mean Time Between Failure.
$\sigma$	Standard Deviation (in reference to the MTBF).
$P_X$	Probability of Success for X, e.g., $P_{V1}$ .
FS	Feed System. Either CPF, TP, or SFS.
$N$	Total Number of Feed Systems.
$M$	Number of Feed Systems needed for overall success.
$Z$	Number of combinations of successful Feed Systems needed for overall success.

With the nomenclature defined, the derivation of the reliability equations is a simple matter of applying the probability equations to the block diagrams. For the CPF system the probability of success for a single quad valve is given as

$$P_{CPF,QV} = P_A + P_B - P_A P_B , \quad (2.40)$$

where  $A$  and  $B$  represent the two branches of the valve. Branch  $A$  is the side containing  $V1$  and  $V2$ , expressed as

$$P_A = P_{V1}P_{V2} = (P_{V1})^2, \quad (2.41)$$

and from the similarity assumption  $P_B$  can be evaluated as

$$P_B = P_{V3}P_{V4} = (P_{V1})^2. \quad (2.42)$$

The result is the expression for the probability of success of one of the CPF quad-valves:

$$P_{CPF,QV} = (P_{V1})^2 + (P_{V1})^2 - (P_{V1})^2 \cdot (P_{V1})^2. \quad (2.43)$$

The total probability of success for the CPF is the equation for two quad-valves in series,

$$P_{CPF} = [2(P_{V1})^2 - (P_{V1})^4]^2 = 4(P_{V1})^4 - 4(P_{V1})^6 + (P_{V1})^8. \quad (2.44)$$

The turbopump equation is just a modification of the CPF equation, adding a  $V5$  term in series with the equation for a quad-valve:

$$P_{TP} = [2(P_{V1})^2 - (P_{V1})^4] \cdot P_{V5}. \quad (2.45)$$

And lastly, the SFS equation is derived by evaluating a three-element parallel equation. This equation may be obtained by applying the two-element parallel equation ( $A$  or  $B$ ) recursively. This yields the equation

$$P_{SFS} = P_A + P_B + P_C - P_AP_B - P_AP_C - P_BP_C + P_AP_BP_C, \quad (2.46)$$

where

$$P_A = P_{T1}P_{T2} = (P_{T1})^2 = (P_{V1})^8 \quad (2.47)$$

and

$$P_A = P_B = P_C, \quad (2.48)$$

which results in the final equation

$$P_{SFS} = 3(P_{V1})^8 - 3(P_{V1})^{16} + (P_{V1})^{24}. \quad (2.49)$$

With these equations, the probability of success for any of the three feed system types can be calculated as long as the probability of success for the component valves is known. In keeping with the probability model, the method chosen for modeling probability of success was a Gaussian distribution, centered on the mean time between failure (MTBF).

To evaluate the reliability model in a useful manner, it was necessary to develop relations to obtain reliability as a function of time for each feed system. To do that, the probability that a failure had occurred needed to be calculated for each time of interest. From probability theory,<sup>23</sup> the equation for the Gaussian distribution is given as

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(t - MTBF)^2}{2\sigma^2}\right], \quad (2.50)$$

such that  $f(t)dt$  is defined as the probability that a single measurement of  $t$  is between  $t$  and  $t + dt$ .

When applied to the reliability model, this means that all valves will eventually fail, and that the number of failures as a function of time is distributed normally about the

MTBF. To evaluate the reliability equations, we would like to know the probability that a valve has failed at a given time  $t$ . The expression for the probability of finding a failure in some range  $\Delta t$  about the MTBF is a modification of Equation (2.50), given as

$$\text{Pr ob}(\Delta t) = \int_{MTBF-\Delta t}^{MTBF+\Delta t} \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{-(t-MTBF)^2}{2\sigma^2}\right]. \quad (2.51)$$

The probability of success or probability of failure-free operation for a specified length of time is defined<sup>24</sup> as the integral of the probability distribution, Equation (2.50), from the start time to the operating time. Some modifications to Equation (2.51) can be made so that the probability is evaluated about that interval, not some  $\Delta t$  about the MTBF. If the substitution

$$\tau = \frac{t-MTBF}{\sigma} \quad (2.52)$$

is made, then Equation (2.51) can be rewritten as

$$\text{Pr ob}(\tau_1) = \frac{1}{\sqrt{2\pi}} \int_{-\tau_1}^{\tau_1} e^{\frac{-\tau^2}{2}}. \quad (2.53)$$

Though this integral cannot be evaluated analytically, results are tabulated<sup>23</sup> for various values of  $\tau$ . A valve's probability of failure can then be evaluated using the following equations. First defining the start point of the range of interest at  $t = 0$ ,

$$\tau_1 = \frac{0-MTBF}{\sigma} = \frac{-MTBF}{\sigma}, \quad (2.54)$$

and the end point of the range of interest at  $t$ ,

$$\tau_2 = \frac{t-MTBF}{\sigma}. \quad (2.55)$$

The range of interest for a given value of  $t$  is often not centered on the MTBF, so that Equation (2.53) cannot be evaluated directly. Instead, the tabular values<sup>23</sup> for probability as a function of  $\tau$  can be added together as needed to get the probability that a failure occurred between  $\tau_1$  and  $\tau_2$ . For the case where  $t < MTBF$  ( $\tau_2 < 0$ ), the probability of failure is

$$P_F = \frac{1}{2} \text{Pr ob}(\tau_1) - \frac{1}{2} \text{Pr ob}(\tau_2). \quad (2.56)$$

The  $\text{Pr ob}(\tau_1)$  and  $\text{Pr ob}(\tau_2)$  terms are evaluated using the tabulated values for Equation (2.53). Likewise, for the case where  $t = MTBF$  ( $\tau_2 = 0$ ),

$$P_F = 0.5. \quad (2.57)$$

And lastly, for the case where  $t > MTBF$  ( $\tau_2 > 0$ ),

$$P_F = 0.5 + \frac{1}{2} \text{Pr ob}(\tau_2). \quad (2.58)$$

With the probabilities of failure calculated for a value of  $t$ , the probability of success is found simply by subtracting the probability of failure from one.

### 2.3.3 Application

The reliability equations derived for each type of feed system are useful for comparison purposes on the conceptual level. Additionally, these equations can be applied to allow for comparison of the reliability of complete rocket systems. For the purposes of this analysis, only rocket systems that use the same type of feed system are considered.

The simplest case is that of a single-engine rocket. In keeping with the assumptions of this analysis, the engine itself will be considered 100% reliable, and the same goes for the rest of the rocket hardware other than the valves. The reliability equation for the rocket then becomes the reliability equation for the feed system(s). For a monopropellant rocket only one feed system is required, while for a bipropellant rocket two feed systems must work in series in order for the engine to function (one feed system for the fuel and another for the oxidizer). As a result, the reliability equations are

$$P_{S,Monopropellant} = P_{FS} \quad (2.59)$$

and

$$P_{S,Bipropellant} = (P_{FS})(P_{FS}) = (P_{FS})^2. \quad (2.60)$$

For the case of  $N$  engines there are two ways to do the analysis: the case where no engine failures are allowed and the case where there are  $M$  engines ( $M < N$ ) that must function in order for the rocket to be successful. For the latter case, this typically involves an “engine out” capability. In the no-failure case, the reliability model is simply  $N$  engines in series. For a monopropellant rocket, that means that  $N$  feed systems are required, and for a bipropellant rocket  $2N$  feed systems are required. The equations are then

$$P_{S,Monopropellant} = (P_{FS1}) \cdot (P_{FS2}) \cdot \dots \cdot (P_{FSN}) = (P_{FS})^N \quad (2.61)$$

and

$$P_{S,Bipropellant} = [(P_{FS1,oxidizer}) \cdot (P_{FS1,fuel})] \cdot \dots \cdot [(P_{FSN,oxidizer}) \cdot (P_{FSN,fuel})] = (P_{FS})^{2N}. \quad (2.62)$$

The allowed failure case is more complex when generalized to an arbitrary number of allowed engines. To simplify matters, this analysis assumes that since the engines themselves are considered 100% reliable only, the number of feed systems is what matters. This model can then describe a variety of rockets systems, such as one with  $N$  engines and a feed system for each, but only  $M$  engines are required for success, as well as a rocket with  $N$  engines with common feed systems that allow for one feed system to take over the workload of another should it fail. Taking this approach the reliability block diagram for the rocket system is as shown in Figure 2.10.

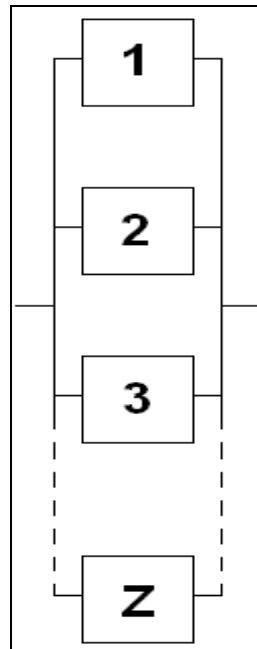


Figure 2.10 Block Diagram for N-Feed Systems with Allowed Failure Modes

The elements 1 through  $Z$  on the block diagram represent possible combinations of successful feed systems. A way to understand this is to look at the SFS block diagram as a derivative of this general diagram. In the SFS case there are three possible equipments ( $N = 3$ ), and only two of them must work for success ( $M = 2$ ), so there are three possible combinations that result in success ( $Z = 3$ ). Evaluating  $Z$  for arbitrary values of  $M$  and  $N$  is done by applying the combination equation from probability,<sup>25</sup> choosing  $M$  unordered outcomes from  $N$  possibilities:

$$Z = {}_N C_M = \frac{N!}{M!(N-M)!}. \quad (2.63)$$

Evaluation of the general block diagram can be done analytically, yielding an equation similar to the single engine case, but doing so is cumbersome for  $Z > 3$ . The general equation from probability theory<sup>25</sup> for any  $Z$  is given as

$$\begin{aligned} P(A_1 \cup A_2 \cup \dots \cup A_n) = & \sum_i P(A_i) - \sum_{i < j} P(A_i A_j) + \sum_{i < j < k} P(A_i A_j A_k) \dots \\ & + (-1)^{n-1} P(A_1 A_2 \dots A_n). \end{aligned} \quad (2.64)$$

For those cases with many successful feed system combinations, it was determined that the best approach was to evaluate the equation numerically using a recursive algorithm. This technique is discussed in the next section. In either evaluation method, it is important to make the correct substitution depending on whether a monopropellant or bipropellant rocket system is being analyzed. For a monopropellant system, each element of the block diagram in Figure 2.10 is  $M$  feed systems in series. Bipropellant rockets, in keeping with the single-engine analysis, require two feed systems in series. As such, each element in the diagram is then  $2M$  feed systems. The probabilities of success for each block on the diagram in Figure 2.10 are given as



$$P_1 = P_2 = \dots = P_Z = (P_{FS,monopropellant})^M \quad (2.65)$$

and

$$P_1 = \dots = P_Z = (P_{FS,bipropellant})^{2M} . \quad (2.66)$$

### 2.3.4 Numerical Implementation

The numerical analysis of this project was done using an Excel spreadsheet named Reliability. The goal was to generate charts of reliability versus time for all three types of feed systems in the kinds of cases described in the last section. Reliability data were generated for all three types of feed system from time equal to zero to twice the MTBF. The spreadsheet input section only requires the MTBF and standard deviation data for the three feed systems, as shown below in Table 2.7.

Table 2.7 Input Section of Reliability spreadsheet

<b>Conventional Pressure Fed</b>	MTBF	Std Dev
Quad-valve Component	1000	100
<b>Turbopump</b>		
Quad-valve Component	1000	100
Secondary Valve	1000	100
<b>SFS</b>		
Tank Valve	1000	100
<b>Rocket System</b>		
Total Number of Feed Systems	5	
Required Number of Feed Systems	4	

It should be noted that the program inputs selected and shown in Table 2.7 are completely arbitrary. The code does not require a specific time unit, and the user can adjust the inputs to be in whatever time scale is convenient. Additionally, the Rocket System inputs were chosen to represent a 5-engine system with an engine-out capability. Even though the inputs used for this analysis are arbitrary, the important point is that the comparison is made using the same inputs for all three feed system types. Physically, this means that the feed systems that are compared in this analysis are ones whose valves are equally reliable.

Reliability was calculated in the spreadsheet cells using the previously developed equations. Once the reliability of each valve was determined from the Gaussian model, that value could then be used to calculate the reliability of the feed system as a whole. That, in turn, was applied to the single and  $N$  feed system cases with no failures.

Evaluation of the simple equations was done within the Excel cells themselves, as the equations were easy to code. Since no limitation was made on the value of  $N$  or  $M$ , the treatment of the  $N$  feed system with failure case had to be different, in order to allow for a large  $Z$  value. The failure case was evaluated using the Visual Basic for Applications (VBA) programming language built in to Excel. The program used the calculated valve reliabilities at each time point and calculated the reliability of the  $N$  feed system case using a recursive method (the code of which is included in Section A.2.3). The intent of this method was to only calculate the simplest probability equation, that of the  $Z = 2$  case. By using the similarity assumption, the reliability number used in the probability calculation would be the same throughout.

### 2.3.5 Previous Work

A similar reliability analysis was done as part of the previous analytical work<sup>10</sup> for the SFS. The current reliability analysis was developed using the same starting point in the military standard,<sup>22</sup> but the project was conducted separately from the previous work. The reason for this was that the current analysis project was a continuation of a past project, but a new method was applied in order to enhance the accuracy of the results.

Though the overall results and conclusion from the former and current reliability analyses are the same, certain changes were made for the current analysis. First, in the former analysis, a CPF system was modeled as one quad-valve. This is a valid CPF system if the quad-valve represents the control of propellant flow between the tank and the engine. A real system that would use such a block diagram would be a Blowdown CPF, since no extra pressurant is added to the tank. The current analysis model views the more likely CPF case of a system where two quad-valves are used: one to control pressurant flow into the tank, and one to control propellant flow into the engine.

The second change was in how the SFS tank subsystem was modeled. The former analysis had a fifth component in series with the four valves shown in Figure 2.9, the tank itself. For the current analysis it was decided to assume that the SFS tank was 100% reliable, in order to allow for a more valid comparison with the CPF and turbopump systems.

Lastly, the most significant change was in how probability of success was calculated. The former analysis used a linear equation based on the definition of MTBF<sup>26</sup> given as

$$MTBF = \frac{\text{operating\_time}}{\text{probability\_of\_failure}}. \quad (2.67)$$

The probability of success is one minus the probability of failure, so using

Equation (2.67) the expression can be written as

$$P_s = 1 - \frac{\text{operating\_time}}{MTBF}. \quad (2.68)$$

It should be noted that using this probability of success model the symbol “MTBF” does not actually the “Mean” time between failures. When operating time exceeds MTBF,  $P_s$  becomes negative, which has no physical meaning. In the old model, the MTBF can be thought of as the “Maximum” time between failures. The same symbol is used because it is an artifact of the earlier analysis, and MTBF was the symbol used in the literature<sup>26</sup> that the old analysis method was based on.

What was found when the two  $P_s$  models (linear for the former analysis, and Gaussian for the current) were compared is that the linear model predicted significant reliability differences between the various feed system types right from the start. The Gaussian model has all feed systems behaving equally reliably until a short time before operating time reaches MTBF.

The way to interpret these results is that the linear model represents systems with failures that can happen right away, and the Gaussian model represents systems that have most failures happen within a band of time around the MTBF. After reviewing the analysis, it was concluded that the Gaussian model is the more physically accurate one, but the implications of the linear model should not be ignored. During the current analysis, values for MTBF and standard deviation were chosen arbitrarily, but in real

rocket systems failures can occur right at the start. Even at system startup the components are not technically “new” as they have typically been run for a length of time during quality testing. To include for the possibility of failures at or near time equal to zero, the Gaussian model needs only to be modified by increasing the value of the standard deviation. For this analysis project there was no research effort to determine what the MTBF and standard deviation of a real valve would be. This task, and refining the current reliability analysis to include the effects of quality testing, continuous operation effects versus switching effects (as occurs in valves), and shelf life are beyond the scope of this project.

#### 2.3.6 Results

Using the inputs shown in Table 2.7 the Reliability spreadsheet was used to evaluate six reliability cases: single feed system,  $N$  engines with no allowed failure, and  $N$  engines with allowed failure (one failure in the test case), each for monopropellant and bipropellant systems. Charts of reliability versus time were generated for all six cases. An example of one of these charts is shown in Figure 2.11. The three feed systems all follow the same trend of the error function,<sup>27</sup> which is to be expected since the probability of success equations are derived from the integral of the normal distribution. The main point of interest is to find how each system compares against the others in terms of probability as a function of time.

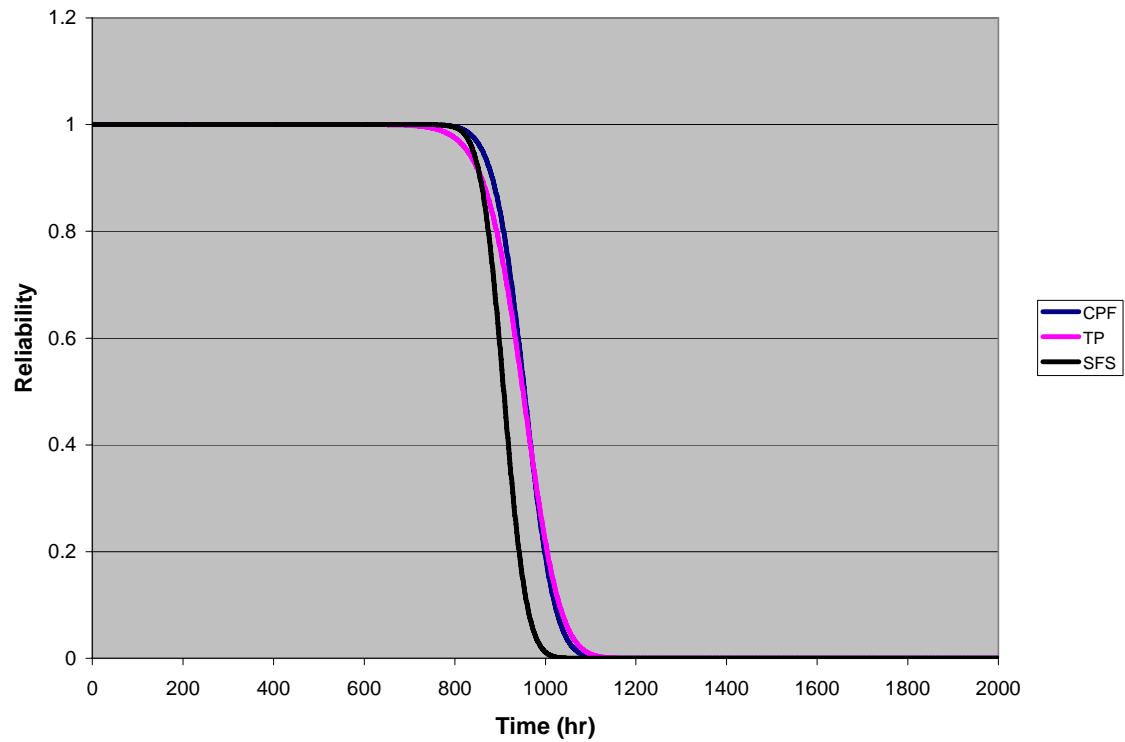


Figure 2.11 Reliability vs. Time Chart

In order to compare the reliability of the three feed system types, two metrics were chosen: the time needed to reach a certain refurbishment reliability, and the reliability at MTBF. The refurbishment reliability was defined as the reliability the system would be allowed to reach before maintenance was done to make it essentially good as new. For this analysis a refurbishment reliability of 0.999 was chosen. This reliability number is somewhat arbitrary, but a one in a thousand chance for a failure seemed like a reasonable standard to apply to refurbishment.

Figure 2.12 shows the same reliability versus time curve as in Figure 2.11, only zoomed in on the section when all three curves approach the refurbishment reliability.

On this scale it is easier to see how the feed systems' behavior diverges over time.

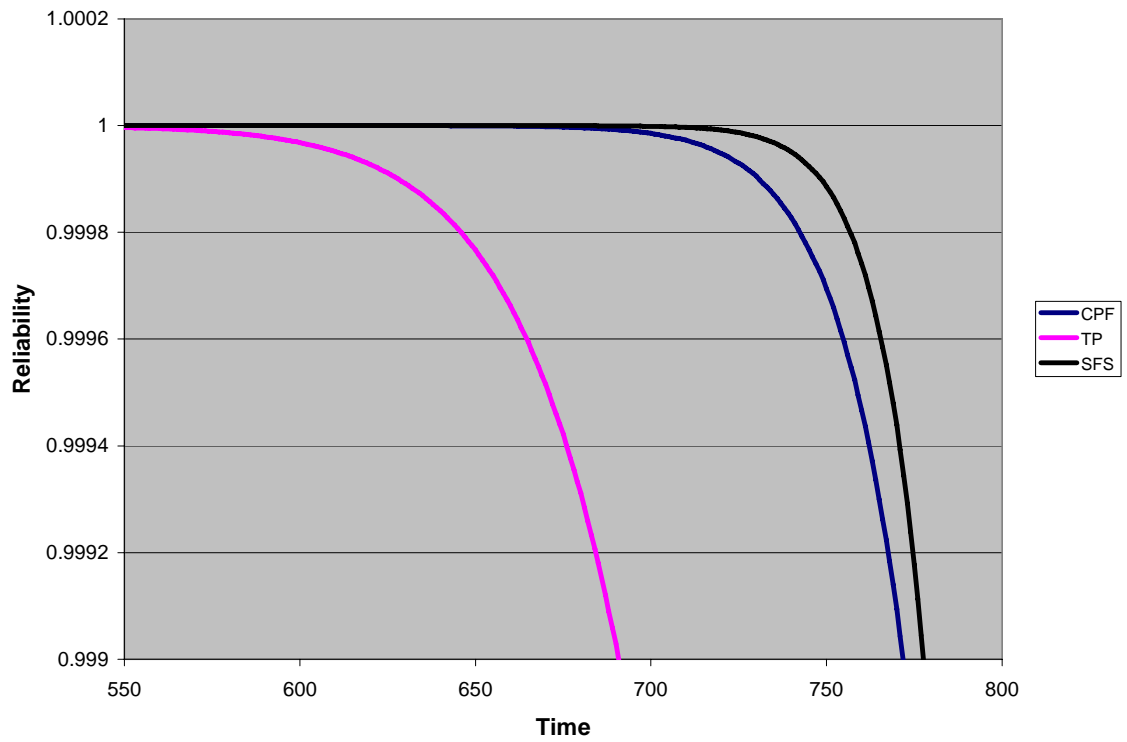


Figure 2.12 Reliability vs. Time Chart, Reduced Scale

The Reliability spreadsheet was coded to look through the data generated by the six reliability cases and evaluate the feed systems in terms of the time to reach the refurbishment reliability and their reliability at MTBF. The results are shown in Table 2.8.

Table 2.8 Reliability Spreadsheet Results

<b>Results Summary</b>						
Refurbishment Reliability	0.999					
<b>Time to Reach Refurbishment Reliability</b>						
	CPF	TP	SFS	1st Best	2nd Best	3rd Best
Single Monopropellant	772	691	778	SFS	CPF	TP
Single Bipropellant	759	671	769	SFS	CPF	TP
N-System Monopropellant	743	646	757	SFS	CPF	TP
N-System Bipropellant	731	628	749	SFS	CPF	TP
N-System Mono, Allowed Failure	872	843	848	CPF	SFS	TP
N-System Bi, Allowed Failure	852	815	835	CPF	SFS	TP
<b>Reliability at MTBF</b>						
	CPF	TP	SFS	1st Best	2nd Best	3rd Best
Single Monopropellant	0.1914	0.2187	0.0116	TP	CPF	SFS
Single Bipropellant	0.0366	0.0478	0.0001	TP	CPF	SFS
N-System Monopropellant	0.0002	0.0005	2.1673E-10	TP	CPF	SFS
N-System Bipropellant	6.6002E-08	2.5088E-07	4.6971E-20	TP	CPF	SFS
N-System Mono, Allowed Failure	0.0066	0.0113	9.2833E-08	TP	CPF	SFS
N-System Bi, Allowed Failure	9.0077E-06	2.6215E-05	1.7236E-15	TP	CPF	SFS

As can be seen from these results, the SFS has a much greater initial reliability than the other systems. It is best in terms of refurbishment reliability because it reaches that reliability level last among the three feed systems. An operational SFS can then be expected to remain operable with less frequent maintenance than another feed system. Even though this reliability benefit does not show up at MTBF, it is important to note that



at this operating time all of the feed systems' reliabilities have dropped significantly. It is highly unlikely that any real system would be allowed to operate that long without refurbishment. Normal engine run times are on the order of minutes. If the inputs used were 1000 seconds, then the time scale shown far exceeds that of typical engine operation. This model is intended to demonstrate that the reliability benefits of the SFS are most apparent for long duration missions. For times much less than MTBF, all feed systems are equally reliable. The difference in reliability is apparent at times closer to MTBF. At the refurbishment reliability, the SFS results show that it is the one feed system that can be run the longest before requiring maintenance.

## 2.4 SFS Vent Stage Analysis

The Vent Stage of the SFS cycle is defined as the period of time after an SFS tank has finished draining and before the tank begins to refill. During this time, the tank's vent valve is opened and excess pressurant gas is expelled from the tank. Determining the amount of time required for the Vent Stage was the primary goal of this analysis. In the course of achieving this goal, characterizations of the pressure, temperature, and pressurant mass as functions of time were also developed. The intent of this project was to use the resulting analytical functions as a means of predicting how long a vent stage should last, to assist with experimental calibrations.

In this analysis the following assumptions were made: 1) the SFS tank has a constant volume, 2) pressurant gas behaves as an ideal gas, with constant specific gas constant and ratio of specific heats, 3) when the vent is opened, the pressurant flow is isentropic, one-dimensional, and through a line of constant inner diameter, and 4) the gas

flow is choked during the vent process (the analysis is designed to end when the flow becomes unchoked).

The symbols used in the analysis equations are summarized below in Table 2.9. Two approaches were made in the course of analyzing this problem: an iterative solution and an analytical solution. The iterative solution is a computer model that calculates small changes in conditions over a period of time. The analytical solution develops equations to describe tank conditions as a function of time.

Table 2.9 Vent Stage Analysis Nomenclature

<b><u>Symbol</u></b>	<b><u>Definition</u></b>
$A$	Pipe Cross-sectional Area
$a$	Speed of Sound
$D$	Inner Pipe Diameter
$dP$	Pressure Increment
$dt$	Time Increment
$m$	Mass
$\dot{m}$	Mass Flow Rate
$n$	Iteration Counter
$P$	Pressure
$P_{ext}$	External Pressure
$P_f$	Final Pressure
$R$	Specific Gas Constant
$T$	Temperature
$t$	Time
$T^*$	Sonic Temperature
$V$	Volume
$\alpha$	Integration Constant
$\Delta$	Change in Property
$\gamma$	Ratio of Specific Heats
Subscript 1	Initial State (typically a Total Condition)
Subscript 2	Next State after Initial
Subscript i	Initial Conditions

### 2.4.1 Iterative Solution

The first approach to this problem was to develop an iterative computer simulation for the flow of gas out of the tank. Early attempts involved adapting some of the formulae from Crane,<sup>28</sup> but it was decided to proceed with a more idealized approach and assume smooth surfaces. Using the isentropic relations for choked flow, the following algorithm was developed.

- 1) Begin with the input parameters:  $V$ ,  $D$ ,  $\gamma$ ,  $R$ ,  $P_i$ ,  $T_i$ ,  $P_{ext}$ ,  $P_f$ , and  $dt$ .
- 2) Calculate the area of the Vent line,

$$A = \pi \left( \frac{D}{2} \right)^2. \quad (2.69)$$

- 3) Calculate initial mass of the pressurant gas inside the tank,

$$m_i = \frac{P_i V}{RT_i}. \quad (2.70)$$

(Note that  $R$  in these equations is for the gas being analyzed, in this case helium, not the universal gas constant.)

- 4) Calculate the pressure ratio at which the simulation should stop, given as

$$\frac{P_n}{P_{ext}} < \left( \frac{\gamma + 1}{2} \right)^{\frac{\gamma}{\gamma - 1}}. \quad (2.71)$$

This pressure ratio is the highest  $P_n$  that allows for choked flow. If the simulation continued after this point, the assumption of choke flow would no longer be valid.

- 5) Initialize all program variables as required, and set  $t = 0$ .

6) Repeat steps 7 through 16 until a stop condition is met: Either the calculated pressure  $P_n$  goes below the input parameter  $P_f$  or the ratio  $P_n/P_{ext}$  is less than the ratio calculated in Step 4.

7) Increment the program's time step,

$$t_{n+1} = t_n + dt . \quad (2.72)$$

8) Calculate the sonic temperature,

$$T_n^* = T_n \cdot \left( \frac{2}{\gamma + 1} \right) . \quad (2.73)$$

9) Calculate the change in pressure,  $dP$ , over the time step  $dt$ ,

$$dP_n = \frac{-\gamma P_n A}{V} dt \cdot \sqrt{\gamma R T_n^*} . \quad (2.74)$$

10) Change the pressure value by adding  $dP$  to the last value of  $P$ ,

$$P_{n+1} = P_n + dP_n . \quad (2.75)$$

11) Calculate the change in mass,

$$\dot{m}_n = \frac{m_n A}{V} \sqrt{\gamma R T_n^*} . \quad (2.76)$$

12) And use that value to change the value of the mass variable for the next time step,

$$m_{n+1} = m_n - (\dot{m}_n dt) . \quad (2.77)$$

13) Calculate the temperature of the next time step by applying the ideal gas law,

$$T_{n+1} = \frac{P_{n+1} V}{m_{n+1} R}. \quad (2.78)$$

16) Check for Stop conditions and output results of current iteration.

17) Output final results.

#### 2.4.2 Analytical Solution

The second approach to the problem was to develop an analytical model that could be used to predict the pressure, temperature, and gas mass some time after the start of the vent process, given initial conditions. The equations that were derived depend greatly on the assumption that the flow out of the SFS tank is choked, and thus the mass flow rate is constant throughout the vent process. In the real world, this assumption would only be valid if the vent area was sufficiently small to allow for choked flow for all but the smallest internal tank pressure. As such, these equations are only approximations for a real system, since in reality the flow of gas out of an SFS tank will not be perfectly isentropic, and may not always be choked. The primary benefit of this part of the analysis is to provide a simple closed-form equation for the venting time required to reach a certain specified pressure. This time can be used as a starting point for system calibrations.

The derivation of the equations starts with the isentropic relations<sup>18</sup> for an ideal gas under the assumed conditions,

$$\frac{T_2}{T_1} = \left( \frac{P_2}{P_1} \right)^{\frac{\gamma-1}{\gamma}} = \left( \frac{m_2}{m_1} \right)^{\gamma-1}, \quad (2.79)$$

the statement of the binomial expansion approximation,<sup>29</sup>

$$(1 + x)^m \approx 1 + mx, \quad (2.80)$$

and the ideal gas law,

$$PV = mRT. \quad (2.81)$$

The incremental changes in properties; pressure, temperature, and mass, are defined as

$$P_2 = P_1 - \Delta P, \quad (2.82)$$

$$T_2 = T_1 - \Delta T, \quad (2.83)$$

and

$$m_2 = m_1 - \Delta m. \quad (2.84)$$

With these equations defined, begin with the isentropic pressure relation,

$$\left( \frac{P_2}{P_1} \right)^{\frac{\gamma-1}{\gamma}} = \left( \frac{m_2}{m_1} \right)^{\gamma-1}, \quad (2.85)$$

which simplifies to

$$\frac{P_2}{P_1} = \left( \frac{m_2}{m_1} \right)^{\gamma}. \quad (2.86)$$

Applying the incremental changes in properties, specifically pressure and mass, to

Equation (2.86) results in the expressions

$$P_2 = P_1 \left( \frac{m_2}{m_1} \right)^{\gamma} = P_1 - \Delta P = P_1 \left( \frac{m_1 - \Delta m}{m_1} \right)^{\gamma} \quad (2.87)$$

and

$$1 - \frac{\Delta P}{P_1} = \left(1 - \frac{\Delta m}{m_1}\right)^\gamma \approx 1 - \gamma \frac{\Delta m}{m_1}. \quad (2.88)$$

Applying the binomial approximation yields

$$\frac{\Delta P}{P_1} = \gamma \frac{\Delta m}{m_1}. \quad (2.89)$$

Next, the mass flow rate equation can be written as

$$\dot{m} = \frac{\Delta m}{\Delta t} \quad (2.90)$$

or

$$\Delta m = \rho a A \Delta t = \frac{m_1}{V} \sqrt{\gamma R T^*} \cdot A \Delta t. \quad (2.91)$$

The definition of sonic temperature,

$$T^* = T_1 \left( \frac{2}{\gamma + 1} \right), \quad (2.92)$$

applied to the change in mass, Equation (2.91), results in

$$\Delta m = \frac{m_1}{V} \sqrt{\gamma R T_1 \left( \frac{2}{\gamma + 1} \right)} \cdot A \Delta t. \quad (2.93)$$

This result is used to cancel the mass terms in the expression for  $\Delta P$ , Equation (2.89),

$$\frac{\Delta P}{P_1} = \gamma \frac{\Delta m}{m_1} = \frac{\gamma A}{V} \sqrt{\gamma R T_1 \left( \frac{2}{\gamma + 1} \right)} \cdot \Delta t. \quad (2.94)$$

The result of Equation (2.94) can be applied to the other isentropic relations. First, the temperature relation is

$$\frac{T_2}{T_1} = \left( \frac{P_2}{P_1} \right)^{\frac{\gamma-1}{\gamma}}, \quad (2.95)$$

which simplifies to

$$\frac{T_1 - \Delta T}{T_1} = \left( \frac{P_1 - \Delta P}{P_1} \right)^{\frac{\gamma-1}{\gamma}} \quad (2.96)$$

and

$$1 - \frac{\Delta T}{T_1} = \left( 1 - \frac{\Delta P}{P_1} \right)^{\frac{\gamma-1}{\gamma}} \approx 1 - \left( \frac{\gamma-1}{\gamma} \right) \frac{\Delta P}{P_1}. \quad (2.97)$$

Applying the binomial approximation to Equation (2.97) results in

$$\frac{\Delta T}{T_1} = \left( \frac{\gamma-1}{\gamma} \right) \frac{\Delta P}{P_1}. \quad (2.98)$$

This approximation can be combined with the previous equation for  $\Delta P$ , Equation (2.94),

to give the expression

$$\frac{\Delta T}{T_1} = \left( \frac{\gamma-1}{\gamma} \right) \frac{\gamma A}{V} \sqrt{\gamma R T_1 \left( \frac{2}{\gamma+1} \right)} \cdot \Delta t. \quad (2.99)$$

Bringing the differential terms to the same side and simplifying the equation results in

$$\frac{\Delta T}{\Delta t} \approx \frac{dT}{dt} = \frac{(\gamma-1)A}{V} \sqrt{\gamma R \left( \frac{2}{\gamma+1} \right)} \cdot T^{\frac{3}{2}}. \quad (2.100)$$

Making the approximation that the changes in temperature and time are very small allows

Equation (2.100) to be integrated,

$$\int \frac{V}{(\gamma-1)A} \sqrt{\frac{\gamma+1}{2\gamma R}} \cdot T^{-\frac{3}{2}} dT = \int dt. \quad (2.101)$$

For convenience, define a constant  $\alpha$  such that



$$\alpha = \frac{V}{(\gamma - 1)A} \sqrt{\frac{\gamma + 1}{2\gamma R}}. \quad (2.102)$$

Equation (2.101) can then be integrated from  $T_i$  at  $t = 0$  and  $T$  at  $t = t$ . The result of this integration is

$$\alpha \int_{T_i}^T T^{-3/2} dT = \frac{-2\alpha}{\pm \sqrt{T}} \Big|_{T_i}^T = t. \quad (2.103)$$

A positive square root would result in a negative time whenever the temperature  $T$  was less than the initial temperature  $T_i$ . This is inconsistent with the physical reality, so the negative square root is used.

$$\frac{-2\alpha}{-\sqrt{T}} \Big|_{T_i}^T = 2\alpha \left( \frac{1}{\sqrt{T}} - \frac{1}{\sqrt{T_i}} \right) = t. \quad (2.104)$$

Solving Equation (2.104) for temperature gives

$$\frac{2\alpha}{\sqrt{T}} = t + \frac{2\alpha}{\sqrt{T_i}}, \quad (2.105)$$

$$\frac{1}{\sqrt{T}} = \frac{t}{2\alpha} + \frac{1}{\sqrt{T_i}}, \quad (2.106)$$

and

$$T = \left( \frac{t}{2\alpha} + \frac{1}{\sqrt{T_i}} \right)^{-2}. \quad (2.107)$$

Equations for pressure and mass as functions of time can be obtained by applying the temperature result to the isentropic relations. Rewriting Equation (2.107) in the form of the isentropic relation gives

$$\frac{T}{T_i} = \frac{1}{T_i \left( \frac{t}{2\alpha} + \frac{1}{\sqrt{T_i}} \right)^2} = \left( \frac{t\sqrt{T_i}}{2\alpha} + 1 \right)^{-2}, \quad (2.108)$$

which can then be substituted in to Equation (2.79) to give the pressure and mass equations:

$$\frac{P}{P_i} = \left( \frac{t\sqrt{T_i}}{2\alpha} + 1 \right)^{\frac{-2\gamma}{\gamma-1}} \quad (2.109)$$

and

$$\frac{m}{m_i} = \left( \frac{t\sqrt{T_i}}{2\alpha} + 1 \right)^{\frac{-2}{\gamma-1}}. \quad (2.110)$$

Equation (2.108), Equation (2.109), and Equation (2.110) may be used to predict the general trend of temperature, pressure, and mass during the vent process.

Additionally, two other equations can be developed to predict the time the vent process will take in order for the gas flow to reach a specified final pressure and how long to

unchoke. The time to reach a specific pressure can be found by rearranging

Equation (2.109) and solving for time. With the final pressure defined as  $P_f$ , the equation becomes

$$\frac{P_f}{P_i} = \left( \frac{t\sqrt{T_i}}{2\alpha} + 1 \right)^{\frac{-2\gamma}{\gamma-1}}, \quad (2.111)$$

which simplifies to

$$\left( \frac{P_f}{P_i} \right)^{\frac{-(\gamma-1)}{2\gamma}} = \frac{t\sqrt{T_i}}{2\alpha} + 1 \quad (2.112)$$

and

$$\frac{t\sqrt{T_i}}{2\alpha} = \left(\frac{P_f}{P_i}\right)^{\frac{-(\gamma-1)}{2\gamma}} - 1. \quad (2.113)$$

Solving this expression for time gives

$$t = \frac{2\alpha}{\sqrt{T_i}} \left[ \left(\frac{P_f}{P_i}\right)^{\frac{-(\gamma-1)}{2\gamma}} - 1 \right] \quad (2.114)$$

as the time for the pressure to reach the value of  $P_f$ .

The condition for unchoked flow is<sup>18</sup> given as

$$P = P_{ext} \left( \frac{\gamma+1}{2} \right)^{\frac{\gamma}{\gamma-1}}. \quad (2.115)$$

Equation (2.111) is used with the unchoked flow condition as the final pressure. The result is the time it takes for the tank to vent to a point where the flow is no longer choked,

$$t = \frac{2\alpha}{\sqrt{T_i}} \left( \left[ \frac{P_{ext}}{P_i} \left( \frac{\gamma+1}{2} \right)^{\frac{\gamma}{\gamma-1}} \right]^{\frac{-(\gamma-1)}{2\gamma}} - 1 \right). \quad (2.116)$$

### 2.4.3 Implementation and Results

The iterative and analytical solutions were coded into an Excel spreadsheet named Blowdown. This spreadsheet allowed the user to input parameters necessary for both kinds of calculations and then view the results for pressure, temperature, and pressurant mass as functions of time. Additional columns at the end of the spreadsheet listed the percent difference between the results of the two approaches.

Table 2.10 shows the inputs used for a typical test case of the spreadsheet code. The inputs are based on a 10 gallon SFS tank using a 1 inch outer diameter vent line and helium as the pressurant gas. The values of Area, Alpha, and Initial Mass at the bottom of Table 2.10 are calculated from the input data by cell formulas.

Table 2.10 Blowdown Spreadsheet Inputs

<b>Inputs</b>	
Tank Volume (cubic feet)	1.34
Line Inner Diameter (in)	0.96
Ratio of Specific Heats	1.4
Specific Gas Constant (ft-lbf/lbm-deg R)	51.5
Initial Pressure (psia)	1000
Initial Temperature (deg R)	500
External Pressure (psia)	14
Final Pressure (psia)	20
Time Step (s)	0.01

When one of the input parameters was changed, the program would automatically update both the iterative and analytical solutions. The time step parameter was used in the iterative solution and it would run until one of its stop conditions was met (step 6 in Section 2.4.1). For the input data that was used, this meant the simulation ended when the flow became unchoked. Pressure, temperature, and mass data for each time step were output to columns in the spreadsheet, along with a summary of the final simulation time and tank conditions.

Once the iterative simulation has been completed, the program copies the time data for the analytical solution, so that the equations developed in Section 2.4.2 can be

evaluated at the same time points as the iterative solution. Unlike the iterative solution, the analytical equations always assume choked flow, so that the results summary includes the time needed for the flow to unchoke and the time needed to reach the final pressure (as set in the input parameter). These results are shown regardless of what stop condition the iterative solution reaches.

The results summaries for the two approaches, using the input data from Table 2.10, are shown in Table 2.11. It should be noted that the iterative simulation gives the final pressure (at the point where the flow becomes unchoked) at 26 psi. The analytical solution assumes (incorrectly, in this case) that the flow remains choked all the way down to the set final pressure of 20 psi. While the time to final pressure value from the analytical solution may not be physically valid, it should be noted that both approaches to the simulation give a very similar result for the time needed for the flow under the specified conditions to unchoke.

Table 2.11 Blowdown Spreadsheet Outputs

<b>Iterative Solution</b>		<b>Analytical Solution</b>
Vent Time:		Time to Unchoke:
0.91		0.92
Final Pressure:		Time to Final Pressure:
26.04		1.01
Final Temperature:		
173.64		
Final Mass:		
0.56		

Charts of the data for both simulations also show a nearly exponential decrease in pressure, temperature, and mass over time, as is to be expected for the venting process that is simulated. This can be seen in Figure 2.13.

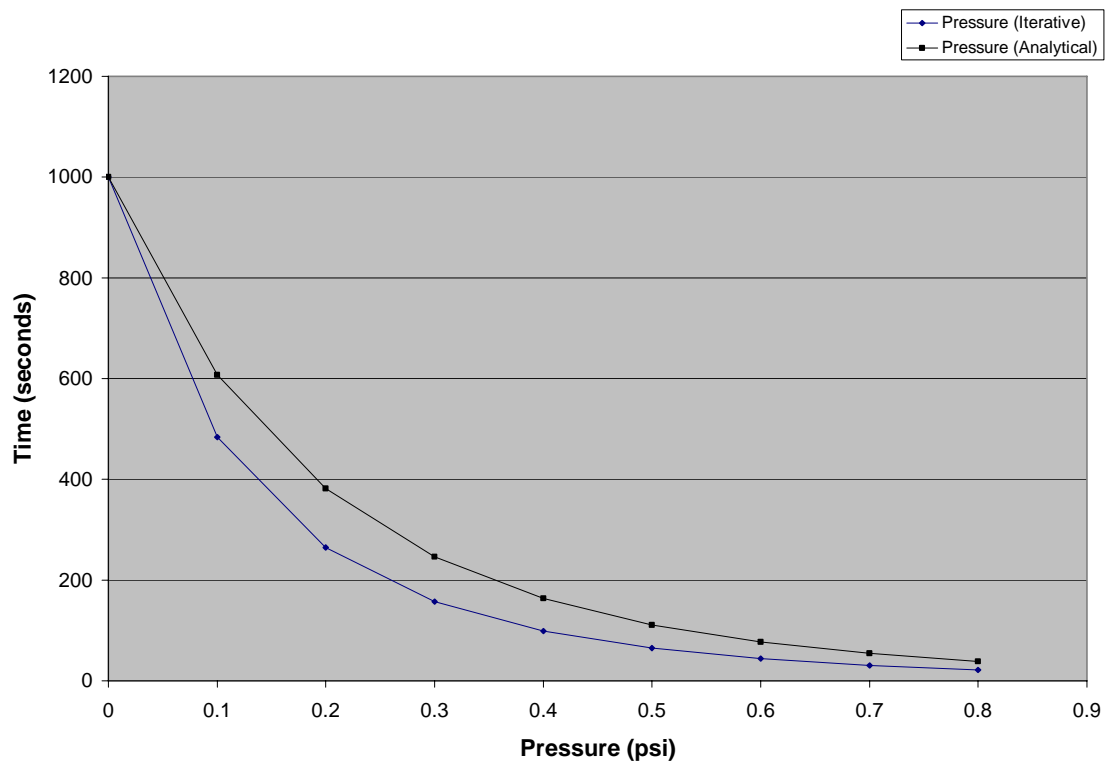


Figure 2.13 Vent Analysis Results Chart

The columns that compare the values of pressure, temperature, and mass data from the simulation show that the percent difference begins small and then increases as the simulations run. In general, the analytical solution predicts a much steeper drop in the quantities than the iterative simulation. A similar trend is observed when comparing

the analytical predictions with data taken from previous SFS testing.<sup>10</sup> This would seem to indicate that while the analytical solution may be more convenient for first-order estimates (a closed-form solution), the iterative solution may be more accurate.

Given that the idealized assumptions inherent in both simulations exclude them from describing the actual vent process in the SFS, it is difficult to validate either kind of simulation. These approaches represent ideal cases of the problem, and as such they are most useful as a first order estimate of how long the vent process will take. Even though the iterative and analytical simulations are not in point-to-point agreement, the final results from the test cases run show that they can be used to arrive at a guess for the unchoke time that is consistent between the two approaches.

## 2.5 SFS Refill Stage Analysis

In addition to analysis of the Vent Stage of the SFS Cycle, an analysis project was also conducted to determine the time needed for the Fill Stage of the SFS Cycle. To this end, the Refill program was developed and implemented in Excel using Visual Basic for Applications (VBA). This section describes the mathematics of the calculations, as well as the program algorithm.

The refill scenario concerns the part in the SFS cycle where the valve between the main propellant tank(s) and an empty SFS tank is opened and propellant is sent into the SFS tank. At this point there will be a small amount of pressure in the SFS tank, but less than in the propellant storage tank. This pressure differential will drive the propellant into the SFS. Some amount of frictional effects will be present in the pipeline, and those effects must be accounted for. Lastly, the vent valve on the SFS will be closed for part of

the refill process, and then later opened once there is enough propellant in the tank to avoid any sloshing out of the open vent valve from the liquid inflow through the diffuser at the bottom of the tank. The vent valve will be opened so that the incoming propellant can push out any remaining pressurant gas, and will remain open until the refill process is complete.

### 2.5.1 Derivation of Equations

The program that was developed simulates two refill procedure options and calculates the time it takes to complete each procedure. Each option has two stages to it, stage one where the vent valve is closed, and stage two where the vent valve is opened. The difference between the options is the criteria used to determine when the vent valve is opened. In option one, the vent valve is opened as soon as the amount of propellant in the SFS tank is enough to cover the six-inch long diffuser at the entrance to the SFS tank. In option two, the vent valve is opened once the remaining pressurant gas equilibrates in pressure with the main propellant tank.

For the pipe system between the main tank and SFS tanks that is assumed in this simulation, the governing equation is given as<sup>30</sup>

$$\frac{P_{main} - P_{SFS}}{\rho} + g(z_{main} - z_{SFS}) = \frac{v_e^2}{2} \left( 1 + f \frac{L}{D} + \sum K \right). \quad (2.117)$$

The symbols used in this equation are summarized in Table 2.12.



Table 2.12 Refill Analysis Symbols

<b><u>Symbol</u></b>	<b><u>Definition</u></b>
$P$	Pressure
$\rho$	Propellant Density
$g$	Acceleration (which may or may not be gravitational)
$z$	Relative Height
$v_e$	Fluid Exit Velocity
$f$	Friction Factor
$L$	Pipe Length
$D$	Pipe Inner Diameter
$\Sigma K$	Total Resistance

The volumetric flow rate can be determined by solving Equation (2.117) for exit velocity,

$$v_e = \left[ \frac{2 \left( \frac{P_{main} - P_{SFS}}{\rho} + g(z_{main} - z_{SFS}) \right)}{\left( 1 + f \frac{L}{D} + \Sigma K \right)} \right]^{\frac{1}{2}}, \quad (2.118)$$

and then substituting that result into the standard flow rate expression,

$$Q = \pi \frac{D^2}{4} v_e = A v_e. \quad (2.119)$$

By knowing the volumetric flow rate, the amount of propellant in the SFS tank can be determined. The Refill program uses an iterative approach to calculate how much propellant has gone into the SFS tank after a given time step.

When evaluated by the program, most of the terms in Equation (2.117) are inputs from the user. The only quantity that remains unknown is the friction factor. It is

assumed that the friction factor is constant and that the pipes are smooth, thus  $f$  is given by the expression<sup>31</sup>

$$f = \left[ -1.8 \log \left( \frac{6.9}{\text{Re}} \right) \right]^{-2}, \quad (2.120)$$

with Reynold's number and dynamic viscosity defined<sup>31</sup> as

$$\text{Re} = \frac{v_e D}{\nu} \quad (2.121)$$

and

$$\nu = \frac{\mu}{\rho}. \quad (2.122)$$

The Refill program uses these quantities to do an iterative calculation to determine the fluid velocity, flow rate, SFS pressure, and the volume of the SFS tank that is filled. It is necessary to do these calculations with time steps because during stage one, when the vent valve is closed, the pressurant gas is compressed as propellant enters the SFS tank. This compression leads to a change in the pressure differential between the main and SFS tanks, which in turn changes the flow velocity and flow rate. It is assumed that the flow rate into the SFS tanks is constant during the time step, so the calculation is redone and updated once enough propellant has gone into the SFS tank to increase the pressure by two psia. Assuming isentropic compression<sup>18</sup> and using the fact that the pressurant mass remains constant, expressions for the change in tank volume and the time step needed to achieve that change in tank volume can be derived. This derivation starts with the isentropic relations

$$\frac{P_2}{P_1} = \left( \frac{m/V_2}{m/V_1} \right)^\gamma = \left( \frac{V_1}{V_2} \right)^\gamma, \quad (2.123)$$

rewritten as

$$V_2 = \frac{V_1}{\left( 1 + \frac{2}{P_1} \right)^{\frac{1}{\gamma}}}. \quad (2.124)$$

The change in volume needed to increase RFS tank pressure by two psia is given as

$$V_{change} = (V_{total} - V_{filled}) \left( 1 - \left( 1 + \frac{2}{P_1} \right)^{-\frac{1}{\gamma}} \right). \quad (2.125)$$

The time step is derived from this volume and the flow rate,

$$Q = \frac{V_{change}}{dt} \quad (2.126)$$

and

$$dt = \frac{V_{change}}{Q}. \quad (2.127)$$

### 2.5.2 Implementation and Algorithm

The goal of the Refill program was to take in inputs for an SFS system and compare the two refill options based on how long it took to refill the tanks. The inputs to the program are Main Tank Pressure (psia), Initial SFS Pressure (psia), External Pressure (psia), Propellant Density (lbm/ft<sup>3</sup>), SFS Tank Radius (ft), Pipe Inner Diameter (ft), Change in Height (ft),  $\Delta z = z_{main} - z_{SFS}$ , Acceleration (ft/s<sup>2</sup>), Pipe Length (ft), Total K,

and Dynamic Viscosity (lbf-s/ft<sup>2</sup>, represented by the symbol  $\mu$ ). Other necessary quantities can be calculated from these inputs.

The algorithm of the program is designed to evaluate both options in a single run. For the expected SFS tank geometry, it is assumed that the stop condition for option one (filled tank volume covers the diffuser) will occur before the stop condition for option two (SFS tank pressure equilibrates with main tank pressure). The iterative simulation will run until stage one of option one is complete, then shift into finishing the rest of the simulation as stage one of option two.

The stage two part of both options can be evaluated right away once the respective stop conditions are met. Stage two indicates that the vent valve is open, so the internal SFS tank pressure stops increasing as the refill proceeds, and the flow rate into the tank doesn't need to be recalculated any more. The time to fill up the rest of the tank is then determined immediately from the flow rate and the remaining volume to be filled.

The detailed algorithm of the program is explained in the following steps:

- 1) Declare variables and read in input data.
- 2) Determine  $f$  using Equation (2.120).
- 3) Begin iterations to fill the SFS tank, recalculating the flow rate when the volume changes enough to increase the SFS tank pressure by two psia, outputting results to the screen as they are done.
- 4) During this loop, make a check to see if the volume filled is greater than that needed to cover the six-inch long diffuser (option one stop condition for stage one).

- 4a) If so, pause the iteration to calculate the stage two time for option one. This is done by calculating the flow rate using the external pressure instead of the SFS tank pressure and finding the time necessary to fill the remaining volume at that flow rate.
- 5) Once the option one calculations are complete, continue the iterations for the option two scenario, until such time as the pressure in the SFS tank is just below the main tank pressure.
- 6) After the stop condition for option two is reached, the loop exits and the stage two time for option two is calculated and output the same way as it was for option one.
- 7) Program finishes with an output of the time it takes to complete stage one (vent valve closed), stage two (vent valve open), and the total time for both options one (fill to above diffuser level) and two (fill to pressure equilibration).

The assumptions in the operation of this program are that the system uses a Spherical SFS tank, change in height measured from the exit of the main tank to the entrance of the SFS tank, the diffuser itself occupies no volume in the SFS tank (the cutoff condition is the volume necessary to cover a six-inch depth hemisphere inside the larger spherical SFS), for both options it is assumed that there is a six-inch deep ullage space left in the SFS tank (this volume is equal to that necessary to cover up the diffuser), the propellant used is incompressible, the temperature of the system does not cause any chemical or pressure changes to occur.

### 2.5.3 Results

Included below in Table 2.13 are the results of a test run using an approximately 10-gallon SFS tank with kerosene. The test case that was run indicates that option one

provides the faster refill time. This is because by allowing the tank pressures to equilibrate (as is done in option two) the flow rate decreases for a longer time before the vent valve is opened and the flow rate reaches its final value. In addition to the final results, the program also includes the step-by-step simulation results, which are shown in Table 2.14.

Table 2.13 Refill Program Inputs and Outputs

<b>Inputs</b>	
Main Tank Pressure (psia)	50
Initial RFS Pressure (psia)	30
External Pressure (psia)	14.7
Propellant Density (lbm/ft <sup>3</sup> )	51.5
RFS Tank Radius (ft)	1.34
Pipe Inner Diameter (ft)	0.33
Change in Height	5
Acceleration (ft/s <sup>2</sup> )	32.2
Pipe Length (ft)	5
Total K	3.07
Dynamic Viscosity (lbf-s/ft <sup>2</sup> )	4.49E-05
<b>Outputs</b>	
<i>Option One</i>	
Stage I Time	0.65
Stage II Time	1.99
Total Refill Time	2.64
<i>Option Two</i>	
Stage I Time	1.34
Stage II Time	2.11
Total Refill Time	3.45
f	0.002
Initial Reynolds #	368653
Area (ft <sup>2</sup> )	0.08
Total Volume (ft <sup>3</sup> )	10.07
Diffuser Volume (ft <sup>3</sup> )	1.47
Stage II Flow Rate (ft <sup>3</sup> /s)	3.51
Stage II Velocity (ft/s)	40.31

Table 2.14 Refill Program Simulation Results

Time (s)	Velocity (ft/s)	Flow Rate (ft <sup>3</sup> /s)	SFS Pressure (psia)	Volume Filled (ft <sup>3</sup> )
Option One				
0	31.04	0	30	0
0.14	30.90	2.69	32	0.38
0.27	29.44	2.56	34	0.72
0.40	27.91	2.43	36	1.04
0.53	26.30	2.29	38	1.33
0.65	24.58	2.14	40	1.59
Option Two				
0.77	22.72	1.98	42	1.84
0.90	20.71	1.80	44	2.06
1.03	18.47	1.61	46	2.27
1.17	15.92	1.38	48	2.47
1.33	12.88	1.12	50	2.66

In summary, the Refill program developed for this analysis allows for rapid calculation of the time necessary to refill an SFS tank. Preliminary test cases have shown that for the size of SFS tanks we have looked at the refill time can be on the order of a few seconds. In the future it may be necessary to revise this program's assumption of spherical tanks, but for now the tool is still very useful for confirming expectations and calculating different possible systems.

## CHAPTER 3

### SOFTWARE: INTEGRATED CONTROL AND MONITORING PROGRAM

The original test bed at MSFC<sup>10</sup> utilized two separate computers for hardware control and data acquisition. After the conclusion of the first test program, it was decided to integrate the two functions. The resulting computer program is called the SFS Integrated Control and Monitoring (ICM) program and was implemented using LabVIEW. The general purpose of the ICM was to control the operation of a three-tank SFS test bed, as well as monitor and record pressure, temperature, and flow rate data. Additionally, the program was intended to provide the user interface for the fail-operational tests, and as such the program was designed to inject a simulated failure into a normal operation sequence, detect that failure (through several means), and respond automatically to that failure.

#### 3.1 Design

The overall design of the ICM was motivated by the goal of improving the old control system and developing a means to automate both normal experiment operation and fail-operational testing. This section describes how the ICM was designed to accomplish these goals and the reasons for the assumptions and algorithms built into the program.



### 3.1.1 Program Features

Because of the scheduling of the SFS research project, the ICM was designed for the most part in the absence of a functional SFS test bed. As such, certain assumptions had to be made regarding the test bed architecture in order to determine what the ICM algorithms should be. These assumptions were made during program design and development; the motivation behind those assumptions and the effect they had on the program are as follows.

The very first assumption that had to be made was in regards to what the program would control; specifically, what valves would be used and how they would be controlled. The plan for the SFS test program was to reconstruct the test bed that was originally at MSFC, so the ICM program was based directly off the previous SFS test set up. As such, control of the SFS apparatus was assumed to require control of the twelve valves on three tanks, with four valves per tank. Three tanks are used to allow for a fail-operational mode. The program assumes that the valves on each tank are of the following types: Vent, Drain, Fill, and Pressurize. Each valve controls a specific function for the tank, e.g., the Vent valve is used to allow for the venting of pressurant gas from the tank when opened. Also in keeping with the hardware decisions of the original test bed, it was assumed that the Drain, Fill, and Pressurize valves would all be normally closed (open when energized, and closed when de-energized). The Vent valves are assumed to be normally open and have the opposite configuration. The reason for this switch was a safety measure: if the system ever lost power, the flow of liquid and pressurant would be halted immediately, and the gas pressure would not be allowed to stay inside the SFS tanks. In terms of the program, it was found that it was easier to code the valve

commands with “open equals true” logic, and so a special case for the Vent valves had to be coded to flip the command before any control signal was sent to the valves.

With the question of what the program controls answered, the next assumption that had to be made was how the program would control the system. Again, taking after the original control program, it was decided to operate the valves by running a sequence of valve commands over a period of time. The sequence was implemented as a two-dimensional array of Boolean variables, with twelve columns representing each of the twelve valves and a row for each part of the sequence. Each row had a time value associated with it (stored in a parallel array) that would tell the program to execute the commands stored in the row at a specific time.

In ICM terminology, a collection of valve commands is called a “Sequence.” In the original control program the Sequence would be read in from a file, stored in the program’s memory, and then executed when the program was run. The ICM copied this procedure for basic functionality, but expanded on it to implement other features.

Reading a Sequence in from a file (termed a “File Sequence”) is useful for some purposes, such as static testing, but does not allow for flexibility in operation. To allow for the fail-operational testing, the ICM had to be able to detect when a failure had occurred in one of the valves, and automatically alter the remainder of the Sequence to respond to that failure. Simply using a File Sequence would not allow for that, so other sequence types were planned and implemented.

The main Sequence feature intended for use with the fail-operational testing was one wherein the program generates an SFS sequence from user inputs specifying how long each part of the SFS cycle lasts. This Sequence type is called an “Auto Sequence.”

Full description of the Auto Sequence requires that some other terms coined for this program be defined. First, a “Tank Cycle” refers to the sequence of states that each SFS tank goes through: Drain, Vent, Fill with Vent Closed, Fill with Vent Open, Pressurize, Idle, and back to Drain. In turn, each of those states is referred to as a “Cycle Stage,” or simply as a “Stage.” The entire series of Stages listed above is called a “Tank Cycle.” Differing from the Tank Cycle is the “SFS Cycle,” which refers to a Sequence that returns the entire SFS system back to its original state. The SFS Cycle begins with the Drain Stage of tank 1, and ends with the Drain Stage of tank 3 (if the cycle begins again, tank 1 will immediately enter the Drain Stage after tank 3). Likewise, a Tank Cycle is a sequence that returns a single tank back to its original state.

What differentiates the Auto and File Sequences is that when the Auto Sequence is generated, the program enforces the correct order of Stages in all parts of the SFS Cycle. A File Sequence has no specific order beyond whatever the user put in the file to begin with, and so such a Sequence does not necessarily have to be a normal way to operate the SFS. The Auto Sequence is generated programmatically, based on user timing inputs specifying how long each part of the SFS Cycle lasts. Since the SFS Cycle order is enforced, the program can also use the timing inputs to determine what stage the system was in when a failure is detected, and then modify the remaining part of the sequence accordingly. Failure response is only implemented for the auto sequence, and the program responds to real and simulated failure in the same way.

In order for the proper enforcement of Sequence order to be done in the Auto Sequence, certain constraints must be put on the timing inputs used by the program.

First, the time specified for the Drain Stage minus the time specified for Drain Stage Overlap (time when next tank in the Sequence is draining at the end of the previous tank's Drain Stage) must be greater than the sum of the times specified for the Vent, Fill, and Pressurize Stages. The Fill Stage includes Fill with Vent Closed and Fill with Vent Open (abbreviated as Fill-1 and Fill-2 Stages, respectively). The Vent must be initially closed to prevent the propellant from sloshing out of the tank, and it must be opened later in the Fill Stage to alleviate pressure build-up due to inflow of propellant. As such, the time specified for the Fill Stage must be greater than or equal to the time specified for the Fill-2 Stage (the equal to condition is for the case where the Fill-1 Stage time is zero and the entire Fill Stage is taken up by Fill-2).

A third type of Sequence was also planned for the ICM, but was not implemented. The "Condition Sequence" is defined as one generated by using data from tank level, liquid mass, or volume sensors, as well as other means, to determine the sequence of valve controls. This kind of sequence would be difficult to implement and impossible to verify without a working SFS test bed, and as such it was not implemented with the other ICM features. Since the requisite sensors were not available later when the SFS test bed was reconstructed, the Condition Sequence was left as an unimplemented feature. It was included in the design because a real SFS on a propulsion system would no doubt have liquid level sensors.

One of the main goals behind the development of the ICM was to allow for fail-operational testing. The criteria that define a failure are defined as follows. In terms of what the ICM is concerned with and can control, the only failure that can occur is one that happens in the valves. In reality, it is entirely possible that a failure can occur

elsewhere in the system, a loose fitting causing a leak for example, but that is not included in the programmatic definition of a failure because such an event is beyond the ICM's ability to control. A failure in a valve is defined as a case where the valve is actuated opposite of what it is intended to be. For example in the Drain Stage, the Drain valve is intended to be open. If it is not, then the Drain valve has failed, and the program must adjust the sequence to respond to this failure.

By defining failures as actions contrary to expectations, the program lends itself very easily to simulated failures. When an Auto Sequence is initiated, the program generates the Sequence and stores it in memory, running through the commands and executing the valve position changes as needed. An additional input is included that allows the user to specify where and when a failure should take place. When the appropriate conditions are met, all the ICM has to do is intercept the valve control command and change the affected valve's value, before sending the command to the valve hardware. The result of this is to have the program's right hand, in effect, not know what the left is doing. One part of the program has told a valve to behave in a fashion contrary to expectations, while another part of the program is keeping track of what the valves are expected to do (based on the original sequence data). When there is a discrepancy, the failure response procedures activate to remedy the problem. By injecting a simulated failure in this manner, it is possible to test the effects of particular failures in a systematic and repeatable fashion, without actually requiring faulty valves.

The final system assumption that had to be made was in regard to the data that would be acquired by the program. This part of the program changed slightly during reconstruction of the test bed to accommodate the newer system. In the original program,

the data recorded were: pressure from each of the three tanks, temperature on one of the tanks, temperature and pressure from the liquid inlet, temperature and pressure from the liquid outlet, and flow rate at the outlet. In the original program flow rate was calculated from a pressure measurement across an orifice, as no flow rate sensor was available. The ICM is configured to measure temperature and pressure on all three tanks, the liquid inlet, the liquid outlet, and the air inlet. Additionally, flow rate is calculated from pressure and temperature readings on the liquid outlet which is also equipped with a venturi tube.

### 3.1.2 Interface

In LabVIEW terminology, the user interface is termed the front panel. The front panel of the ICM is divided into three sections by a tab object. The three tabs are named “Settings” (Figure 3.1), “Data” (Figure 3.2), and “Sequence” (Figure 3.3). The front panel is divided into two sections: primary indicators on the left and function tab on the right. The primary indicators are the valve states (open equals true), temperature and pressure values, and failure state (failure equals true) for each of the three tanks. The tab object allows different parts of the user interface to be hidden from the user when not in use. The primary indicators are always visible, and the valve state indicators are set so they can be manipulated by the user when a program sequence is not running to manually control the valve positions.

The Settings tab, shown in Figure 3.1, contains the user controls for selecting, configuring, and running a Sequence, as well as inputting the text that is written in the header lines of an output file. The type of Sequence that will be used by the program is controlled by the ring control labeled “Sequence Type.” Selecting a specific Sequence

type directs the program to use or ignore other inputs on the settings tab. For example, when an Auto Sequence is selected, the program will use the “Cycle Times” inputs and ignore the “Filename” input, the later being specific to a File Sequence type.

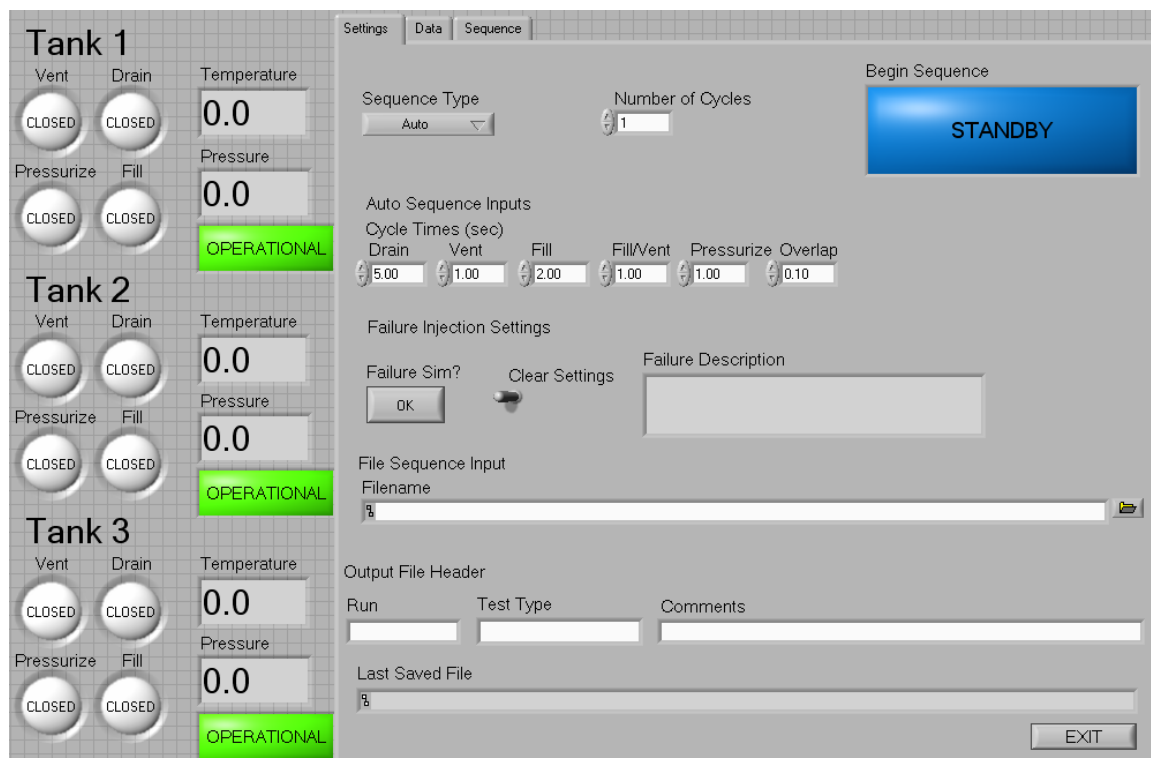


Figure 3.1 ICM Settings Tab

Inputs that are specific to a Sequence type or other program function are labeled on the tab. These groups include “Auto Sequence Cycle Times,” “Failure Injection Settings,” “File Sequence,” and “Output File Header.” For an Auto Sequence, Cycle Stage timing is determined by the six numeric controls: “Drain,” “Vent,” “Fill,” “Fill/Vent,” “Pressurize,” and “Overlap.” Each control represents how long (in seconds)

each Cycle Stage should last. Note that the “Fill” time represents the sum of the Fill-1 and Fill-2 Stage times, while the “Fill/Vent” time represents the duration of the Fill-2 Stage, how long at the end of the Fill Stage the Vent will be open. There is not an actual Overlap Stage, so the “Overlap” control is used to specify the length of time that the Drain Stage of the next tank in the sequence will overlap with the Drain Stage of the first tank.

The “Failure Injection Settings” include the “Failure Sim?” control which, when clicked, brings up a window to let the user select what type of failure and when to inject it (this window is generated by a sub-VI, or a LabVIEW “Virtual Instrument” program that is activated by another LabVIEW VI). Also included are the “Clear Settings” switch and the “Failure Description” text box. “Failure Description” contains a text description of the failure that is to be injected. An empty text box will mean no failure is to be injected, and “Clear Settings” can be used to erase any previously set up failure injection. The program code uses “Failure Description” as a check to see if there is any set failure to be injected, which happens any time there is a non-empty string stored in the variable.

The input “Filename” applies only to a File Sequence. This path control allows the user to select the file that is to be used for the sequence. The program expects that the file selected is a tab delimited text file. Each row of the file should first contain a time entry, followed by twelve ones or zeros. The time entry specifies the elapsed time when that group of valve commands should be enacted. The ones and zeros correspond to the Boolean values for the valve commands (one equals true equals valve open). The order of these Booleans is the same as for the Sequence tab, see Figure 3.3 and the associated description below.



Lastly, the “Output File Header” controls are string controls that let the user add in optional text to the start of an output data file. The header line will include text describing what test “Run” the data file is for, what the “Test Type” is, and any other “Comments.” When the user has initialized the controls, the “Begin Sequence” control is used to tell the program to begin the sequence that the user has selected, with the settings made on the other controls.

The Data tab (shown in Figure 3.2) displays the data that is collected by the program in numeric and graphical form, as well as allows the user to start and stop data reporting to an output file. The emergency stop button is also on this tab. By default, once the user initiates a Sequence, the program will switch to view this tab.



Figure 3.2 ICM Data Tab

Arranged vertically on the left side of the tab are the propellant and pressurant indicators: temperature and pressure readings for the liquid inlet, liquid outlet, air inlet, and flow rate at the liquid outlet. Located at the top right of the tab is the emergency stop button. When the user clicks on this control, all of the valves will be commanded to go to a safe position (de-energized: all vents open, all others closed) and the program will halt execution. To the right of the the propellant and pressurant indicators are fields that display the “Elapsed Time” for the current sequence, a “Clock” reading for the current time, and a log of all of the errors and responses that have occurred since the start of the last Sequence.

Most of the data tab is taken up by the “Data Plot,” which is used to display the various data that is collected by the program to the user in real time. While the program is running, this chart will be continuously updated as new data comes in. The range of the “Data Plot” may be modified by using the “YMax” and “YMin” controls to the right of the chart.

Lastly, at the bottom left of the tab are two Boolean indicators and at the bottom right a control button. The “Acquire Data” button tells the program to start outputting recorded data to a file when first clicked, and then to stop the file output when clicked again. The “Display” indicator will light up when the data is only being displayed to the screen. “Display and Acquire” will blink when the data is being displayed on screen and output to a file.

The “Sequence” tab (shown in Figure 3.3) displays the currently loaded valve command sequence. The one-dimensional vertical numeric array on the left contains all of the timing data for a particular part of the sequence. The two-dimensional Boolean

array contains the actual valve position commands for the entire sequence. In this array, in keeping with the program conventions, a Boolean value of true means the valve should be open. Each row of the 2D Boolean array corresponds to the element in the 1D numeric time array to the left of that row. For example, a row with an element of value 0.1 represents the valve settings that should be set when the elapsed time reaches 0.1 seconds. Likewise, each column in the 2D array represents one of the valves for one of the tanks as given in the text above each column. V1 stands for Tank 1 Vent, and so on for the other columns. Since the program reads row by row, the time array must have elements ordered such that the highest numbers are in later rows.

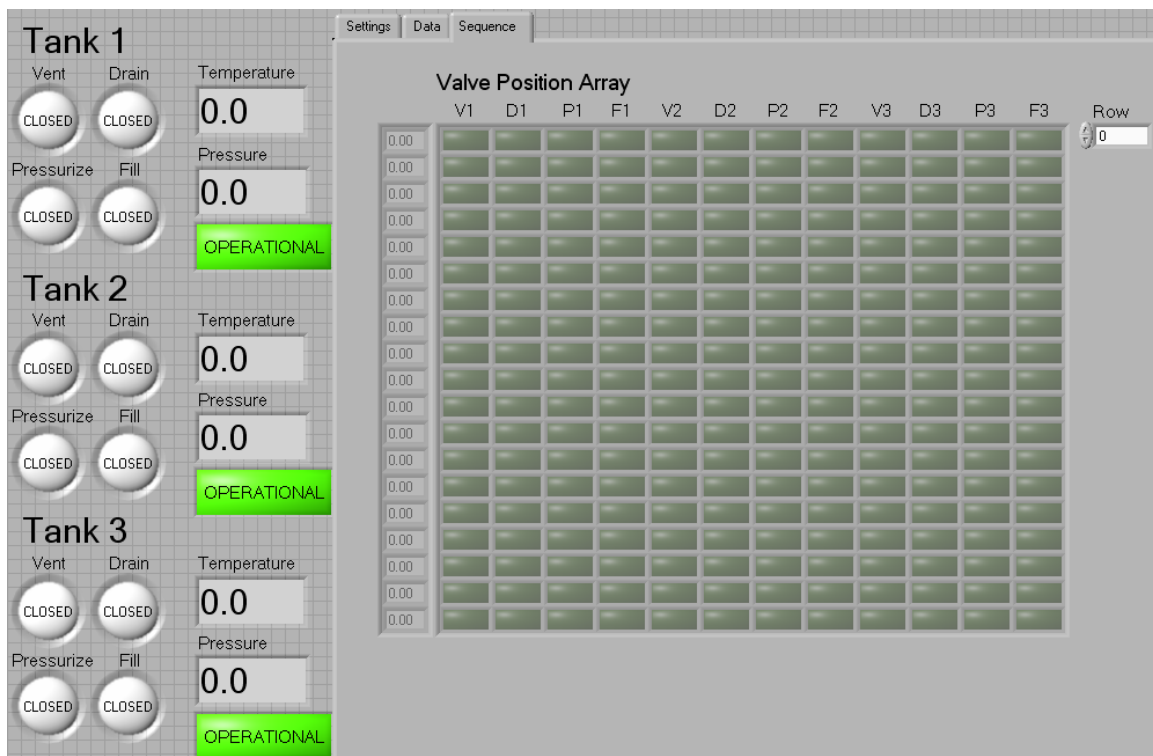


Figure 3.3 ICM Sequence Tab

### 3.1.3 Algorithms

The ICM is designed in such a way as to allow for continuous operation, so the user does not have to restart the program for every test run. The program allows for user input, and then runs the necessary operations based on that input, finally returning to the original state that will wait for the user to command the next set of operations. To achieve this design goal, the program operates in a continuous loop that is stopped only by the Emergency Stop button or the standard Exit button (which is programmatically linked with the Stop button). The operation of the program within the loop is depicted schematically below in Figure 3.4.

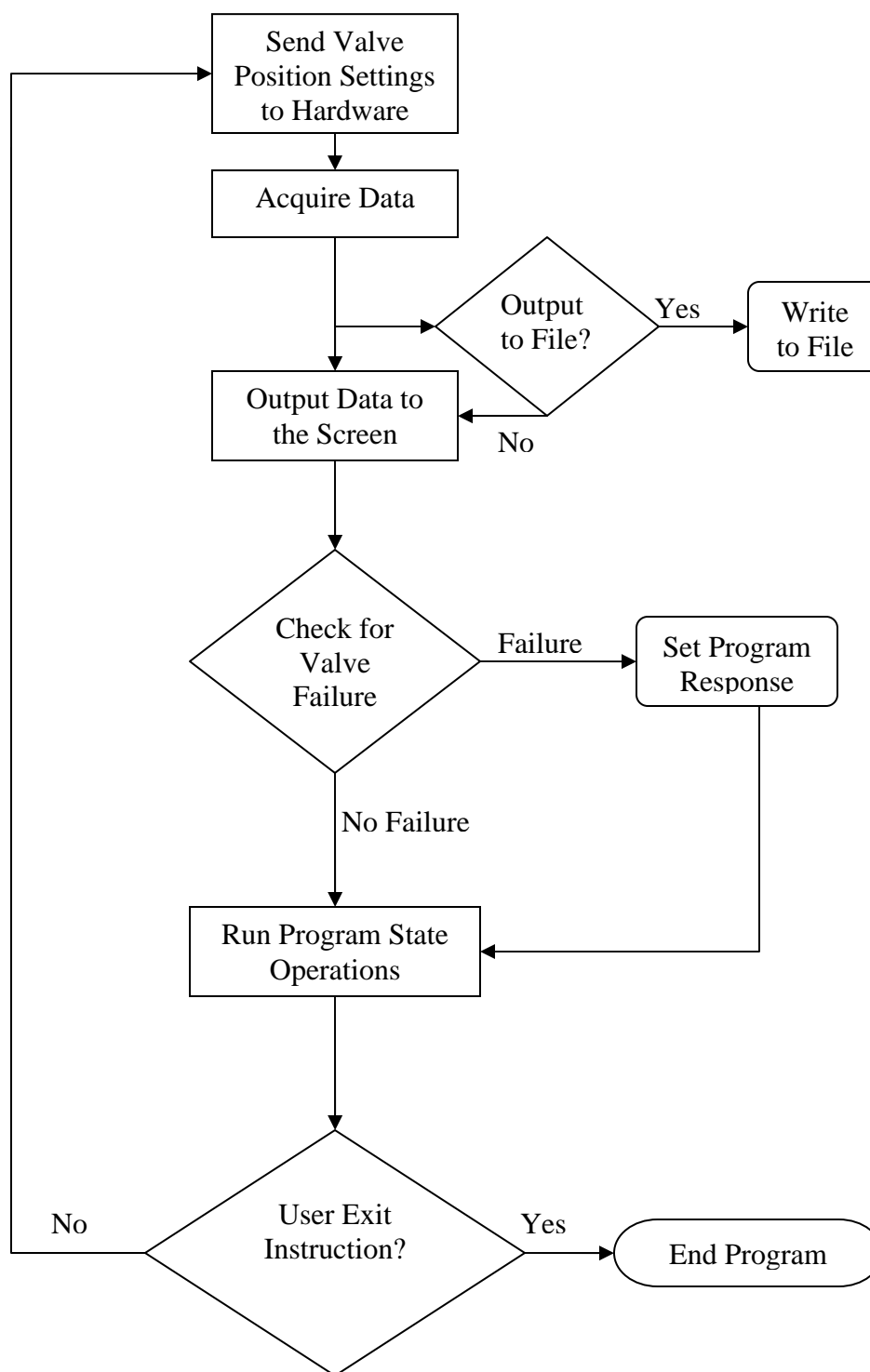


Figure 3.4 Flow Chart of ICM program operation

The important aspects of the program flow chart in Figure 3.4 are as follows. Each time through the loop the program will update the valve control signals. This means that the valve position commands must be continually stored by the program. For the most part, the hardware commands and valve settings will not change from one iteration to the next, because the program will run faster than the Sequence requires, but the intent of continuous updates is so that the program only has to call the hardware interface at one point in the program, as opposed to numerous special cases every time there is a change to the position settings. Hardware interfacing is handled by a sub-VI so that the top level of the ICM can remain as general as possible. This feature is also to allow for simple changes should new hardware be used with the system. The only assumptions that the ICM makes have been documented earlier. How specific valve and control hardware interprets the ICM commands can be handled in the sub-VI. If new hardware is used, then the valve control sub-VI can be rewritten or replaced entirely, and no changes need to be made to the ICM.

Like the valve control signals, data is also acquired continuously throughout the operation of the program. Each time the main loop executes, data will be collected, output to the screen, and output to a file (if the user has instructed the program to do so). For generality, data acquisition is handled by a sub-VI to allow for hardware specific code to be isolated from the rest of the program.

Since in a real world situation a failure can occur at any time, the failure detection procedures must run continuously. After the newest valve settings have been sent to the hardware and the data have been collected, the program must check to see if there has been a failure and respond accordingly. The ICM is intended to have more than one way

to detect failures, called multiple tier detection. Similar to hardware control and data acquisition, the failure detection function is handled in a sub-VI. Doing so allows for the specific detection code to be separate from the main ICM program, so the various detection tiers can be implemented and integrated into the program without requiring a rewrite of the ICM code. Currently, the only form of failure detection that has been implemented is termed “Basic Failure Detection.” Basic detection means the program looks at the settings on the digital I/O port that controls the valves, and compares those settings to what is expected. This kind of detection only detects injected failures, ones where the program has overwritten a control signal. The tiers of failure detection considered for the ICM are summarized below in Table 3.1.

Table 3.1 Failure Detection Tiers

<b><u>Failure Tier</u></b>	<b><u>Description</u></b>
Basic	Determine when failure has occurred based on settings of the DIO card, as compared to expected settings.
Electronic	Valve control signals monitored, and failure detected when a signal is other than expected.
Actuation	Actuation sensors placed on all valves, and failure detected when the sensor readings are contrary to those expected.
Sensor	Expected temperature and pressure profiles are programmed in to the ICM, and failure detected when the data from these sensors is outside expected parameters.

The last operation in the flow chart before the program decides if it will continue or not is labeled “Run Program State Operations.” The overall intent is to have the program take commands, run an operation according to those commands, and then return.

As such, any program operations that do not need to occur every time the loop executes (i.e., the items listed above) must occur when the program is in a particular “State.” The program state describes what kind of function will be executed.

#### 3.1.4 States

A program State variable is maintained in order to tell the program which set of functions to execute each time through the loop. The States and the functions that go with them are summarized in Table 3.2. Additionally, flow charts on the following pages depict how each program State will operate. The Idle State is shown in Figure 3.5, the File and Auto States are shown in Figure 3.6, and the Response State is shown in Figure 3.7.

It is important to note that while Figure 3.6 describes the operation of both the File and Auto Sequences, there are significant programmatic differences between the two Sequence types. The difference originates in how the Sequence data is generated. In the case of a File Sequence, the user inputs a file name, and the Sequence data is read in from that file during the Idle State (see Figure 3.5). In the case of the Auto Sequence, the program has to initialize the Sequence data before the Sequence is run. This is done in the Idle State by calling a sub-VI to do the necessary calculations for the Auto Sequence timing.



Table 3.2 ICM Program States

<b>State</b>	<b>Function</b>
Idle	The default State wherein the program waits for user input. The program will look for such input from the controls on the front panel and change the state accordingly. Some features, such as setting up a simulated failure injection, will not require a change in State.
File	Runs a File Sequence from the specified file. After completion the State will be reset to Idle.
Auto	Runs an Auto Sequence from the specified timing inputs. After completion the state will be reset to Idle.
Condition	Runs a Condition Sequence based on sensor data. Since the data required to assess the condition of each tank is hardware specific, that code will be handled in a Sub-VI. After completion the State will be reset to Idle. This State is not implemented.
Response	Handles any detected failure in any of the valves. Currently, the Response State may only be triggered when a failure is detected during an Auto Sequence.

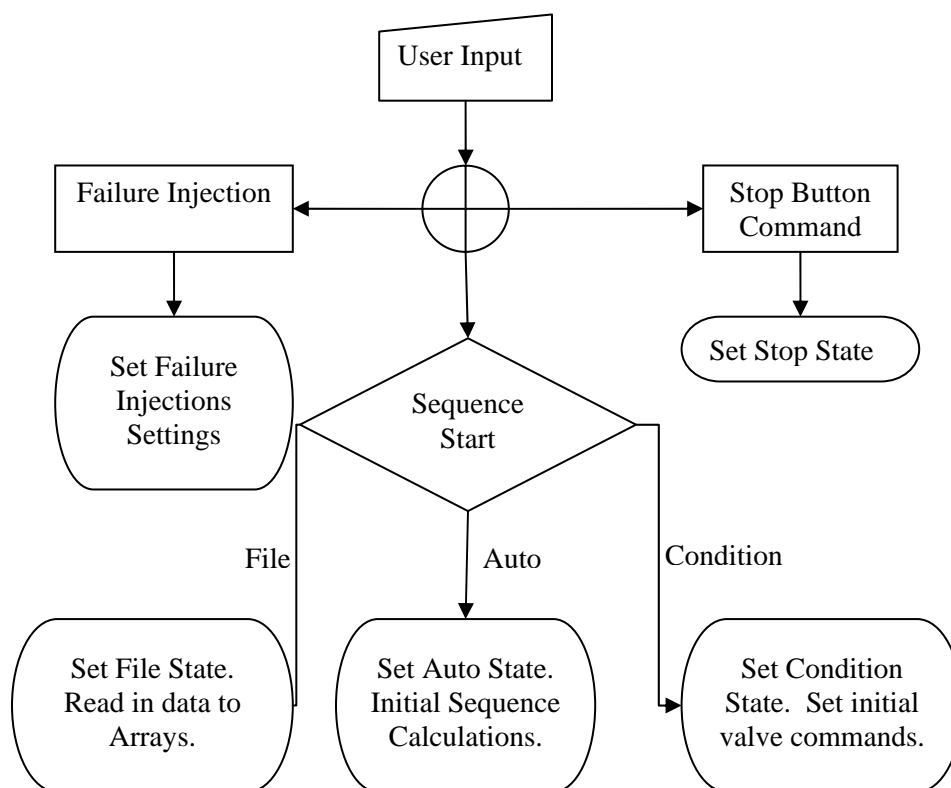


Figure 3.5 Flow Chart of Idle State

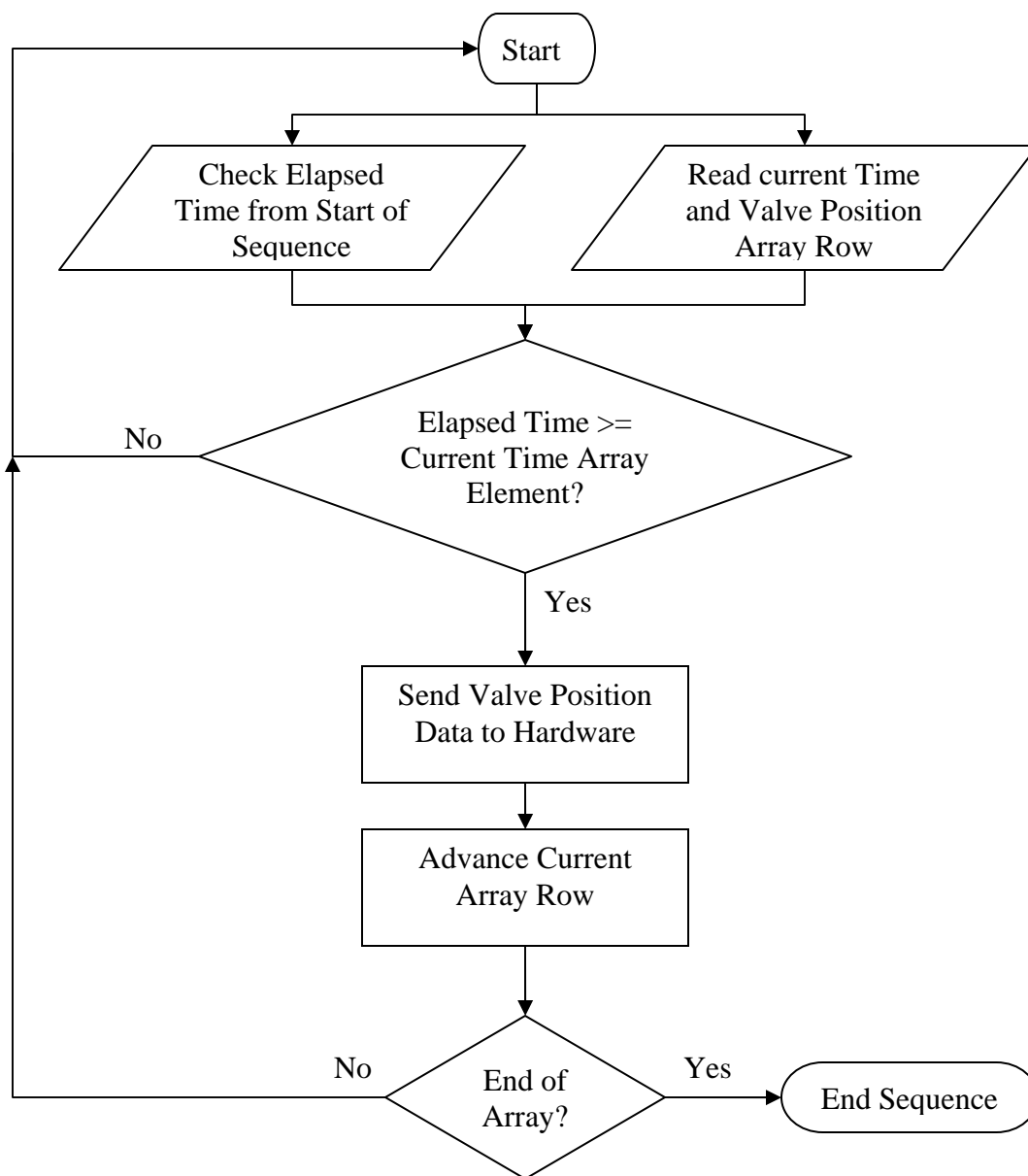


Figure 3.6 Flow Chart of File and Auto States

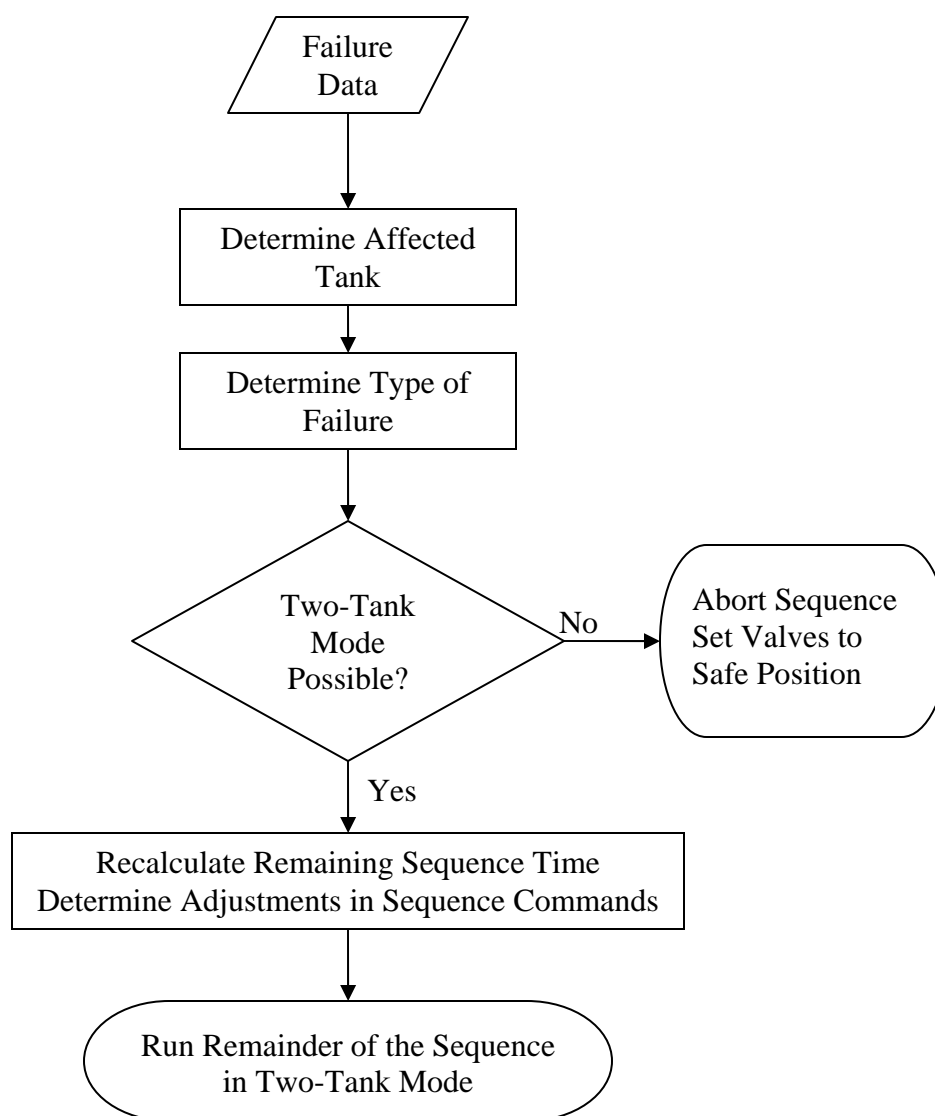


Figure 3.7 Flow Chart of Response State

### 3.2 Implementation

Major coding for the ICM was completed in the time between when the MSFC experiments ended and when the test bed was reassembled at the UAH Propulsion Research Center (PRC). Since the new test bed was a modified version of the old one, only small changes needed to be made to the ICM once the test bed was reassembled.

The operation of the ICM program follows the general form specified in the flow chart of Figure 3.4. What this means programmatically is that at program start-up the variables are initialized to default values and then the program enters a while loop. This loop is controlled by the “Stop” and “Exit” buttons on the front panel, and the program will continue to run until one of these buttons is pressed. Within the while loop is a LabVIEW sequence structure (not to be confused with Sequence as used with the SFS). What this structure does is enforce that the LabVIEW code blocks execute in a specific order, as well as allows similar code to be grouped together. There are four parts to this sequence structure: hardware control and failure injection, data acquisition and reporting, failure detection, and state functions. As a complete overview of the code is beyond the scope of the thesis, this section is intended to summarize the implementation and point out any changes made to the design and the reason for those changes.

#### 3.2.1 Hardware Control

In LabVIEW, a Sequence Structure is divided into frames, and viewed one at a time like the pages of a book. On each frame the code is executed, and the program will only move on to the next frame once all the objects on the current frame have finished their tasks. Frame 0 of the ICM’s sequence structure contains the code for valve

hardware control. Following the program design, the program was implemented in such a way that the hardware control code is called once during every execution of the program loop. The reason for this was that changes to the settings to the valves can be made at several places in the code, and keeping the hardware control code in one section eliminates the need for duplicate coding elsewhere in the program.

In order to further simplify the hardware control code, the ICM handles interfacing with the valves through the `valveControl` sub-VI. Valve settings are determined by the information stored in the twelve Boolean valve indicators. This data is compressed into an array for use with `valveControl`. (Note that the convention for sub-VI naming followed for most of the ICM sub-VIs is to have the name as one word with the first word start with a lower case letter, e.g., valve control becomes `valveControl`.)

The `valveControl` sub-VI is very simple programmatically. The only task of this VI is to take the Boolean valve position data, translate that data to a binary code, and write the result to the DIO card. This function is encapsulated within a sub-VI in order to allow the ICM to be flexible with the valve control hardware. The current version of `valveControl` was implemented for a National Instruments PCI6503 card. If other hardware is used in the future, the ICM does not need to be modified, only the code in `valveControl`.

Implementation of `valveControl` was done without any real valves to actuate, so the hardware control features were done by focusing on the electronic signals that the computer would send out to a relay panel, which would ultimately control the power signals to the physical valves. The DIO card that was used for the old control system was retained and used as the model hardware controller. The outputs of this card were wired

to a breadboard with 15 LEDs, twelve for the valves and three more for failure indicators. The use of LEDs on the breadboard allowed for visual confirmation that the program was sending the expected signals, to the right circuits, and in the right order.

### 3.2.2 Data Acquisition

After the code on frame 0 of the sequence structure is completed, the ICM moves to frame 1, which handles data acquisition and output. From the ICM's standpoint, data acquisition is fairly simple: the dataAcq sub-VI handles all of the hardware specific details and outputs a numeric array. This data in this array is extracted and reordered by the ICM for display on the Front Panel's Data Plot.

As valveControl does for hardware interfacing, the dataAcq sub-VI encapsulates all of the data acquisition functions for the ICM. Unlike the other ICM sub-VIs, this program was implemented very late in the development of the ICM, after the data acquisition system for the test bed (see Section 4.2.4) was assembled. This unavoidable delay was the motivation behind making data acquisition a sub-VI module. This sub-VI requires no inputs from the ICM and only returns a numeric array, "Output." By using this configuration, the ICM only has to deal with how the array data is displayed to the user and written to a file. The actual generation of the array and interfacing with hardware to get the data is handled by dataAcq.

The current implementation uses a LabVIEW Express VI called DAQ Assistant to monitor twenty four channels of data. The Express VIs are user-configurable tools that simplify complex functions. The output from DAQ Assistant is a collection of the waveforms that are acquired from the data acquisition hardware. The main task of

dataAcq is to sort out the waveforms and average them to get individual data points. After that, all of the data is concatenated into an array (with flow rate data calculated from temperature and pressure readings by the flowCalc sub-VI) and the result is sent back to the ICM.

The data collected by dataAcq are as follows: one instance of DAQ Assistant collects twelve voltage signals from the valve control wires (connected to the DIO card), with DAQ Assistant set to take only one sample. A second instance collects six temperature signals from K-type thermocouples and six voltage signals from pressure transducers. Temperature and Pressure measurements were done with DAQ Assistant set to collect 7000 samples at 70,000Hz. This selection was made after much trial and error and proved to be the fastest setting that DAQ Assistant could handle, while providing the greatest reduction in noise. For the thermocouples, DAQ Assistant is configured to automatically convert the voltage signal into degrees Fahrenheit. To convert the pressure transducer signals into psig, the DAQ Assistant's calibration feature was used to collect calibration data. Readings from DAQ Assistant were translated from voltage into psig using this calibration data. The thermocouples were not calibrated due to time constraints and the fact that temperature data was not as critical as pressure data. Similarly, the voltage signals from the DIO card did not need calibration. Those signals were connected to a pull-up resistor on the relay panel, so it was expected that as long as the 5V power supply was active, the control signals would be around zero or five volts. Voltage data collected from the dataAcq sub-VI confirms this assumption.

After dataAcq executes and returns the array of data to the ICM, two blocks of code are executed. The first block handles the output of the data to a file. This feature is

controlled by the “Acquire Data” button on the Front Panel. When clicked, the program will run a check to see if an output data file exists. If it does, the new batch of data from dataAcq is formatted into a string and appended to that file. If a data file does not exist, the ICM creates it before writing the data. Clicking “Acquire Data” a second time closes the current data file and ends data reporting. Acquisition of new data continues as long as the ICM is running, but the results are only output to the Data Plot.

The second task the ICM does with the data from dataAcq is to check the water inlet pressure. After the SFS test bed was reassembled, it was found that certain hardware on the water inlet side of the test bed cannot withstand pressure greater than 100psi. To prevent this, the program checks to make sure that if the water inlet pressure ever reads higher than 75 psig, then the emergency stop button will be activated, as if the user had clicked it manually. Additionally, a second variable “Fill RV” is set so that when the program loops back to frame zero a special case is sent as the final valve command set: opening all of the Vent, Drain, and Fill valves. This code functions as a programmatic relief valve. If the pressure in the inlet line gets higher than it should, the program will allow the excess pressure to vent through the test bed until other corrective action can be taken.

### 3.2.3 Failure Simulation

The simulation of a valve failure comprises several program features in various parts of the code. The main features required for a failure simulation to work are failure settings setup, failure injection, failure detection, and automatic response. The operation of the SFS with a simulated failure was a driving force behind the development of the



ICM, and a main goal of the testing that took place in 2007 was to verify the fail-operational SFS concept.

In order to show that the SFS was fail-operational, the ICM needed to be able to simulate not only a valve failure, but the response to that failure as well. This means the program had to inject a failure, detect that failure, and respond to it. Failure injection means that a valve command is overwritten with some other command, so that even though the hardware is actually working, it behaves as if it not and has done something unexpected. The failure injection process is handled first by the failureSetup sub-VI, which lets the user specify what failure to inject. The actual failure injection takes place when the program loops back to the hardware control code. Detection of failures and the setup for the automatic response occurs after data acquisition.

The failureSetup sub-VI is called whenever the ICM is in the Idle State and the user clicks on the “Failure Sim?” button. When called, the VI opens its front panel, as shown in Figure 3.8, and waits until the user clicks the “Save” button. The controls that let the user select the failure type are on the left side of the window. When the user clicks “Save,” the failure data is sent back to the ICM.

To generate the failure settings from user input, the program interprets the data from the “Failure Type,” “Valve Type,” and “Cycle Stage” controls to generate three pieces of data for the user and the ICM. The first is the “Occurs At” array, the purpose of which is to tell the ICM what failure is happening, when it should happen, and what valve is affected.

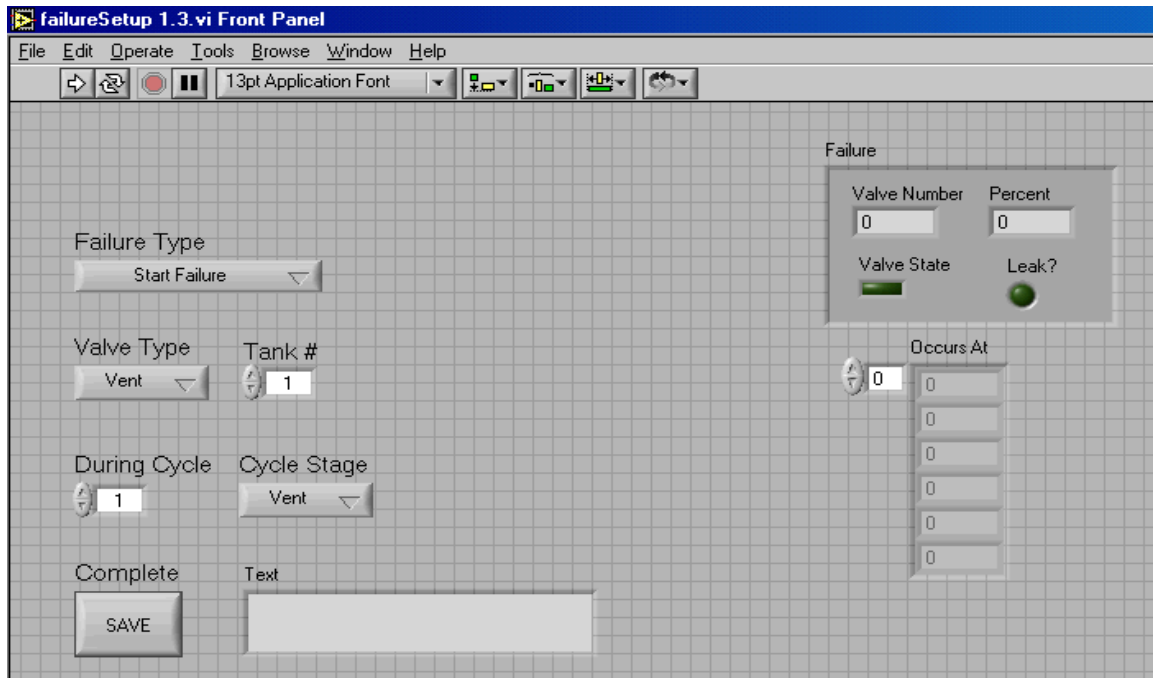


Figure 3.8 Sub-VI failureSetup Front Panel

The second piece of data generated by failureSetup is a text description of the failure. This is shown in the “Text” string indicator while failureSetup is running and in the “Failure Description” indicator when the data is passed back to the ICM. The purpose of this is to show the user in words what their settings mean, as well as let the ICM know that a failure injection has been configured. At some points in the program, the ICM checks “Failure Description” to see if it is an empty string (indicating no failure injection setting). Lastly, failureSetup generates the “Failure” cluster used in the failure injection step in order to tell the valve hardware (through the valveControl sub-VI) how to implement the failure when it comes time to inject it.

While the valve types and Cycle Stages have been defined thus far, the Failure Types considered by the ICM have not. During the design of the program, it was decided

to assume five failure modes that could happen to the valves during a Sequence. A Start failure is defined as a valve that fails to actuate at the start of a Stage (e.g., a Drain valve doesn't open at the beginning of a Drain Stage). Likewise a Stop failure is defined as a valve that fails to actuate at the end of a Stage. Power failures are cases of unexpected valve actuation in the middle of a Stage. For example, a Power On failure would be a valve that gets an erroneous power signal in the middle of a stage that causes the valve to actuate. To distinguish between Start/Stop failures and Power failures, the program sets the cutoff for plus or minus 0.25seconds from the start of a Stage. Any failure that happens within that window is considered a Start or Stop failure. This number was arrived at arbitrarily, but is intended to account for actuation response times of the valves. Lastly, a Leak failure is the case in which a valve does not close completely. Leak failures were not examined thoroughly because there was no way to actually simulate them in the program without additional hardware. References to Leak failures were kept in the code in case this feature could be implemented in the future.

It should be noted that not all possible settings on failureSetup will result in a real failure that can be injected in an Auto Sequence. For example, a Start failure of the Fill valve in Vent Stage can be set in failureSetup, but that failure is meaningless because the Fill valve doesn't do anything in the Vent Stage. By definition, a Start failure is one in which the valve fails to open at the start of a Stage. The Fill valve does not open in the Vent Stage, so no such failure can occur. The ICM will inject this kind of failure, but it won't be detected until the Fill Stage starts. This is because when the failure is injected, the Fill valve will be commanded to stay closed, which it was going to do anyway

throughout the Vent Stage. When the Fill Stage starts, the valve will still be commanded to be closed, but at this point, it should be open and the failure detection will kick in.

After the user has selected failure settings through `failureSetup` and started an Auto Sequence, the hardware control section of the code the program checks to see if the time to inject the failure has arrived (measured by the elapsed time from the start of a Sequence) and then `valveControl` takes the settings input by the user and overwrites the valve commands accordingly. At this point, the failure has been injected into the Sequence. The valve in question has been told to do something contrary to its expected behavior. The next step in the failure simulation is for the program to recognize, through examining the state of the valves, that a failure has occurred.

Sequence frame 2 (after data acquisition) contains the code for failure detection and setting up the automatic response to a failure. While the design of the program included multiple-tier failure detection, it was only feasible to implement the Basic tier (see Table 3.1). The specific code for comparing hardware settings with the expected settings is handled in the `basicDetect` sub-VI, which is only called when a Sequence is running. The output from `basicDetect` is a four digit Error Code. This code specifies the exact kind of failure that was detected so that the program can respond. Program conventions for error codes are shown below in Figure 3.9.

Error Code Conventions
4 digits
Thousands--Tank Number
Hundreds--Failure Type
Tens--Valve Type
Ones--Cycle Stage
Failure Types
0--Start
1--Stop
2--Power Off
3--Power On
4--Leak
Valve Types
0--Vent
1--Drain
2--Press
3--Fill
Cycle Stages
0--Vent
1--Drain
2--Press
3--Fill
4--Idle

Figure 3.9 ICM Error Code Conventions

Programmatically, all that failure detection required was a comparison of two Boolean arrays: one from the hardware and one from the expected settings. Detecting that there was a failure was the easy part, detecting the exact kind of failure was the more difficult part. The greater part of the implementation of the basicDetect sub-VI went into verifying that it would always detect the right kind of failure.

The program code for determining the Error Code is fairly complex, with multiple nested case and sequence structures to handle the various permutations. For brevity, the process is summarized in the flow chart shown below in Figure 3.10.

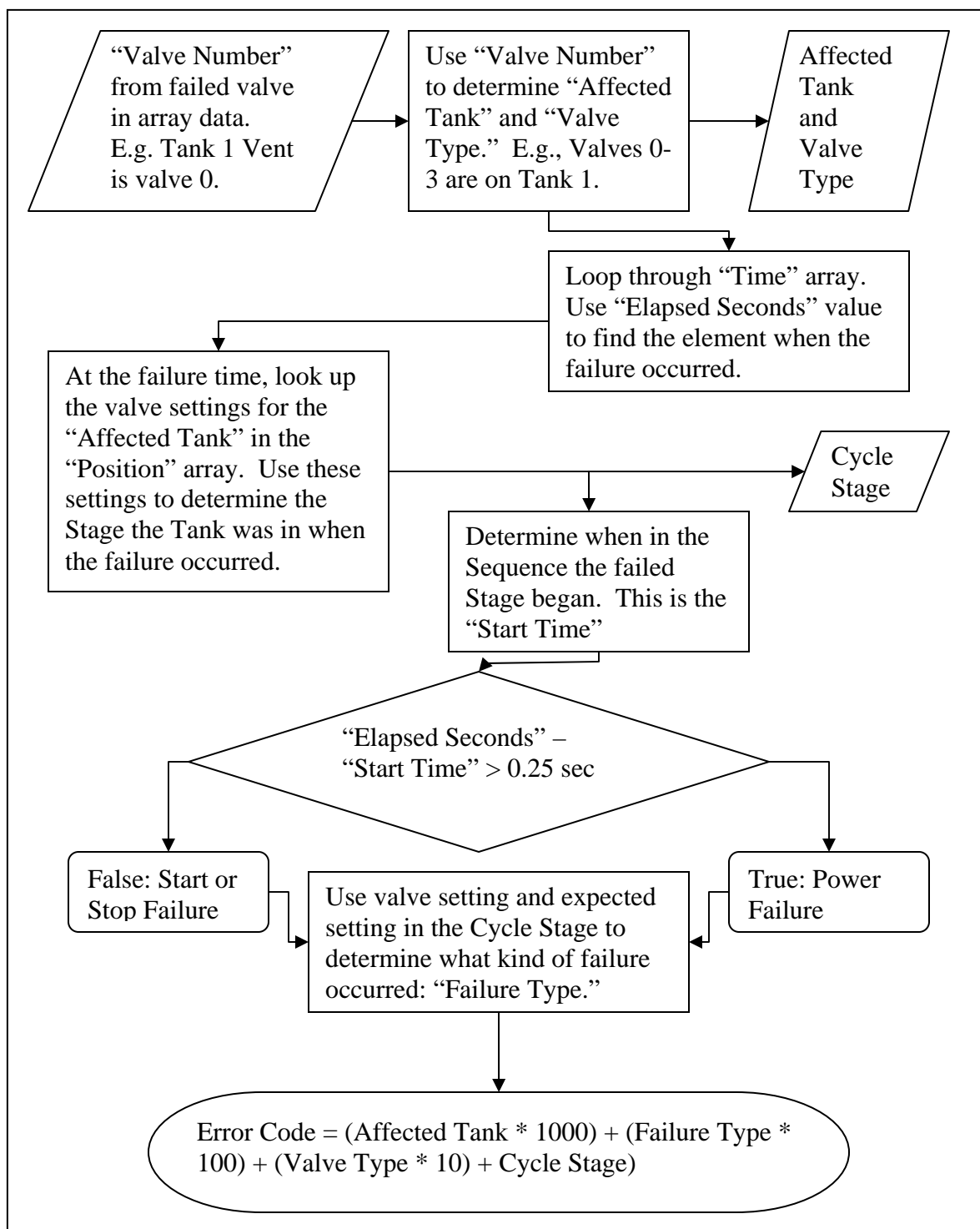


Figure 3.10 Sub-VI basicDetect Program Flow Chart

The testing of the basicDetect sub-VI was done by taking the part of the code that determines the Error Code and running it through a driver program. The driver supplied the abridged version of basicDetect with standard Time and Position arrays for a three-tank two-cycle Sequence, as well as a valve number and elapsed time value. The Sequence data remained constant, but the driver iterated through a number of valve number/elapsed time sets and recorded the Error Code that basicDetect sent back.

This approach was used so that the ICM did not have to be running for each test of basicDetect. The part of the basicDetect code that actually looked for a failure was left out of the test driver, and instead the driver told basicDetect what the valve number was and when it happened. With the complete sub-VI, the valve number would have been determined by checking the hardware settings, and the elapsed time is passed in as a parameter.

The test driver inputs covered all of the possible failure modes, except Power and Leak failures, that could occur in the second cycle of the Sequence. Power failures were omitted because the test program only planned for Start and Stop failures (in a real system, it is most likely that a valve failure would be a mechanical problem, which would show up when an attempt to actuate the valves is made). Leak failures were planned in the ICM, but could not be implemented, so no Leak failure tests were planned. The failure tests were done in the second cycle because the plan for the actual tests was to have some pre-failure time when data could be taken as the system functioned normally. Once this data was obtained, the failure could be injected, the automatic response could be initiated, and the system behavior pre and post failure could be observed.

Revisions to the basicDetect code were made when the Error Code output was something other than expected. When testing was complete, the Error Code generation part of basicDetect was reintegrated with the rest of the sub-VI and put back into use with the ICM.

The actual program response to a detected failure is done in the State functions part of the sequence structure (frame 3). When basicDetect returns with an Error code representing a failure (anything other than the default value of -1), the code on frame 2 of the sequence structure puts text in the “Warning” box on the Front Panel, and sets the “State” variable’s value to “Response” (see Section 3.2.5). The ICM is configured to automatically exit if more than one failure is detected in a single Sequence, as the SFS is only fail-operational for a single failure. It should be noted that error handling is not done in sequence frame 2. The failure detection part of the sequence frame only determines if a failure has occurred and what type of failure that is. This data is held on to until the program reaches the Response State, where the Sequence data is adjusted to respond to the failure (see Section 3.2.4).

The reason for this manner of implementing failure detection and handling was two-fold. First, separating the tasks of detecting failures and responding to them kept the code on each sequence frame simpler and more focused. Second, this design allows for easier integration of the multiple detection tier feature, should that be implemented in the future. With multiple detection tiers, it is possible that conflicting error detections could occur. A faulty data acquisition wire could send an erroneous signal to the Basic detection tier, while if Electronic or Actuation detection were implemented, the fault in Basic would not be noticed. This conflict would have to be resolved before corrective



action was taken. Encapsulating the failure detection on one frame of the sequence allows that to happen well before the program gets to the failure response code.

### 3.2.4 Auto Sequence Generation

The Auto Sequence feature takes the timing input data from the user and generates a valve control Sequence for normal test operation. The generation of Auto Sequence data is initiated at the start of an Auto Sequence and when the program responds to a detected failure. The code that handles Sequence generation is encapsulated in a sub-VI called fileGen. This sub-VI is intended for use only with the Auto Sequence, and always generates the timing and position data with the normal SFS Cycle Stage order enforced.

Because the order of valve positions is always the same (Drain, followed by Vent, followed by Fill, and so on), the implementation of this program centered on figuring out the time when a particular Tank Cycle began. During analysis of this problem, it was realized that the start time for any Cycle Stage could be related to the number of Drain Stages that had taken place. The start times for a typical SFS Cycle are shown below in Table 3.3. For this table and for program purposes, the Fill with Vent Closed Stage is simplified to the Fill-1 Stage, and Fill with Vent Open as Fill-2. The timing inputs from the ICM are abbreviated in the Start Times column as follows: D = Drain, O = Overlap, V = Vent, F = Fill, FV = Fill/Vent, and P = Pressurize.

Table 3.3 SFS Cycle Start Times

Start Time	Tank 1	Tank 2	Tank 3
Zero	Drain (Cycle 1)	Idle	Idle
D – O		Drain	
D			
D + V			
D + V + (F – FV)			
D + V + F			
D + V + F + P	Idle	Drain	Drain
2 (D – O)			
2D – O			
2D – O + V			
2D – O + V + (F – FV)			
2D – O +V + F			
2D – O + V + F + P	Idle	Drain	Drain
3 (D – O)			
3D – O			
3D – O + V			
3D – O + V + (F – FV)			
3D – O +V + F			
3D – O + V + F + P	Drain (Cycle 2)	Idle	Idle

In drawing this table a relationship between the number of Drains and Stage start times was realized. Defining  $N$  as the number of Drain stages that have occurred (beginning at one for the first Drain Stage of the Sequence), the equations in Table 3.4 can be developed.

Table 3.4 SFS Cycle Stage Start Time Equations

Start of Stage	Expression
Drain	$(N - 1)(D - O)$
Vent	$ND - (N - 1)O$
Fill 1	$ND - (N - 1)O + V$
Fill 2	$ND - (N - 1)O + V + (F - FV)$
Press	$ND - (N - 1)O + V + F$
Idle	$ND - (N - 1)O + V + F + P$

For the normal three-tank Sequence generation, fileGen simply implements the above equations. This is done in a sequence structure, in which the first frame handles the population of three sets of numeric/Boolean arrays, and the second frame merges those into one final set of arrays for output. Each tank has a numeric and Boolean array associate with it, which behave like the Time and Position arrays in the ICM.

For failure handling, fileGen modifies the procedure of three-tank Sequence generation. Sequences generated for failure response have to setup the two-tank operation in such a way that the system will continue to run for as long as it should have run if there was not a failure. In addition, whatever valve experienced the failure has to be commanded to stay in its failed state (open or closed) so that the failure detection code is tripped again. To determine how to start the two-tank mode, the program has to determine, based on the input Error Code, if the next tank needs to start its Drain Stage immediately, or if the failure is such that a tank's refill process can be halted and the tank skipped afterward.

In the former case, fileGen's "Drain Counter" and "Total Drains" variables are modified to represent the Sequence beginning again from  $N = 1$ . The Sequence will be offset by the elapsed time passed in when the sub-VI is called, so the new Sequence data

picks up where the old one left off. If no immediate drain is required, the program continues with N at its old value and there is no time offset.

Once the timing of the response Sequence is determined from N, the position data is generated. For two-tank mode fileGen does the following: determines the settings for the failed tank, sets the last tank to finish any refill process it may have started (so it's ready to go), and supplies the right valve position data to the two unfailed tanks.

The failed tank settings are based on what the failure type was and what valve was affected. For example, a Start failure type is one where a valve stayed closed, so the failure settings are to keep all valve closed. The timing data for this tank will be such that these settings stay the same as long as the Sequence runs. All told, the settings generated will keep the failed tank in the same state while the other two are running the rest of the Sequence.

With the time and positions settings done for the failed tank, the next part of fileGen handles the generation of the data for the other two tanks. This is done by alternating between the two remaining tanks, using the N value to determine timing. Lastly, the three array sets (one set per tank) are merged. In the case that the next Drain Stage needs to happen immediately, the program treats the Sequence as starting from  $N = 1$ . This keeps the calculations simple, but starts the Time arrays at zero, which would not work when passed back to the ICM. To correct this, a time offset (the "Elapsed Time" value) is added to the final Time Array. In the case where the Drain Stage did not need to happen immediately, this offset is zero because no recalculation of timing needed to be done, just modification to the remaining valve commands.

When the sequence structure is finished for the failure case, a new set of Sequence data (Time and Position) is generated for the required two-tank mode. This Sequence begins at the time when the failure occurred, so in normal operation, the data is passed back to the ICM and overwrites the old Sequence data. The settings on the failed tank are such that they won't trigger the failure detection again, and the rest of the Sequence plays out alternating between the remaining two tanks.

During operation of the ICM, the fileGen sub-VI is called without displaying the front panel to the user. The VI is also configured to run as a stand alone program. This feature was added to allow the user to generate Sequence data files that could later be run in the ICM as File Sequences. When fileGen is called as a sub-VI, the program will just return the Sequence data to the calling VI when it is done with the calculations. Running fileGen apart from the ICM allows the user to output Sequence data to a text file.

### 3.2.5 State Functions

The State functions part of the ICM design was intended to include the features that only execute after user input or at specific times in the program's operation. The three States that were implemented for the ICM are Idle, Sequence, and Response. State functions are handled in the final frame of the ICM's sequence structure. This frame consists of a case structure that is switched by the "State" variable. Each State has its own string associated with it, as described in the design (see Table 3.2). During implementation, it was decided to combine the "File" and "Auto" States into a single "Sequence" State. The reason for this is that both kinds of Sequences follow the same

basic procedure (see Figure 3.6). The only difference is in how the program variables are initialized during Sequence set up.

The default State is “Idle,” set by the program when it is first started. In this State, the program waits for user input to before activating any additional features. The code to access the failureSetup sub-VI is located in the Idle State, as is the code to set up and run File and Auto Sequences. When the user presses the “Begin Sequence” button, the “State” variable is set to “Sequence” for the next loop iteration.

If the “Sequence Type” is set to “File Sequence,” then the program will take the data in the “Filename” variable to specify which file to read data from, and that data is stored in the “Time Array” and “Valve Position Array” variables before the State ends. For a Auto Sequence, the fileGen sub-VI is called to generate the same time and valve position data before the State ends. After all of the Sequence set up code has been executed, the program will finish the loop and go back to frame 0 of the sequence structure (hardware control). When the loop arrives back at the State Functions, the new Sequence State will begin in place of the Idle State.

When a Sequence is started, the system clock’s millisecond timer value is stored in the “Start Time” variable. When in the Sequence State, each time through the program’s main while loop the “Elapsed Seconds” variable is calculated by comparing the current millisecond timer value to the value of “Start Time.” After this is done, the program looks up one of the elements in the “Time Array,” specified by the “Row Counter” variable (the data can be seen in Figure 3.3). If “Elapsed Seconds” is greater than or equal to the “Time Array” element at “Row Counter,” then the program finds the corresponding row in “Valve Position Array” and writes the valve command data to the

twelve valve indicators. On the next time through the loop, this new command data will be sent out as a command signal to the physical valves (see Section 3.2.1).

“Row Counter” is always initialized to zero at the start of either a File or Auto Sequence. This initialization is done in the Idle Stage. This means that the program will always look at the start of the “Time Array,” and move down from there. For example, the typical element in “Time Array” at position zero is also zero, corresponding to the valve commands that should be issued at the start of the Sequence when time equals zero. The very first row of “Valve Position Array,” row zero, corresponds to the time specified in element zero of “Time Array.”

The second function that is performed during the Sequence State is to check if the Sequence has ended. When new valve commands are used the value of “Row Counter” is incremented. If the new value of “Row Counter” is greater than the number of elements in “Time Array,” then this means that the program has gone through all the available timing and position data and the Sequence is at an end. The program then resets the State to Idle.

The last State used in the ICM is the Response State, where the automatic response to detected failures occurs. Unlike the Sequence State, the Response State is not set by direct user action, but by the program in frame 2 of the sequence structure if a failure is detected. This State is only implemented to work with Auto Sequences. If the program gets to the Response State in a File Sequence, then the program will simply shut down.

Failure response is done by taking the Error Code from the basicDetect sub-VI and running the fileGen sub-VI (see Section 3.2.4). When an Error Code is sent to

fileGen as a parameter, it will generate a new Sequence using the two remaining un-failed tanks. The specific Code allows fileGen to know where to pick up the Sequence. When fileGen returns the new Sequence data, the old data in “Time Array” and “Valve Position Array” is overwritten, the first row of the valve data is sent to the valve indicators, and the State is reset to Sequence. The program will carry on from there, running the new Sequence until completion or until it encounters a second failure.

### 3.2.6 Sub-VIs

Many ICM functions were incorporated as separate sub-VIs by design. This was to simplify the ICM, by grouping certain tasks in separate programs, as well as add flexibility to the program, by keeping hardware specific functions in easily replaced or modified program modules. A basic description of the sub-VIs that were written for use with the ICM is included in Table 3.5.

Table 3.5 Sub-VI Descriptions

<b><u>Sub-VI Name</u></b>	<b><u>Function</u></b>
basicDetect	Reads current settings on the DIO card and compares those settings to an array of expected values to determine if a failure has been injected into a Sequence.
chooseName	Generates data file names.
failureSetup	Opens user interface for selecting failure injection settings and sends necessary data back to ICM.
codeText	Translates a four-digit Error Code into a text description of the failure.
dataAcq	Acquires data from the temperature sensors, pressure sensors, and valve control signals.



Table 3.5 Continued

<b><u>Sub-VI Name</u></b>	<b><u>Function</u></b>
timeCalc	Calculates when a specified failure should occur.
fileGen	Generates Sequence data given Cycle Stage timing.
readFile	Reads Sequence data from a specified file and formats it into the time and valve position array formats.
valveControl	Writes Boolean data of valve indicators to DIO card to change valve control signals.
flowCalc	Calculates flow rate through a venture tube based on temperature, pressure, and tube geometry.
getValue	Returns the numeric value of one of the waveforms returned from the DAQ Assistant Express VI.
interpolateP	Applies calibration curves to voltage signals from the DAQ Assistant Express VI to get pressure in psig.
digits	Translates a multi-digit number into a numeric array in which each array element has one of the number's digits.
mergeArrays	Combines three time/position arrays for SFS tanks into one time/position array for all three tanks.
startTimes2	Determines the starting times of the various Cycle Stages given timing data and the number of tank drains that have occurred.
tankSwitch	Returns the next and last tanks in the SFS order (e.g. tank 3 after tank 2).

Within the ICM, sub-VIs are used as either a function VI, or a helper VI. A function VI is a program that does a specific, but complex task. The ICM itself is a function VI, as is fileGen. A helper VI is a simple piece of code that is used within a function VI to do a larger task. Often it was decided to make a helper VI when the same piece of code appeared several times in a function VI, and it was visually simpler to encapsulate that code within a sub-VI.

## CHAPTER 4

### EXPERIMENTAL WORK

After the completion of the proof-of-concept testing<sup>10</sup> at NASA MSFC, the SFS was transported to the University of Alabama in Huntsville's Propulsion Research Center (UAH PRC). The test bed was reassembled and adapted to work with the existing hardware at the PRC. New tests using the reconstructed test bed were conducted in 2007. The goal of this testing was to verify and quantify the fail-operational capability of the SFS. This chapter describes how the test bed was assembled at the PRC and how the testing was conducted.

#### 4.1 Experimental Goals

In order to evaluate the fail-operational capability of the SFS, the ICM was developed to the point where it could be used to run the SFS test bed as it was used in the proof-of-concept testing (the 2005 experiment), as well as inject a simulated failure and respond automatically with corrective action. The move to the PRC for the 2007 experiment was done because the test site at MSFC was no longer available. At the PRC, the test bed was reassembled and integrated with the air and water supply hardware that was located on site. With a functioning test bed and control system, the goals of the experimental testing were to 1) Calibrate system stage times with new hardware, 2) Run

baseline tests to replicate previous test results with the new hardware, and 3) Conduct fail-operational tests.

Calibration of Cycle Stage timing was done in the 2005 experiment<sup>10</sup> to determine how to construct the Sequence files for the proof-of-concept testing. For the 2007 experiment, Auto Sequences could be run through the ICM. In order to take advantage of this feature, the Stage timing needed to be well defined. An additional nuance in the current experiment, one that did not exist for the previous experiment, was that the ICM assumed that the Stage Timing would be the same for all three tanks. Due to project constraints, the test bed could not be engineered to be exactly symmetrical in all components, so some irregularities did exist and were expected. The calibration testing then served a two-fold purpose: characterize the specific Cycle Stage timing for all three tanks and provide a basis for Auto Sequence Stage timing that could be used for all three tanks at once, and still provide acceptable results.

In order to fully evaluate the quantitative effects of injected failures on the SFS, there needed to be some baseline to compare the fail-operational test data to. The second goal of the testing was to run a series of full-cycle tests without any simulated failures. These baseline tests would serve the purposes of verifying that the test bed was functioning as expected, provide replication of previous results, and create a standard data set for how the system behaves under normal operating conditions. The baseline tests consisted of two-parts: a two-cycle test and drain overlap calibrations. The two-cycle test was the basic Auto Sequence that used the Stage timing data from the calibration tests and ran for two cycles (for a total of six drains). At the end of the 2005 experiment, it was found that by overlapping the Drain Stages (starting the next tank's

Drain shortly before the end of the first tank's) a reduction in pressure fluctuations could be achieved. Given that the test bed was using different hardware in some cases for the 2007 test, it was decided to calibrate the length of time for an overlap through the baseline tests. When an overlap time was found that minimized pressure fluctuations, the data set from that test would then be used as the definitive baseline for future comparisons.

Lastly, the fail-operational tests were the ultimate goal of the 2007 experiment. The ICM was designed with five failure types in mind: Start, Stop, Power On, Power Off, and Leak (see Section 3.2.3). Leak failure injections were not implemented in the software for the 2007 experiment. While the ability to detect and respond to Power failures was implemented in the ICM, it was decided to keep the testing simple by only evaluating Start and Stop failures (a failure of the valve to actuate that beginning or end of a Cycle Stage, respectively). The rationale behind this was that the electrical system controlling the valves would be much more reliable than the valves themselves on a real system. Any valve failures would then most likely be mechanical in nature, and show up as a Start or Stop failure. With the scope of the fail-operational testing decided, the test program was then chosen to examine the effects of all the possible Start and Stop failures that could occur and influence the ability of the system to provide constant outflow. To standardize these tests, it was decided that the Auto Sequences would be initially set to run a two-cycle test (like the baseline test) and that failure injections would be done during the second cycle. This was to allow a full SFS Cycle to be failure-free, and thus the data from the first (normal-operation) part of the Cycle could be compared to the second (fail-operational) part.

## 4.2 Test Bed Configuration

While most of the SFS hardware was moved to the PRC, including the triangular frame, COPVs, and valves, significant alterations to the test bed had to be made in order to adapt it to the air and water supply systems at the PRC. A schematic of the test bed is shown below in Figure 4.1.

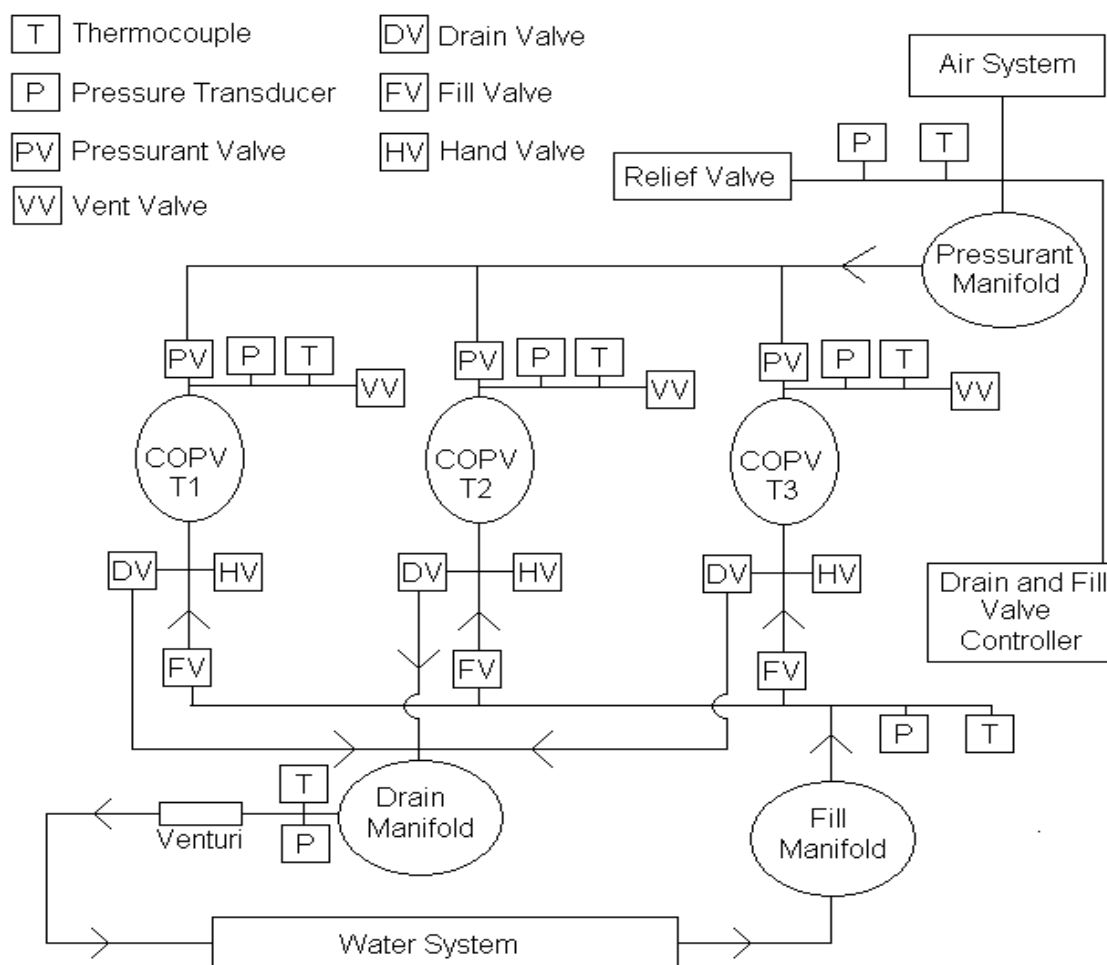


Figure 4.1 Test Bed Schematic

This section documents how the 2007 experiment's test bed was put together. For convenience the test bed is broken up into the following subsystems, described below in Table 4.1.

Table 4.1 Test Bed Subsystem Descriptions

<u>SFS Test Bed Subsystem</u>	<u>Description of Subsystem</u>
Run Tanks	Tanks, valves, manifolds, and associated tubing components that are not part of other subsystems.
Water	Components used to supply water flow to the SFS subsystem.
Air	Components used to supply pressurized air to the SFS subsystem.
Data Acquisition (DAQ)	Pressure transducers, thermocouples, associated wiring, and computer equipment used to monitor the temperature and pressure at various points on the SFS subsystem.
Control	Computer equipment (hardware and software) and associated wiring used to control the actuation of valves on the SFS subsystem.

#### 4.2.1 Test Bed Subsystems – Run Tanks

The Run Tank subsystem is the central part of the SFS. All of the other subsystems are used to supply the Run Tank subsystem with what it needs to operate. While most of this subsystem is a recreation of its counter-part from the 2005 experiment, certain changes were made in order to integrate the entire system with PRC infrastructure or to avoid problems encountered in the last experiment.

The Run Tank subsystem is built on a triangular frame to provide symmetry for the three SFS run tanks. Each tank is an 11-gallon COPV used in the 2005 experiment. Located in the center of the frame and supported by tubing connections to the tanks are

three custom-built manifolds. The manifolds are cylinders with two 1" ports on each end (one capped and one open) and three 1" ports 120° apart from each other on the outside. The manifolds are referred to by their function: the pressurization manifold is used for flow of pressurant gas into the three tanks, the fill manifold is for flow of water into the tanks, and the drain manifold is for flow of water out of the tanks. The COPVs mounted on the frame can be seen below in Figure 4.2, and the drain and fill manifolds are shown in Figure 4.3.



Figure 4.2 SFS Run Tanks and Frame

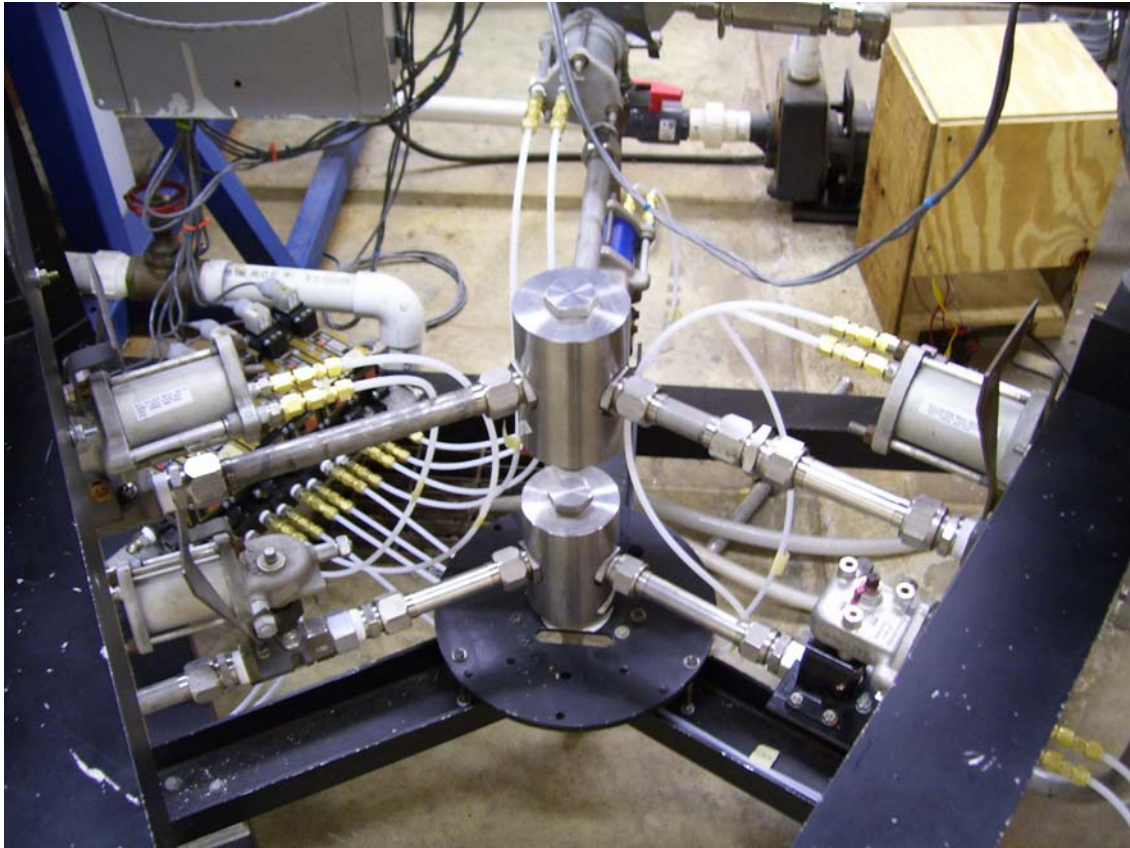


Figure 4.3 Drain and Fill Manifolds

Describing the construction of the Run Tank subsystem can be simplified because of the symmetry involved. All three tanks have the same kinds of hardware and tubing connections, so the entire subsystem is composed of the following description repeated three times around the central manifolds.

The central component on each section of the SFS frame is the COPV. Each COPV is equipped with two 1¼" ports, one on each side of the tank. Each COPV port is also fitted with a diffuser, which is used to disperse flow into the tank and keep the liquid



inside. The diffusers also reduce the tube size to 1", in order to interface with the rest of the tubing in the subsystem.

For convenience in description, the Run Tank subsystem can be broken up further into two other subsystems: the Gas Valve Assembly (GVA) and Liquid Valve Assembly (LVA). The GVA is everything above the top COPV port until the pressurization manifold, and the LVA is everything below the bottom COPV port until the fill and drain manifolds. The manifolds are not included in the GVAs and LVAs because they are shared among all three run tanks.

The standard GVA set up is pictured in Figure 4.4, and the description of the hardware is given in Table 4.2. It should be noted that there are two vent valves on each GVA, one with a 1" port and one with a 1/2" port. The 1" valve is pilot operated, while the 1/2" is not. Both valves are wired together with the same control signal. Because the 1/2" secondary valve is not pilot operated, it will open immediately when commanded to, and this will cause the pressure to change enough to open the primary valve as well. The secondary valve was added to the system in the 2005 experiment because it was found that without it, the primary valve would not open with the control signal, at least not in a repeatable fashion. The secondary valve ensures correct vent valve operation and repeatability.

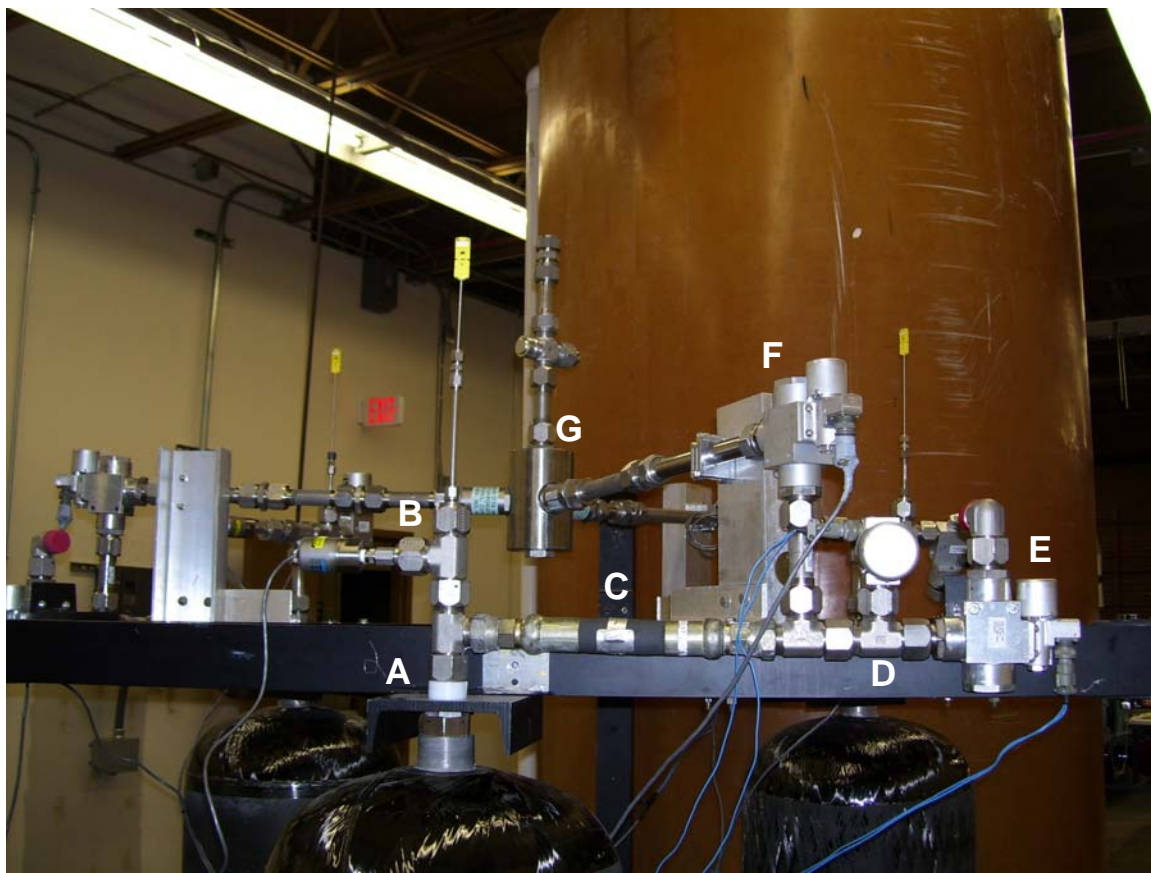


Figure 4.4 Gas Valve Assembly

Table 4.2 GVA Components

A	Top COPV port connected to a 1" tee.
B	Pressure and Temperature sensors (DAQ subsystem).
C	Flex Hose to account for volume changes in the tank under pressurized conditions.
D	Secondary Vent Valve, ½" Marotta Solenoid Operated Valve (SOV).
E	Primary Vent Valve, 1" Marotta SOV.
F	Pressurization Valve, 1" Marotta SOV, and Support Bracket.
G	Pressurization Manifold with Marotta check valves installed on all three outlet ports.

One important difference is the inclusion of check valves on the pressurization manifold's outlets. During the 2005 experiment, it was observed that when more than one pressurization valve was open, pressurant would tend to flow back into the manifold from one tank into another. In order to avoid this problem, the check valves were installed on the ports that connect to the GVAs.

The standard LVA is shown in Figures Figure 4.5. The descriptions of the hardware are given after each figure in Table 4.3. Both the drain and fill valves shown below are 1" Jamesbury ball valves. They are pneumatically actuated from a Ross valve controller. Brackets attached to the drain valves connect the LVAs to the frame and support drain valves and the drain manifolds. The plate attached to the bottom of the frame supports the fill manifold, and beneath that plate is a check valve which prevents water from flowing out of the SFS and back into the Water subsystem.

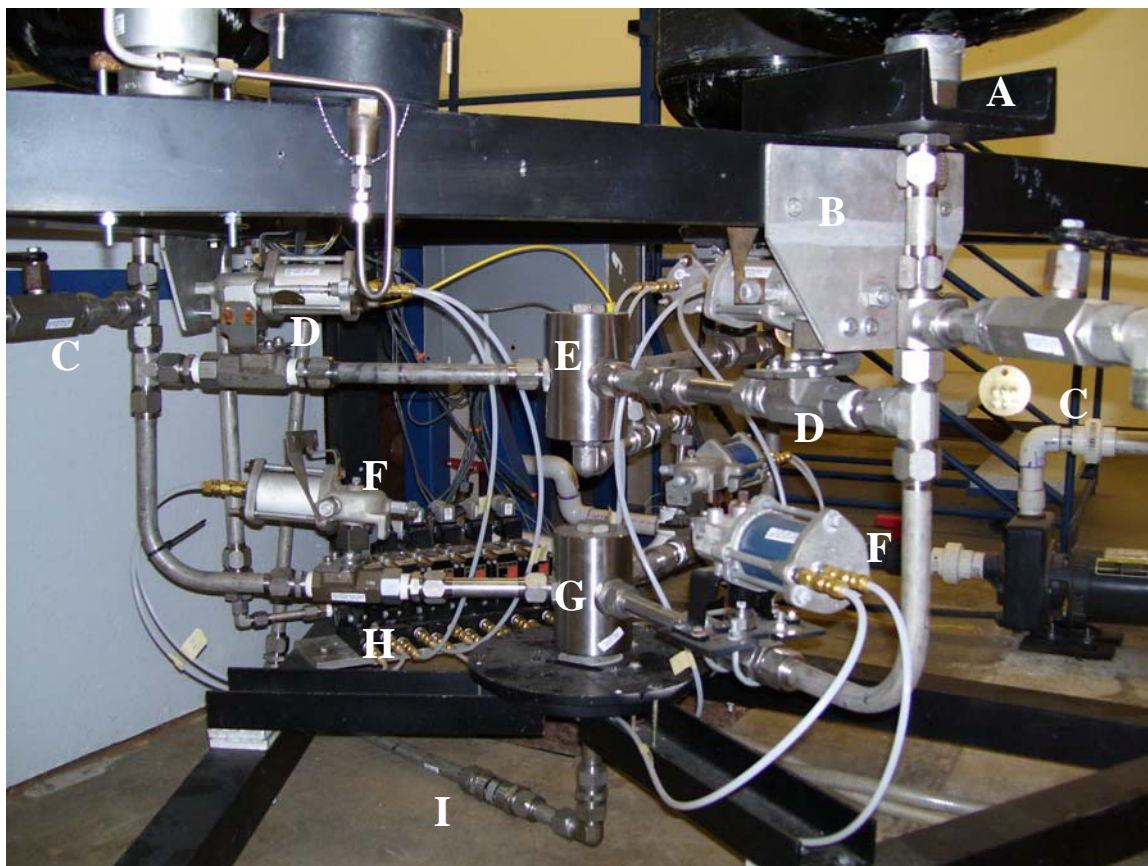


Figure 4.5 Liquid Valve Assembly

Table 4.3 LVA Components

A	Bottom COPV port.
B	Drain Valve Support Bracket.
C	Jamesbury 1" Manual Drain Valve
D	Drain Valves
E	Drain Manifold
F	Fill Valves
G	Fill Manifold
H	Ross Pneumatic Valve Controller (Control Subsystem)
I	Fill Line Check Valve

#### 4.2.2 Test Bed Subsystems – Water

While the 2005 experiment used a pressurized tank to supply water (as would be found on a real SFS-equipped rocket system), the 2007 experiment had to use a pump to move water from the supply tank (filled with normal tap water) into the SFS. The reason for this was that it was not feasible to use a pressurized tank in the JRC and it was far simpler to adapt the existing hardware for the SFS experiment. This is shown schematically below in Figure 4.6.

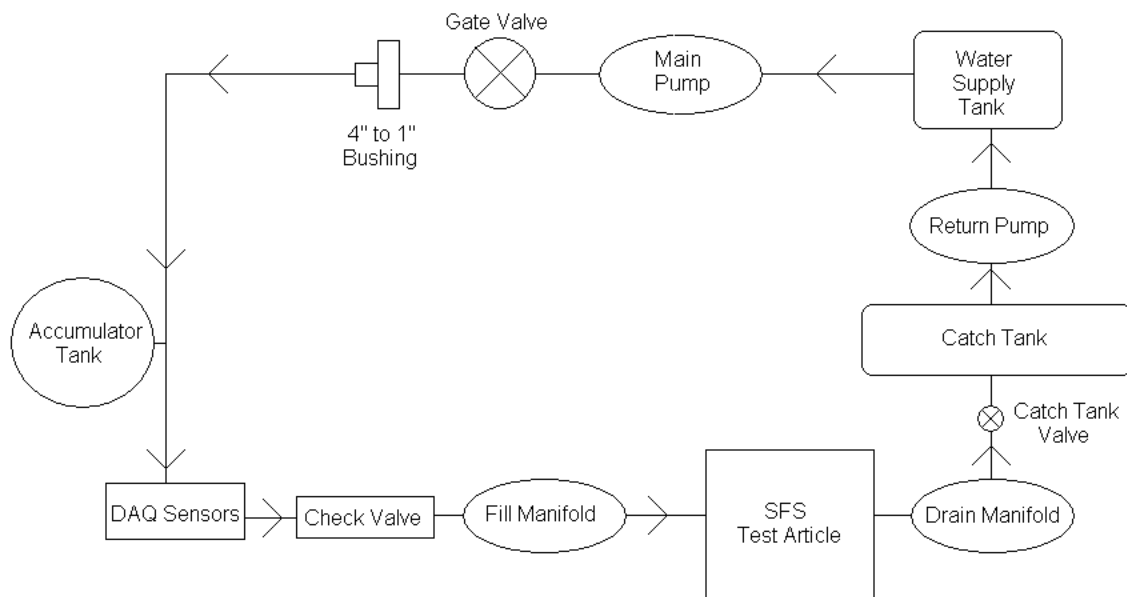


Figure 4.6 Water Subsystem Schematic

The JRC water system has two pumps: the main pump (Figure 4.7) was built by Taco and had a capacity of 1250 gallons per minute when operating at 1750 rpm. It was used to move water from the supply tank (large brown cylinder, below). The return

pump was a basic water pump and was used to move water from the catch tank (located underneath the platform to the right side of Figure 4.7) back into the supply tank.

Normally the water would go through the experiment, drain into the catch tank, and then be returned when the supply tank ran low. The return pump passes the water through a filter before it flows back into the supply tank.



Figure 4.7 Main Pump and Water Supply Tank

The main pump of the Water subsystem is a volumetric pump, and the pressure of the water is determined by downstream flow restrictions. Since the tubing geometry is fixed, the desired water pressure could be achieved by modulating the pump rate. The accumulator tank was added to even out any pressure variations that may exist from pump operation. After assembly, tests on the water system found that a reasonably consistent water inlet pressure of around 29 psig could be achieved when the pump was run at a setting of 50 Hz.

The outlet of the main pump is a 4" line that goes into a Plexiglas tank where another PRC experiment is housed. The SFS tubing was all 1", so at the flange on the entrance to the water table a reducer was installed to bring the tube size down to 1". This reducer consisted of two pipe bushings (4" to 2" and 2" to 1") and a pipe to Swagelok adapter. After the tube size was reduced, the water was routed to the accumulator tank and the fill manifold. The tube from the main pump outlet to the inlet of the fill manifold is called the fill line. This is shown below in Figure 4.8 and Figure 4.9, and the components are listed in Table 4.4 and Table 4.5.



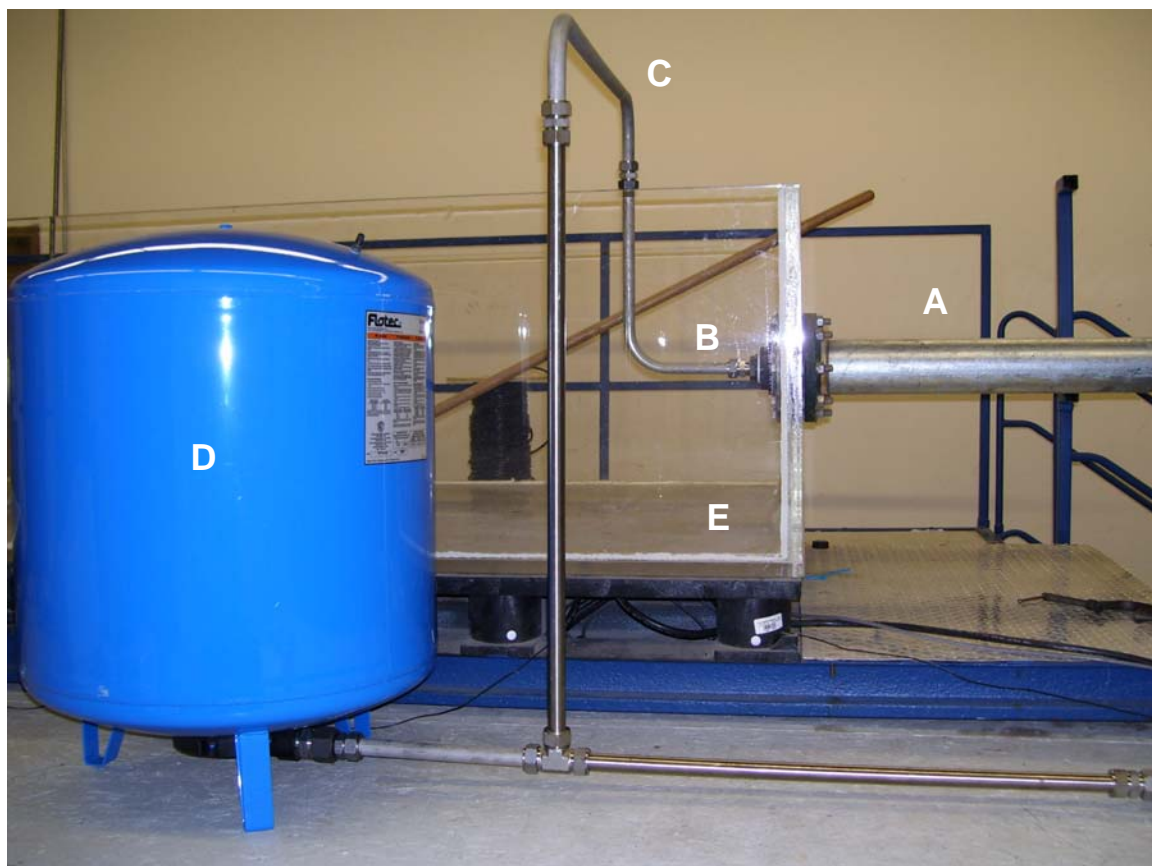


Figure 4.8 Pump Outlet and Accumulator Tank

Table 4.4 Water Subsystem Components

A	4" Main Pump Outlet Line.
B	4" to 1" Reducing Fittings.
C	1" Tube.
D	Accumulator Tank.
E	Water Table Experiment Tank (catch tank is located underneath this platform).



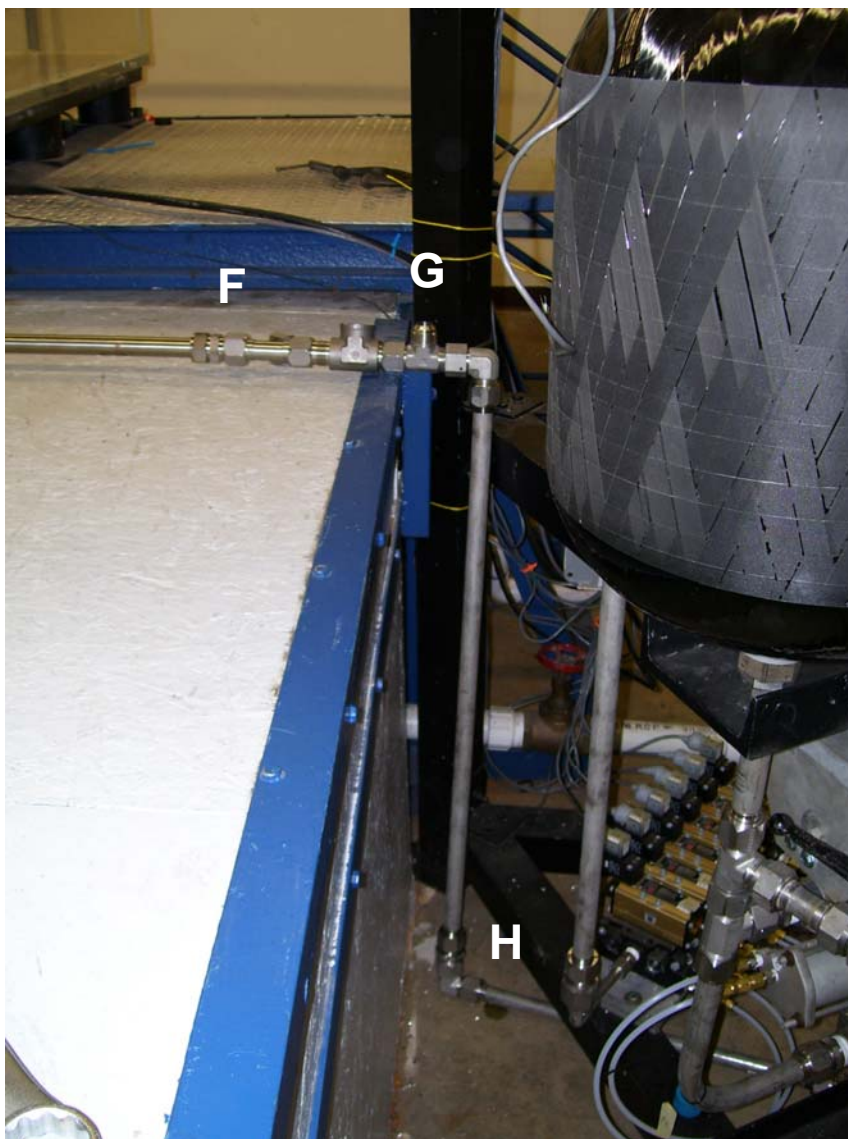


Figure 4.9 Fill Line into Run Tank Subsystem

Table 4.5 Water Subsystem Components Continued

F	Swagelok to AN adapter fitting.
G	Fittings for water inlet temperature and pressure sensors (DAQ Subsystem).
H	Line to Fill Manifold (Marotta check valve installed underneath the support plate below the manifold).

As shown in the schematic in Figure 4.6, the Water subsystem includes the hardware that gets the water out of the SFS, as well as the hardware that gets the water to the SFS. It was decided for the 2007 experiment to keep the water subsystem a closed loop, as was done for the original water table experiment. To do this, the drain line was installed from the drain manifold outlet to the manual drain valve of the catch tank. Under normal operations, the catch tank will be kept empty so this valve can be opened and allow the water to drain directly into the tank. Figure 4.10 and Figure 4.11 show the drain line connections, with the components described in Table 4.6 and Table 4.7. Lastly, the return pump and filter are shown in Figure 4.12.

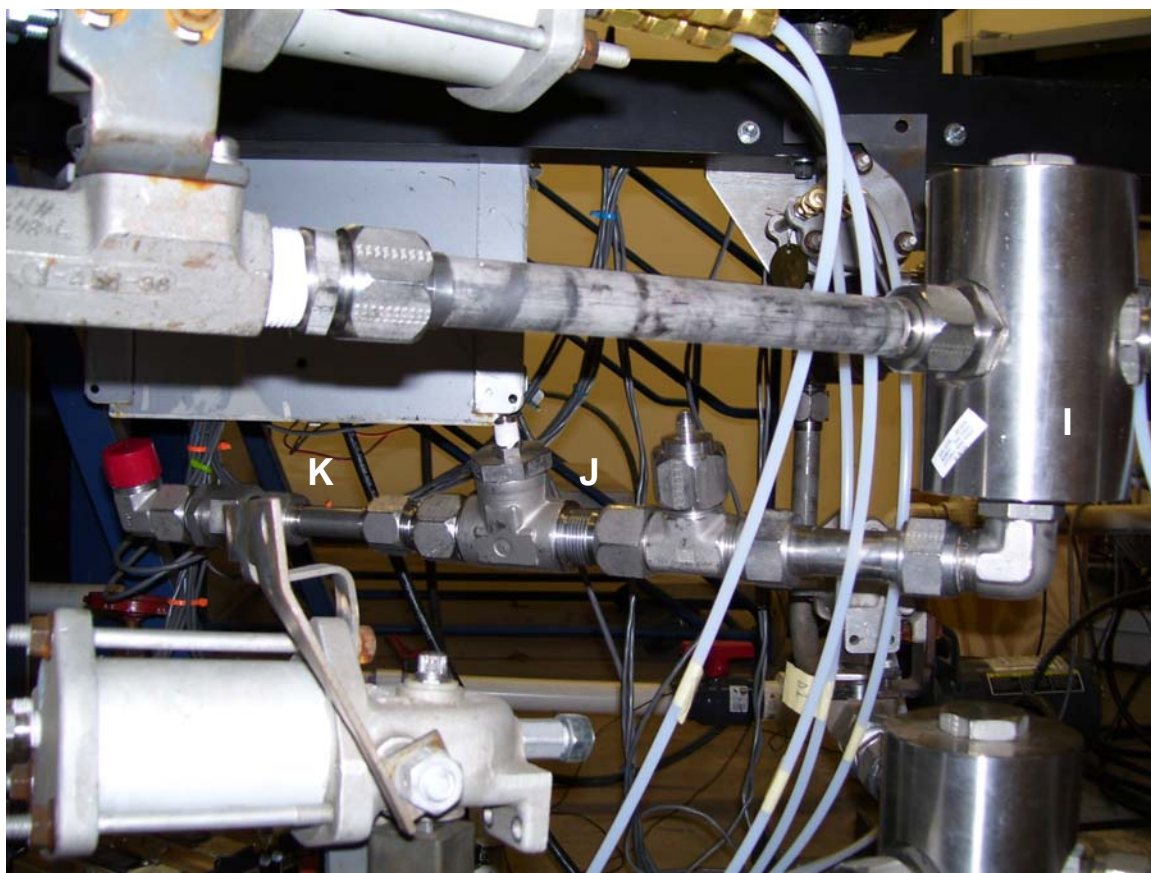


Figure 4.10 Run Tank Subsystem with Drain Line Installed

Table 4.6 Water Subsystem Components Continued

I	Drain Manifold.
J	Fittings for water outlet temperature and pressure sensors (DAQ subsystem).
K	Venturi Tube (inside AN fittings) for flow rate measurement (DAQ subsystem).

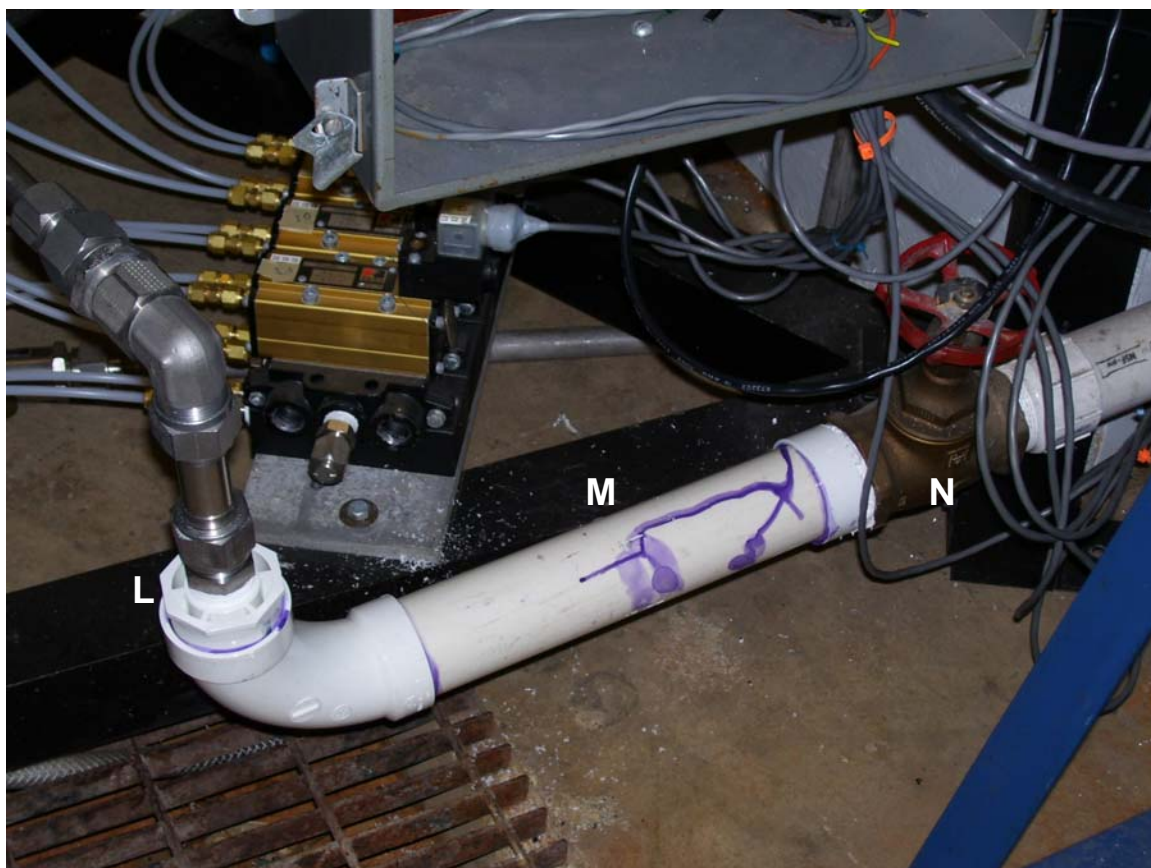


Figure 4.11 Drain Line to Catch Tank

Table 4.7 Water Subsystem Components Continued

L	Adapter fittings for 1" AN to 2" PCV.
M	PCV line to Catch Tank.
N	Catch Tank control valve (former manual drain valve).





Figure 4.12 Return Pump and Filter

#### 4.2.3 Test Bed Subsystems – Air

The Air subsystem has three purposes: 1) maintain constant air pressure at the pressurization manifold inlet, 2) provide necessary air flow to pressurize and drain the COPVs during SFS operation, and 3) supply necessary air flow to the valve controller to actuate the fill and drain valves. The air supply that was used for the 2007 experiment was the high pressure tank available at the PRC. Both the water table platform and the air tank were fixed in place. It was decided to position the SFS frame closer to the water supply and run a pressurant line from the air tank on the other side of the room to the test bed. This is shown schematically in Figure 4.13.

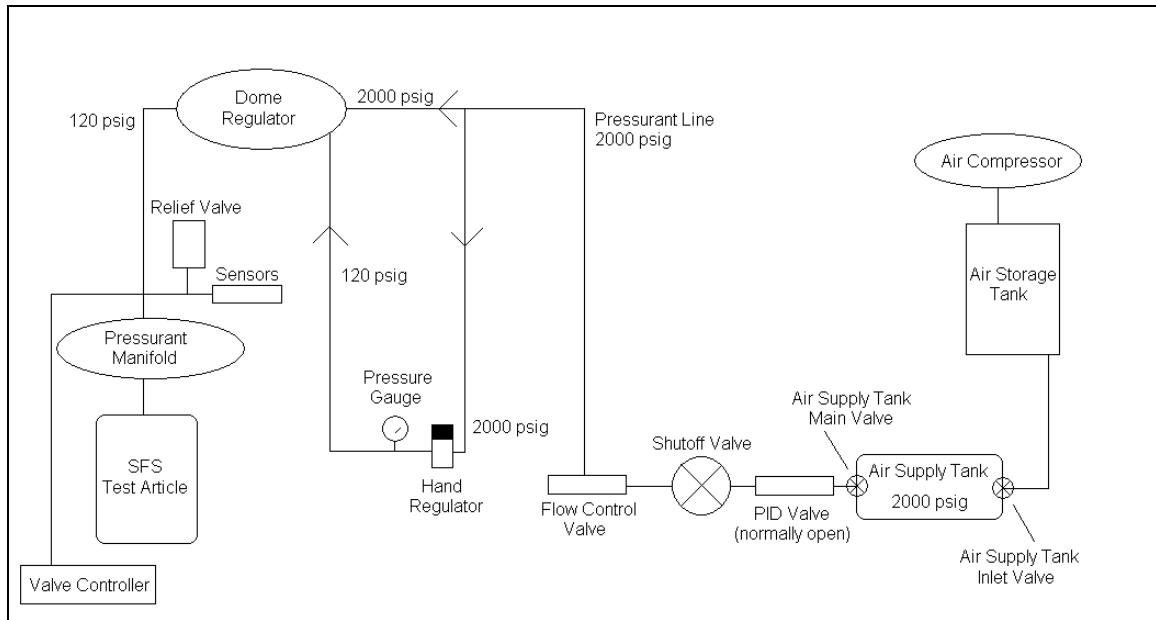


Figure 4.13 Air Subsystem Schematic

The main air supply tank is typically set around 2000psi. Part of the loan agreement that was made in order to use the COPVs for the 2007 testing was that operational pressure was not to exceed 120psi. A dome regulator, shown in Figure 4.14, was installed at the end of the pressurant line near the test bed to reduce the pressure from 2000 to 120 psig.

This regulator used for the 2007 experiment was a Grove dome regulator. Upstream of the regulator inlet a tee was installed to divert some of the 2000 psig air into a Grove hand regulator mounted on the SFS frame. This regulator reduced the pressure to 120 psig, and then was routed back to load the dome regulator. The larger dome regulator was used to regulate the air flow into the pressurant manifold because it had a greater capacity than the hand regulator and would provide a faster response rate.

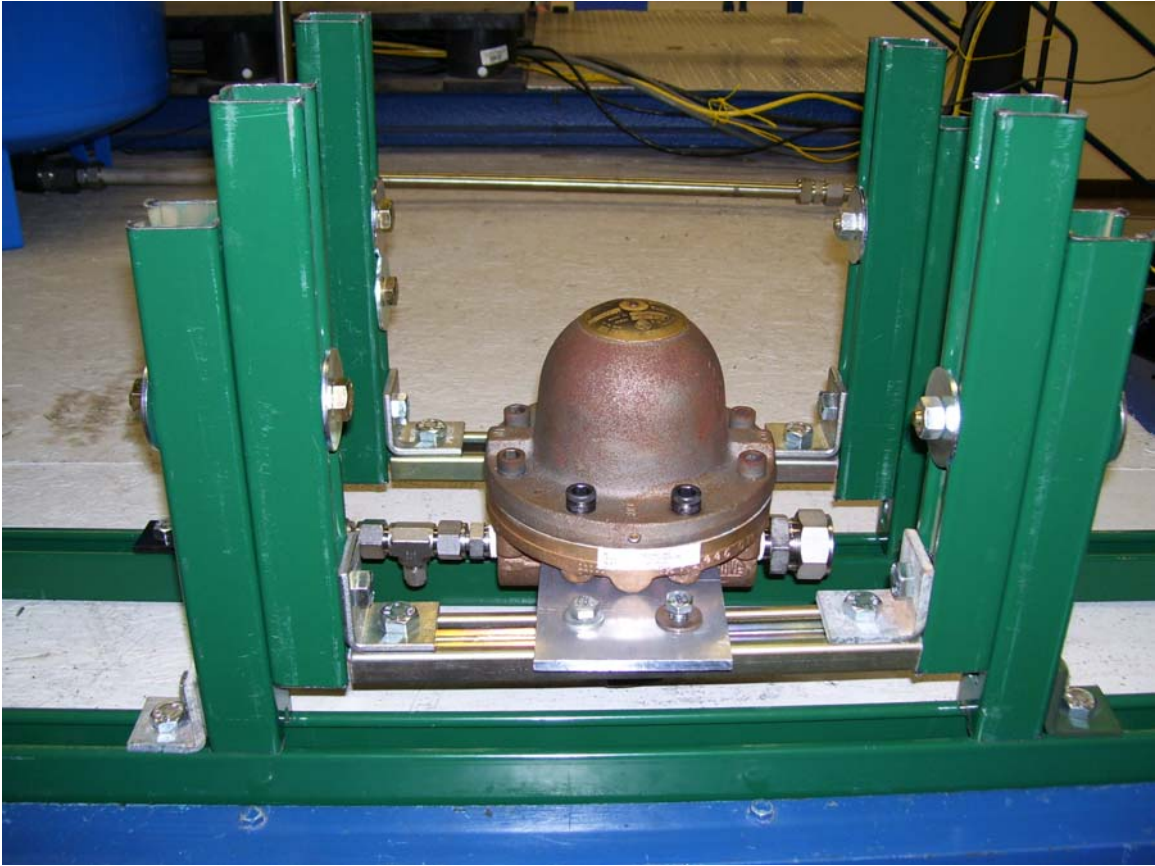


Figure 4.14 Dome Regulator Mounted on Support Bracket

A pressure gauge was installed on the SFS frame and connected to the air inlet. Adjusting the hand regulator would change the amount of pressure allowed through the dome regulator, which would show up on both the pressure gauge and the ICM read-out. Testing of the air system was done so that a hand regulator setting that gave the appropriate air inlet pressure could be found. The regulator and the gauge are shown in Figure 4.15.

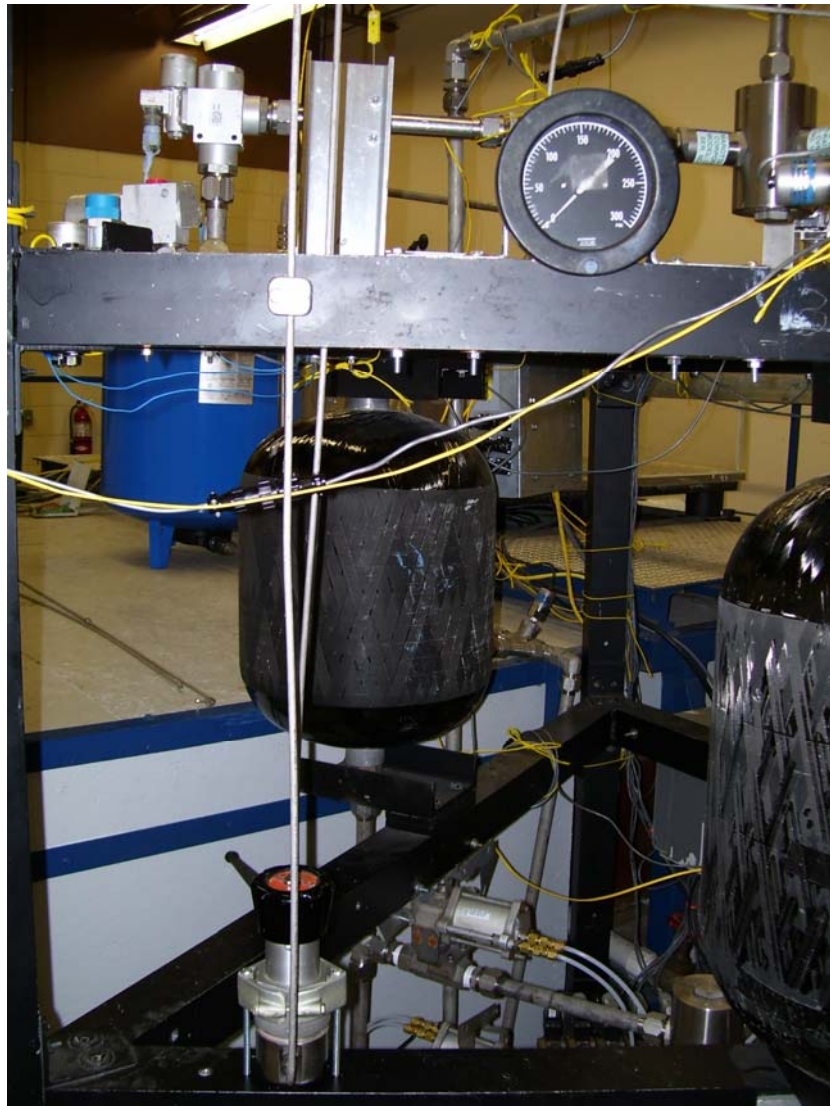


Figure 4.15 Hand Regulator Mounted on SFS Frame

In order to ensure the safety of the system, a relief valve was installed downstream of the regulator and upstream of the pressurant manifold, shown below in Figure 4.16. The set pressure of this relief valve was 150psi, and the flow capacity was 440scfm. To make sure that the flow through the pressurant line never exceeded this capacity, a Circle Seal needle valve was installed on the pressurant line on the side closer



to the supply tank. This valve had a Cv of 0.34 and was left open during normal operation. As such, the valve acted effectively as an orifice, constricting the flow rate to what the relief valve could handle if anything were to go wrong.



Figure 4.16 Relief Valve Installed above Pressurization Manifold

The last requirement for the air subsystem was to provide airflow necessary to operate the fill and drain valves. It was decided to obtain the air pressure needed by connecting the valve controller to the inlet pressurant line. In the 2005 experiment, the valve controller was operated with 150psig air from a separate regulator, but testing

during construction of the 2007 experiment showed that 120psig air bled from the inlet line would work as well. More information on the valve controller is found in Section 4.2.5.

#### 4.2.4 Test Bed Subsystems – Data Acquisition

The Data Acquisition (DAQ) subsystem was significantly altered from the 2005 configuration. Both the hardware and the software were altered from the original experiment. Table 4.8 below summarizes the key differences in the DAQ subsystems for the 2005 and 2007 experiments.

Table 4.8 DAQ Changes

<b><u>2005 Experiment</u></b>	<b><u>2007 Experiment</u></b>
5 Pressure Transducers and 3 Thermocouples.	6 Pressure Transducers and 6 Thermocouples
LabVIEW DAQ program run from a separate computer from Control subsystem.	Single LabVIEW program operates the DAQ and Control subsystems
Flow rate calculated using orifice.	Flow rate calculated using venturi tube.
Assumed sample rate.	Sample rate measured by the program.

In the 2007 experiment, it was decided to be as complete as possible with regards to the data. Pressure and Temperature measurements are made at the following locations:

1) all COPVs (sensors located in the GVAs), 2) water inlet (sensors located after the accumulator tank), 3) water outlet (sensors located after the drain manifold outlet), and

4) air inlet (sensors located upstream of pressurization manifold inlet). Previously temperature measurements were only made on one of the tanks, the water inlet, and the water outlet.

For all temperature measurements 1/8" Omega K-type thermocouples were used. Viatran model 240R35 pressure transducers (0-200psig range, 0-5V output) were used for all pressure measurements except water inlet. For that measurement a Viatran model with a 0-100psig range was used. The different model transducer was because the water accumulator tank was not rated above 100 psig, and the pump could not create a pressure condition that high. All pressure transducers required an excitation voltage of 28V, and were run off a single power supply. The DAQ sensors for one of the GVAs are shown below in Figure 4.17, and described in Table 4.9.

The signal wires from the DAQ sensors are routed to a breakout box mounted on the SFS frame. From there the signals are then wired into a National Instruments DAQ box, model SCB-100. The SCB-100 is a 100 pin Input/Output device that was used to read in the analog signals from the pressure transducers, thermocouples, and the valve control voltage signals from the Control subsystem.

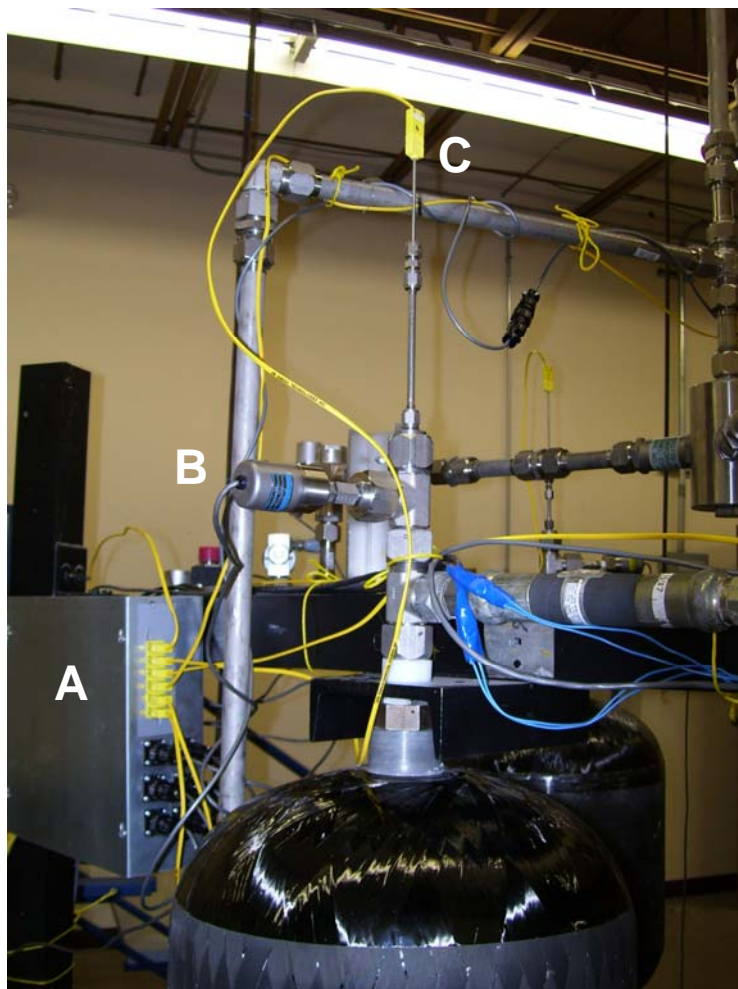


Figure 4.17 Gas Valve Assembly with DAQ Hardware Installed

Table 4.9 DAQ Subsystem Components

A	DAQ Breakout Box.
B	Viatran Pressure Transducer.
C	K-Type Thermocouple.

The software aspect of the DAQ subsystem was handled by the ICM program. The ICM read in the signals from the SCB-100 for processing and recording. Conversion of voltage signals into normal units (psi and degrees Fahrenheit) was done within the

software (see Section 3.2.2). The SCB-100 has a built in temperature sensor for use with thermocouples, and the DAQ Assistant Express VI could be configured to handle K-type thermocouples automatically. Prior to testing, the pressure transducers were calibrated using an oil-based dead weight tester and the DAQ Assistant's calibration feature. The resulting calibration curves were included in the software to change the transducer signals to psi. See Chapter 3 for more details on ICM operation.

#### 4.2.5 Test Bed Subsystems – Control

The purpose of the Control subsystem is to command the valves to actuate in the proper sequence for SFS operation. This subsystem acts as the bridge between the computer software, where the valve commands originate, and the valves themselves. In both the 2005 and 2007 experiments the same 12 valves types were used: each COPV is equipped with a Vent, Pressurize, Fill and Drain valve. As mentioned in the Run Tank subsystem section, the Vent valve is composed of two actual valves. Since these valves are wired to the same control signal, for Control subsystem purposes (as well as software programming) they are considered one valve.

All of the valves used on the SFS test bed are controlled by sending an electronic signal to actuate a solenoid. In the case of the Vent and Pressurization valves, the solenoid is part of the valve itself, and in the case of the Fill and Drain valve, the electronic signal is sent to the valve controller, and the solenoid there provides a pneumatic action to actuate the valves. The valve controller is shown below in Figure 4.18.

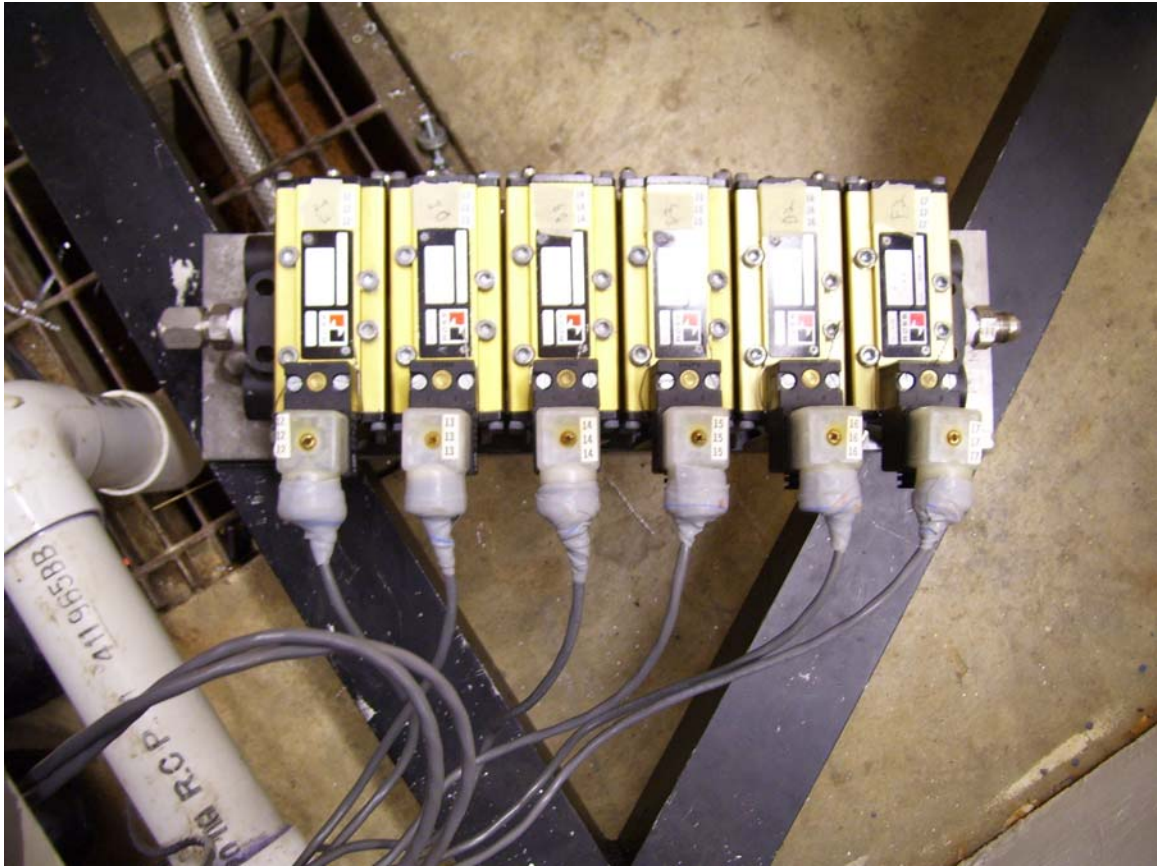


Figure 4.18 Ross Valve Controller

The Pressurize, Fill, and Drain valves are all configured to be closed when the solenoid is in the de-energized state. The vent valves are configured to be open when de-energized. This configuration is inherited from the 2005 experiment, and was implemented as a safety check. Should power ever be lost suddenly, no pressure will be allowed to build up in the tanks.

The valve command signals originate from and are controlled through the ICM. Though the software is different from that used in the 2005 experiment, the physical means that the computer program uses to control the valves are the same. When the ICM



sets valve positions the program writes data to twelve bits on a digital input/output (DIO) card. The output pins of the DIO card are then wired to a relay panel. The valves require a 28V input to actuate, more than the computer can supply, so relays are used with a separate power supply to translate the 0-5V computer signal into a 0-28V signal. When the bit on the DIO card is low (0V), the relay is open and 28V is seen by the valve. This is indicated by a lit LED on the relay module, and means the valve is energized. In order to make sure that the signal received at the relay panel's control strip (bottom set of screws in Figure 4.19 below) is always 0 or 5 volts the relay panel has a pull-up resistor for each relay module and requires a 5V power supply to the panel.

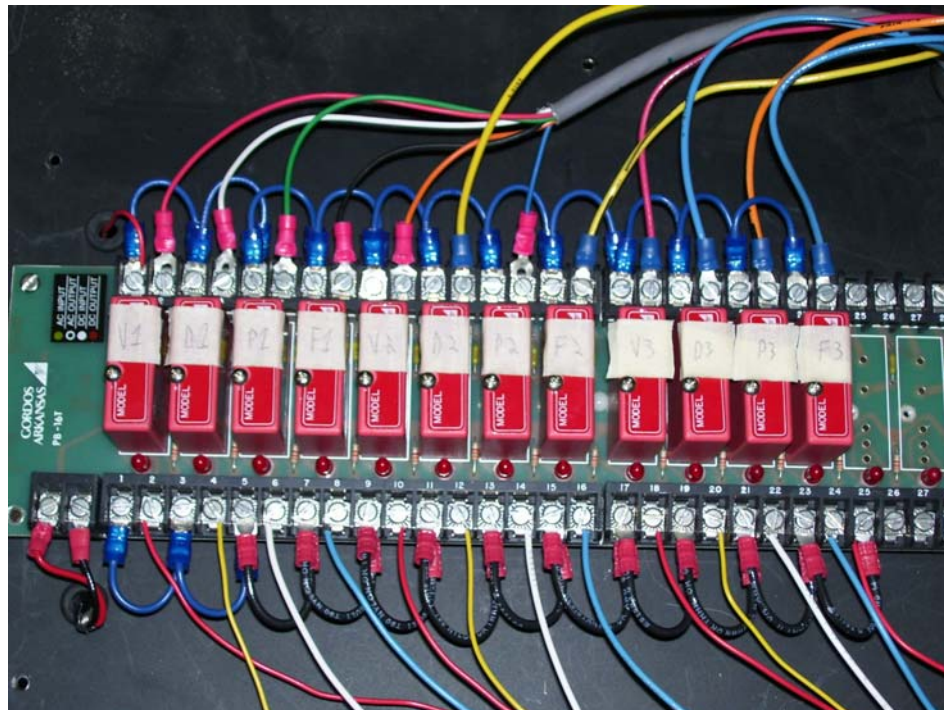


Figure 4.19 Control Subsystem Relay Panel

The field strip of the relay panel (top set of screws in Figure 4.19) is wired to the 28V power supply and to a breakout box mounted on the SFS frame. From the breakout box the signal wires are attached to the individual valves. When the relay is closed, the circuit between power supply and valve is complete and the valve gets 28V.

#### 4.3 Test Program

The goal of the 2007 test program was to experimentally evaluate the fail-operational capability of the SFS using the SFS test bed and the ICM. After the test bed was reconstructed at the PRC a Test Readiness Review was conducted in order to identify any problems and confirm that the test bed was ready. Once the test bed had been modified per the suggestions brought up during the review and tested for functionality, the formal fail-operational test program began. The 2007 experimental testing followed the test matrix shown in Table 4.10.

There were three parts to the test program, as shown in the test matrix. First, calibration testing was done to determine the timing inputs needed for the ICM Auto Sequence feature. Next, Baseline tests were run to evaluate the operation of the test bed when no failures were injected, and what effect overlapping Drain Stages has on outflow pressure. Lastly, the Fail-Operational tests were run to evaluate the effect specific valve failures had on outflow pressure. More details on the procedures used in these tests are given in Appendix B.



Table 4.10 2007 Experiment Test Matrix

<b><u>Step</u></b>	<b><u>Process</u></b>	<b><u>Desired Result</u></b>
1. Fill Stage Calibration	Fill tanks for prescribed periods of time. Manually drain/capture/record water volume.	Characterization of the fill rate of each of the COPVs.
2. Vent Stage Calibration	Pressurize empty COPV, close pressurization valves, open vent valves. Obtain pressure versus time data from the data file.	Characterization of the vent rate of each of the COPVs.
3. Pressurization Stage Calibration	Fill empty COPV with water and pressurize for sixty seconds. Obtain pressure versus time data from the data file.	Characterization of the pressurization rate of each of the COPVs.
4. Drain Stage Calibration	Fill tanks to desired volume. Pressurize and drain for prescribed periods of time. Vent. Manually drain/capture/record water volume.	Characterization of the drain rate of each of the COPVs.
5. Baseline Tests	Use data from calibration tests to run two-cycle sequences. Vary time that Drain Stages are overlapped.	Determine drain overlap time that gives most steady outflow, obtain standard no-failure data file
6. Fail-Operational Tests	Use data from calibration and baseline tests to run two-cycle sequences with injected failures.	Steady, pressurized outflow during drain transition, failure injection, and automated response.

All of the tests run during the test program were operated by the ICM.

Calibration tests were done by running File Sequences so that the system only ran the Tank Stages necessary for the specific calibration test. Baseline tests used the Auto Sequence feature, and the Fail-Operational tests were Auto Sequences that used the failure injection feature.

Before the start of the fail-operational testing, three classifications were developed to describe the results of any fail-operational test. A “Success” was defined as a test in which the system injected the failure correctly, responded to the failure correctly, and the outflow pressure did not fluctuate more than the parameters established by the baseline tests. An “Error” was defined as a test in which the system injected failure correctly, responded to the failure correctly, but the outflow pressure fluctuated more than expected. Lastly, an “Incomplete” test was one in which the system either did not inject the failure correctly or did not respond to the failure correctly. These classifications were developed to differentiate between injected failures that had little or no effect on outflow pressure (Success) and ones that did (Error), as well as failures that were not handled correctly due to hardware malfunction or software error. The specific types of fail-operational tests that were conducted are summarized below in Table 4.11.

Table 4.11 Fail-Operational Test Matrix

<u>Failure Type</u>	<u>Valve</u>	<u>Tank #</u>	<u>Cycle Stage</u>
Start	Vent	1	Vent
Start	Drain	1	Drain
Start	Press	1	Drain
Start	Press	1	Press
Start	Fill	1	Fill
Start	Vent	1	Fill
Stop	Vent	1	Vent
Stop	Drain	1	Drain
Stop	Press	1	Drain
Stop	Press	1	Press
Stop	Fill	1	Fill
Stop	Vent	1	Fill
Start	Vent	2	Vent
Start	Drain	2	Drain

Table 4.11 Continued

Start	Press	2	Drain
Start	Press	2	Press
Start	Fill	2	Fill
Start	Vent	2	Fill
Stop	Vent	2	Vent
Stop	Drain	2	Drain
Stop	Press	2	Drain
Stop	Press	2	Press
Stop	Fill	2	Fill
Stop	Vent	2	Fill
Start	Vent	3	Vent
Start	Drain	3	Drain
Start	Press	3	Drain

## CHAPTER 5

### EXPERIMENTAL RESULTS

One of the major benefits of the Sequential Feed System is that the concept is intrinsically fail-operational. The results of the SFS test program and the 2007 experiment showed that steady ( $<5\%$  pressure variation) outflow could be achieved under normal SFS operation, and that most of the failure modes investigated had little or no effect on outflow pressure or flow rate. The testing was conducted over several weeks, with each test building logically to the next. Calibration tests were done to determine the timing settings to be used for the Auto Sequence, Baseline tests used those timing inputs to verify ICM operation, and Fail-Operational tests modified the Baseline feature to test the response of the ICM and the behavior of the SFS under abnormal conditions.

#### 5.1 Calibration Tests

While in much of the computer programming and analysis it was natural to consider the Drain Stage as the start of a Tank Cycle, the Calibration testing began with calibration of the Fill Stage. This was for the pragmatic reason that water could not be drained out of the COPVs until it was put there. The Fill calibrations were modified from the 2005 experiment procedure in that there were broken into two parts: a fill calibration with the vent closed (Fill-1) and a fill calibration with vent open (Fill-2).

It was quickly discovered during Fill-1 calibrations that if the vent was kept closed through the entire test, the tanks would fill up to a certain point and no farther. This was because the water would compress the remaining air in the tanks until the pressure would effectively counteract the water pump. After this was realized, the Fill-2 calibrations were begun. A trial-and-error approach was used until timing for the Fill and Fill/Vent inputs was arrived at, 25 and 15 seconds, respectively. A Fill-2 calibration test run with these inputs confirmed that the water in the tanks was acceptable (approximately 10.1 to 10.5 gallons) for the other testing.

The Pressurization calibration had to be modified from the original plan because of a pressure effect found during check-out testing of the SFS test bed. It was observed that when a tank was pressurized, it would reach equilibrium with the air inlet, but when the pressurization valve was closed, the pressure in the tanks would drop slightly and the decay in pressure was exponential. This effect was not seen when the valve was kept open for a while and then closed. It was determined that this was a thermal effect. When the valve was only open for a short time, the gas in the tank couldn't reach thermal equilibrium before the external pressure source was removed. To counter this effect, the Pressurization calibration was expanded to sixty seconds in duration. The data showed that the tank pressure reached the desired level very quickly (less than one second), but it was decided that to avoid the thermal effect the Pressurization Stage time would be extended as far as possible. This would minimize the Idle Stage, but that was deemed unimportant. The goal in extending the Pressurization Stage was to keep the tanks as close to the desired pressure level as possible so that when the Pressurization valve was

reopened for the Drain Stage the tank would be ready for operation. After the other calibration tests were done, the Press timing input was selected at 7.5 seconds.

Vent Stage calibrations were done using an empty tank pressurized for sixty seconds (to avoid the aforementioned thermal effects) and then vented to atmospheric pressure. The data matched the expectation that venting all the way to zero psig would take too long, so a compromise timing valve was reached. The Vent Calibration data showed that the time for the pressure in the tanks would go below 1 psig in less than 22 seconds. This was the value selected for the Vent timing input as such a small amount of pressure left in the tanks would not adversely affect the Fill stage.

The initial construction of the SFS test bed for the 2007 experiment used a 0.28 inch diameter venturi tube on the drain line. This component was selected months in advance of the testing and without much forethought. When it came time to run the Drain Calibration tests, it was found that with this venturi in place, the tanks would almost completely drain in 25 seconds. This was an unacceptable Drain Stage time as there was no way to hurry the Fill Stage. In order for the fail-operational mode to work, the entire refill process (Vent, Fill-1, Fill-2, and Pressurization) had to be completed in less time than the Drain Stage.

To correct this, a new venturi tube with a diameter of 0.186 inches was found and installed. The Drain calibrations were run again and found that the tanks could be successfully drained in 55 seconds. There was a fairly significant discrepancy in the behavior of the tanks found in this test. Tank 1 would have about 1.5 gallons of water left in it after a 55 second drain, while Tanks 2 and 3 would have less than 0.25 gallons. It was decided that this ultimately would not be a problem. All of the tanks had at least

enough water in them to avoid running dry during the test, and the amount of water that would be added to the tanks during later fill would not be enough to cause the tanks to overflow. Refill verification and Baseline testing verified this prediction, and the Drain Stage timing input was selected at 55 seconds.

It should be noted here that the extent of the Drain Stage is very much an artifact of the test bed construction. The venturi tube serves a two-fold purpose: first it allows measurement of the flow rate out of the test bed, and second it constricts the flow so that the Auto Sequence timing would work for Fail-Operational testing. The latter purpose is solely to overcome the design limitation of the current SFS test bed. It was only feasible to use 1” tubing for everything, rather than design the system to use optimal line sizes for quick fills and fast drains.

The results of the Calibration tests became the standard timing settings for all of the Auto Sequences used in the Baseline and Fail-Operational tests. The timing inputs are summarized below in Table 5.1.

Table 5.1 Standard Auto Sequence Timing

<b><u>Timing Input</u></b>	<b><u>Value (seconds)</u></b>
Drain	55
Vent	22
Fill	25
Fill/Vent	15
Press	7.5

The last Calibration test that was run was the Refill Verification. The purpose of this test was to confirm that the tanks would not be filled to overflow if a second Fill Stage was done (meaning the Drain Stage had not taken enough water out of the tanks). This test was successful, confirming that after a refill the tanks would have enough water for the next Drain (10.7 to 11.1 gallons) and would not overflow.

## 5.2 Baseline Tests

The main variable that distinguished the Baseline tests was the Drain Overlap time. During the 2005 experiment,<sup>10</sup> the time during which Drain Stages of the tanks were overlapped was something that was experimented with in order to reduce the pressure fluctuations that were seen in the outflow during the transition from one run tank to another. As the 2007 experiment used a slightly differently hardware configuration, it was decided to run a series of normal operation 2-cycle Auto Sequences and vary the Overlap timing. The test that showed the smallest tank transition pressure fluctuation would be used as the Baseline for the Fail-Operational tests.

The six Baseline tests that were run used Overlap times of 0, 0.1, 0.2, 0.3, 0.4, and 0.5 seconds. Initial analysis of the results involved calculating the average outflow pressure for the entire Sequence, specifying a time range in the data and calculating the maximum percent error from the average pressure that was seen in that time range. This was done for all of the tank transitions in each Baseline test. It was found that on average, the tank transitions for the zero Overlap test had the smallest pressure fluctuations during tank transition. For this reason, it was decided to use an Overlap timing value of zero for all of the Fail-Operational testing.



The analysis of the Baseline data was done fairly quickly in order to move on to the Fail-Operational testing, and the method was slightly crude. Selecting time ranges was done by examining the chart of outflow pressure versus time and picking where the pressure fluctuations seemed to start and end. A more quantitative analysis was done later adapting a method used to analyze the Fail-Operational test results. This analysis compared each point of data to the calculated average pressure to get a percent error. Any data points with a percent error over a given threshold (typically 2%) were marked and reported. Using this analysis, it was confirmed the results of the earlier analysis: that the zero overlap test had the smallest drain transition pressure fluctuations (less than 2%) of all of the Baseline tests. See Appendix C for the analysis results.

Using the data from the zero overlap Baseline test, the water outlet pressure is shown in Figure 5.1. Upon close inspection the pressure does vary during operation, but remains within a few psi of the mean pressure.

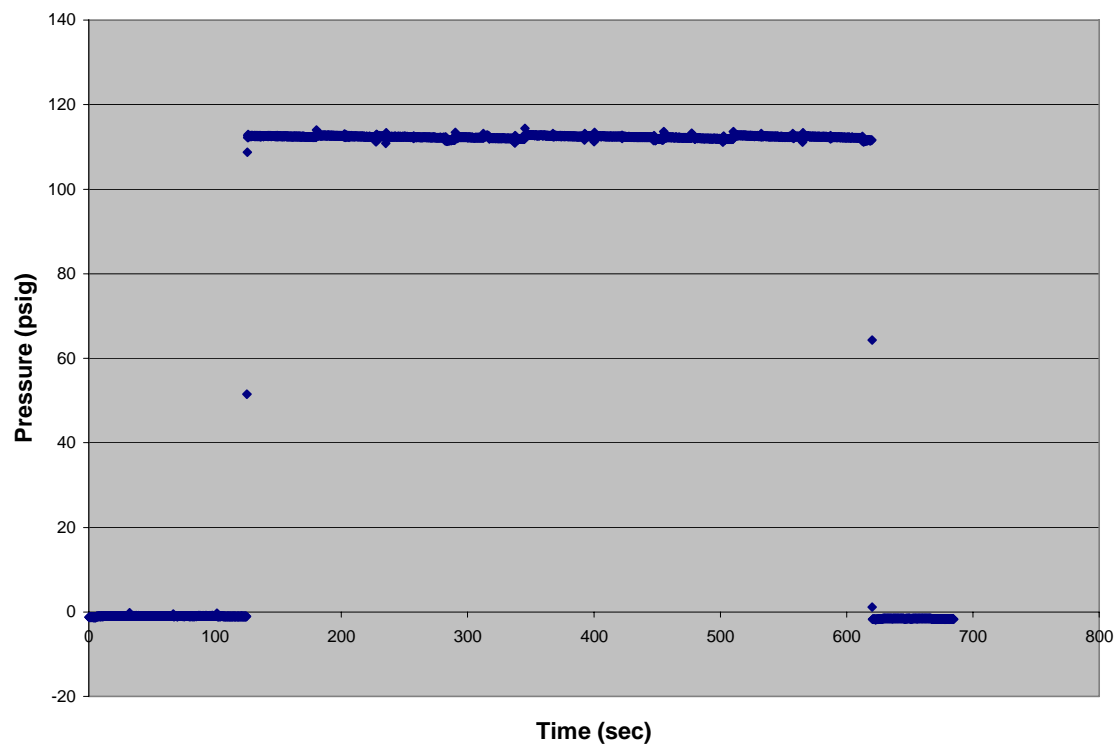


Figure 5.1 Baseline Test Water Outlet Pressure

Figure 5.2 shows the region a few seconds before and after a Drain transition.

The black data points are the water outlet pressure data, and the other points are the Drain valve control signals (plotted on a secondary axis from 0-5). These control signals help identify when in the data set certain events (such as Drain transition) occur.

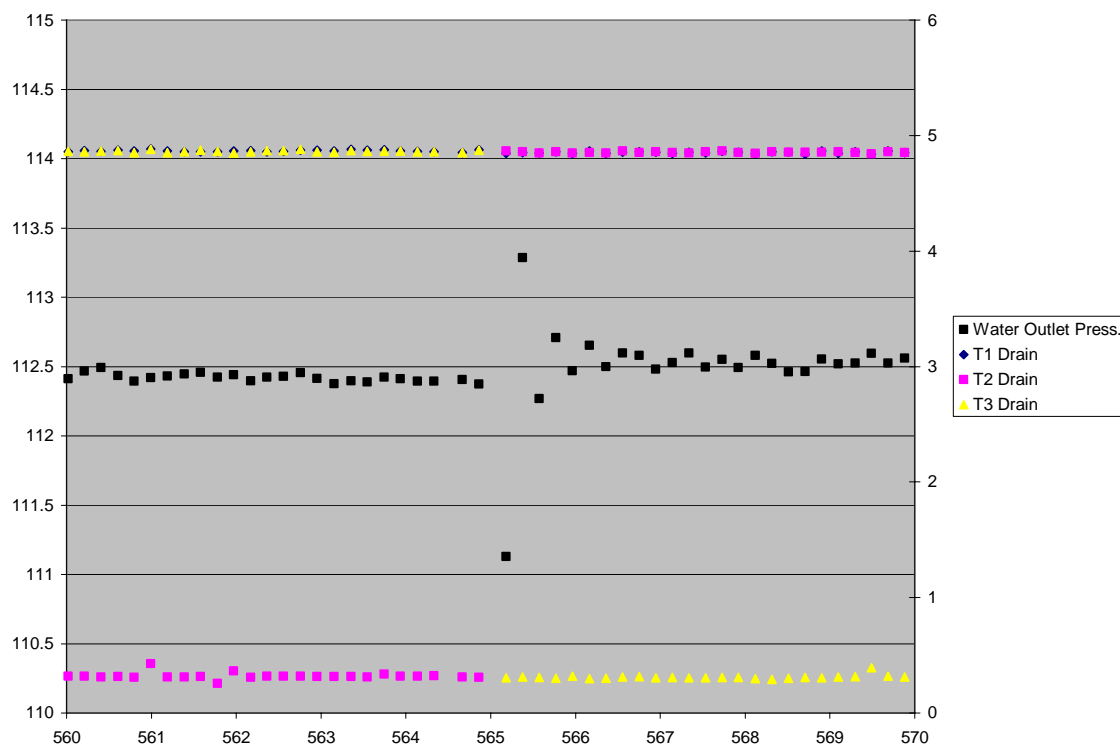


Figure 5.2 Baseline Test Drain Transition

### 5.3 Fail-Operational Tests

A Fail-Operational test consisted of three Sequences run one after the other. The system began completely drained of all water and depressurized. Then, the Refill File Sequence was run to execute a Fill Stage on all three tanks, putting them into a filled and pressurized state. This was the expected precondition for an Auto Sequence. Next, the Auto Sequence was run with the standard timing and whatever failure settings were required for the particular test. After the Auto Sequence was completed a Purge File Sequence was run to completely drain all the tanks and reset the system for the next test.

As defined earlier, a Success result for a test was the case where outflow pressure was not adversely affected by the injected failure, and an Error results was the case with a significant pressure change. The analysis for the tests was done by computing the average of the outflow pressure for the first cycle, since all of the tests held off on failure injection until the second cycle. Each data point of the outflow pressure was then compared to this average to calculate a percent error. An Excel program was written to go through the data and identify pressure “events” where the percent error exceeded a threshold value of 2%. This value was chosen as it was the second lowest outflow variation recorded in the 2005 experiment (the lowest was 1.5%, which is easily rounded to 2%).

To quantify Success and Error conditions, it was decided that a test would be considered to have an Error result if any pressure event had a maximum pressure variation that exceeded 5%. This value was chosen as it was the pressure fluctuation recorded in the 2005 experiment for the zero second drain overlap case. It was found in the analysis that all of the pressure events occurred at or near a drain transition (for the 2% threshold). Detailed results of the analysis are included in Appendix C, and the summarized results are below in Table 5.2.

Table 5.2 Fail-Operational Test Results

<b>Test #</b>	<b>Failure Type</b>	<b>Valve</b>	<b>Tank #</b>	<b>Cycle Stage</b>	<b>Result</b>
1	Start	Vent	1	Vent	Success
2	Start	Drain	1	Drain	Error
3	Start	Press	1	Drain	Success
4	Start	Press	1	Press	Success
5	Start	Fill	1	Fill	Success
6	Start	Vent	1	Fill	Success
7	Stop	Vent	1	Vent	Success
8	Stop	Drain	1	Drain	Error
9	Stop	Press	1	Drain	Error
10	Stop	Press	1	Press	Success
11	Stop	Fill	1	Fill	Success
12	Stop	Vent	1	Fill	Success
13	Start	Vent	2	Vent	Success
14	Start	Drain	2	Drain	Error
15	Start	Press	2	Drain	Error
16	Start	Press	2	Press	Success
17	Start	Fill	2	Fill	Success
18	Start	Vent	2	Fill	Success
19	Stop	Vent	2	Vent	Success
20	Stop	Drain	2	Drain	Error
21	Stop	Press	2	Drain	Error
22	Stop	Press	2	Press	Success
23	Stop	Fill	2	Fill	Success
24	Stop	Vent	2	Fill	Success
25	Start	Vent	3	Vent	Success
26	Start	Drain	3	Drain	Error
27	Start	Press	3	Drain	Error

## CHAPTER 6

### CONCLUSIONS AND FUTURE WORK

The 2007 SFS test program accomplished all of the goals set out in Chapter 4. The SFS test bed was successfully moved to and reconstructed at the PRC facility. Calibration testing of the test bed provided the data necessary to run Auto Sequences through the ICM. The Baseline tests demonstrated that the test bed was functional and could operate with minimal outflow pressure variations during tank transition (less than 2% of the average outflow pressure for the zero drain overlap case). Lastly, the Fail-Operational tests demonstrated that outflow pressure could be maintained during most of the injected failure scenarios and subsequent switch to two-tank operation.

#### 6.1 Conclusions

The key result of the 2007 test program was the quantification of the effects of the various failure modes. Two-thirds of the tests resulted in the Success condition, meaning that at no point during the test did the outflow pressure vary more than 5% from the average outflow pressure. To examine the physical meaning of these results, the explanation for the the Success tests and a commentary on the Error tests is as follows.

In developing the failure response code for the ICM, it was realized that Start and Stop failures fall into two categories: failures that require the next tank to begin draining

immediately and those that do not. An example of the former type of failure are tests 2 and 3 (from Table 5.2). If either the Pressurization or Drain valves fail to open (Start failure) when a Drain Stage is supposed to begin, the Pressurization and Drain valves on the next tank in the sequence must open immediately to maintain outflow. It should be noted that these failures, replicated for tanks 2 and 3 in tests 14 & 15 and 26 & 27, respectively, come up as Error tests. The exception is test 3, wherein the Pressurization valve failed to open. This result can be explained by the fact that the tanks all end up pressurized before the start of the Drain Stage, and that the pressure contained in the tank was enough to maintain outflow pressure just long enough for the system to realize the error and switch to the next tank. That this does not happen in the other similar tests is probably due to asymmetries in the tanks.

The reason why the Start failures in Drain or Pressurization valves (in the Drain Stage) come up with the Error result is due to the control system. The way the ICM is coded the failure is injected through the commands to the valves, and then the data on the hardware controller is read back into the computer to check for failure. All of this takes time, and the results of these tests demonstrate that in order to function correctly the SFS needs a control system that can function very rapidly. A failure that requires an immediate fail-over to two-tank mode needs to be detected in milliseconds, rather than the tenths of seconds the current control system is capable of. For the 2007 experiment, the computer system that was used was what was on hand and easily accessible at the PRC. The machine that ran the ICM was not designed specifically for speed, nor was the ICM optimized to run quickly. Another way to minimize the effects of these types of failures is to run the Sequence with a non-zero drain overlap. Time did not permit testing

of this idea and, as the results of the Baseline tests show, additional work needs to be done to determine how to run the Auto Sequence with drain overlap and minimize tank transition variations in outflow pressure.

The results of the Start failure tests of the Drain and Pressurization valves also provide insight into the cases where the system behaved successfully. It should be noted that all of the other Start failure tests resulted in the Success condition. This is because all of the other Start failures fall into the second category: failures that do not require the next tank to start draining immediately. For example, test 1 had a Start Failure in the Vent valve of Tank 1 during the Vent Stage. When this failure happened, Tank 1 was done with its Drain, Tank 2 was Draining, and Tank 1 was in the middle of its refill process. Since no immediate action was required, Tank 2 continued to drain while the program shut down the refill process of Tank 1. The rest of the Sequence alternated between Tanks 2 and 3, and isolated the failed valve in Tank 1. The similar results (and causes) for the other Start failures show that these types of failure modes are non-detrimental to system operation as long as the failure is identified and the affected tank isolated.

Similar to the Start failures that resulted in Success, many of the Stop failure tests were also successful. This is for the same reason that prevented Start failures in the refill process from affecting outflow. The only difference was in the valve's final state (open for the Stop failure and closed for the Start failure), and as long as that valve could be isolated from the rest of the system no large pressure variations were observed.

The Stop failures that did have a significant effect on outflow pressure were tests 8, 9, 20, and 21. In these cases either the Drain or Pressurization valve failed to



close at the end of a Drain Stage. This always created a problem, especially in the case of the Drain valve because the affected tank could not be isolated from the rest of the system. When a Stop failure in a Drain valve was detected the system automatically closed all of the other valves in that tank, but the Drain valve itself was left open. This means that when the Drain Stages of the remaining two tanks happened, there were two paths for the water: into the drain manifold and into tank with the failed valve. Eventually equilibrium was reached such that the pressure built up in the failed tank was enough to prevent such diversion of water, but at the start of the failure the effects on outflow pressure were significant. A solution to this problem would be the installation of check valves on the Drain manifold, in much the same manner that they were installed on the Pressurization manifold. With check valves, even if a Drain valve failed in the open position, there would be no cross-talk between tanks.

The case of a Stop Failure in the Pressurization valve during a Drain Stage can be explained by the limitations of the air and control system. As mentioned earlier, the control system did not recognize and respond to failures immediately. In this case that would mean that the Vent valve would be open briefly while the Pressurization valve was also open. This is because when the failure is injected the ICM expects that the Drain Stage is over and the Vent Stage should begin, it also doesn't know that the Pressurization valve has failed open yet (as that data hasn't been read in and processed). This direct route from the air supply system to atmospheric pressure sapped its ability to provide air flow to the tank that was in its Drain Stage, and so the pressure in and downstream of that tank was not as high as it should have been. The check valves

installed on the pressurant lines prevented this kind of error from being more problematic than it was, but the more permanent solution is a faster control system.

Overall the results of the Fail-Operational testing are very promising and not outside of expectations. These tests show some of the possible failure modes are of greater concern than others, and identifying those modes was a goal of the test program. By scrutinizing the physical workings of the SFS during these failures, obvious solutions to these problems can be found. Additional testing would have to be done to determine the best corrective action to be taken for the failure modes that resulted in the Error condition.

In conclusion, it has been experimentally demonstrated that the Sequential Feed System is able to provide liquid flow with only minor variations in pressure and flow rate, in both this test program and the 2005 experiment. Additionally, the 2007 test program demonstrated that many of the possible failure modes for Start and Stop failures do not affect outflow pressure in a significant manner, and those that do can be corrected with improved system hardware. Combined with the analysis work that has been done to show that the SFS is better than conventional pressured fed systems and competitive with turbopumps, these results establish that the Sequential Feed System is a promising new technology for liquid propellant feed systems.

## 6.2 Future Work

While the current test bed was adequate to the task of the initial Fail-Operational testing, there are several avenues of research that could not be explored due to time or

hardware constraints. The topics listed in this section are ones that should be developed in future work on the SFS in order to fully demonstrate the system's capabilities.

**Test Bed Modifications.** As mentioned in Section 5.2, the Fail-Operational tests that resulted in the Error condition showed that certain failure modes have more detrimental effects than others. In order to negate these effects, improved or modified system hardware would have to be used. Such modifications include a faster computer system for data acquisition and valve control, check valves installed on the drain manifold to prevent cross talk between tanks, and an improved air system that won't decrease pressurant to one tank if another Pressurization valve is opened.

**Other Failure Types.** Time constraints prevented all of the possible failure modes from being investigated in the 2007 experiment. Specifically omitted from the tests were the Power On and Power Off Failure types. The current test bed could be used to examine the effects of sudden power surges or losses to the various valves at various times during the Sequence. Such an investigation would not require additional hardware, and the ICM is already configured for such tests.

**Unimplemented ICM Features.** Future SFS tests may want to examine the effects of Leak Failures, use multiple failure detection tiers in the software (with the attendant sensor hardware requirements), or run the entire system using a Condition Sequence. The features that were planned for the ICM but were left unimplemented are described more fully in Chapter 3. All of these features would allow for greater realism in the SFS test bed as they represent features that would be present on a real system.

**Propellant Tests.** The SFS experiments thus far have only used simulated propellant and pressurant. The next experimental step to prove the SFS's worth as a feed

system is to conduct tests using a full scale test bed that is equipped to handle real propellant. The current test bed hardware may be inadequate for such a task, so appropriate tanks, valves, and other components would have to be obtained.

## **APPENDICES**

## APPENDIX A

### EXCEL AND VISUAL BASIC FOR APPLICATIONS CODE

#### A.1 Custom VBA I/O Methods

The VBA syntax for reading or writing to an Excel cell can be cumbersome. During the implementation of the analysis spreadsheets, it was decided to write custom cell I/O code to simplify those functions. This code was included in a module in all of the spreadsheets that had need of it. The module code, including the comments (denoted in VBA by a ' character in front of the line) that explain the code operation, is given below.

##### A.1.1 Introductory Comments

```
' Excel I/O Library version 2.0
' By Chris Morton
' Copyright 2006
' Free for non-profit use.
'
' The following methods are intended to simplify cell input
' and output in an Excel worksheet. The methods included
' in this library are read, writeTo, getCell, getAddress,
' and getCol. All methods are public, so they may be
' called by any other procedure in the active workbook.
' Method descriptions and syntax are as follows.
'
' read
' This is an input method for getting data from a cell and
' storing it in a Visual Basic for Applications (VBA)
```

' variable. It has one required input parameter and two  
 ' optional input parameters. A string called TheTarget is  
 ' always required. Typically this string contains the  
 ' address of the cell that is to be read from.  
 ' The optional parameters are RowNum, specifying the row of  
 ' a worksheet where the target cell is located, and  
 ' SheetName, which tells the program which worksheet to  
 read from.

'  
 ' The default mode of read is to read from a specific cell  
 ' on the active worksheet. For this mode only TheTarget is  
 ' required, and that input parameter must be in the same  
 ' format as a cell address that would be used in a  
 ' worksheet formula. For example:

' Result = read("A1")  
 '

' If the column where the cell to be read is known but the  
 ' row is not (in the case where you would like to read in  
 ' the data from many cells in a column), then the RowNum  
 ' input may be used to specify a row to read from. When  
 ' read is used like this TheTarget is no longer a cell  
 ' address, but the name of a column. For example: to read  
 ' from row 1 of column A:

' Result = read("A", 1)  
 '

' Lastly, SheetName can be used to specify any sheet where  
 ' the read is supposed to be done. If SheetName is left  
 ' blank the method will default to the active sheet. Note  
 ' that SheetName is a string, and so using this parameter  
 ' requires you to know the name of the worksheet. For  
 ' Example:

' Result = read("A1", , "Sheet1")  
 ' or  
 ' Result = read("A", 1, "Sheet1")  
 ',  
 '

' writeTo  
 ' This is an output method that has very similar syntax to  
 ' the read method. The input parameters are the same with  
 ' the exception of the Arg Variant. This is a required  
 ' parameter that must be input first. Arg is that data  
 ' that is to be written to the specified cell. The other  
 ' parameters, TheTarget, RowNum, and SheetName, all work  
 ' the same way as in read. For example:

```

'
' Writing to a known cell.
' writeTo "Hello World", "A1"
'
' Writing to a row in a column.
' writeTo "Hello World", "A", 1
'
' Writing to a specific worksheet
' writeTo "Hello World", "A1", , "Sheet1"
' or
' writeTo "Hello World", "A", 1, "Sheet1"
'
'
' getCell
' This is a helper method for writeTo and read. It takes a
' string parameter representing a cell column and an
' integer parameter representing a cell row and returns an
' Excel cell address. For example:
'
' Result = getCell("D", 1)      ' Result = D1
'
' writeTo Temp, "D", 1
' Equivalent to next line of code
' writeTo Temp, getCell("D", 1)
'
'
' getAddress
' This is a helper method for the writeTo and read methods.
' It takes three parameters: Target, ColVal, and RowVal.
' Target is the string cell address for a cell, while
' ColVal and RowVal are Longs that act as output for the
' method, since the variables are passed by reference. The
' function of getAddress is to get the column and row
' values from the string address and return them to the
' calling method. A syntax example is:
'
' Call getAddress("A5", Column, Row)
'
' After getAddress completes, Column will be equal to 1,
' and Row will be equal to 5. Since this method takes
' advantage of pass by reference parameters, it is
' important that the variables used for Row and Column
' output do not have important data in them before
' getAddress is called. Row and Column must also be of the
' Long data type in order for this method to work.
'
'

```



```

' getCol
' This method is intended to allow for horizontal iteration
' across columns using the read and writeTo methods. While
' read and writeTo allow a numerical variable (RowNum) to
' specify the row for input/output, the name of the column
' must be in the form of a string. If it is easier to keep
' track of columns in terms of numbers, then this method
' may be used to convert a number to the corresponding
' column name, with column 1 being column "A." The syntax
' is:
'
' Result = getCol(1)           ' Result = "A"
' Result = getCol(26)          ' Result = "Z"
' Result = getCol(27)          ' Result = "AA"

```

### A.1.2 read Function

```

Public Function read(TheTarget As String, Optional RowNum
As Long, Optional SheetName As String) As Variant
    Dim Row As Long
    Dim Column As Long
    Dim Result As Variant
    Dim Target As String

    If SheetName = "" Then
        SheetName = ActiveSheet.Name
    End If

    If RowNum > 0 Then
        ' Treat the String TheTarget as the Column name
        Target = getCell(TheTarget, RowNum)
    Else
        Target = TheTarget
    End If

    Call getAddress(Target, Column, Row)

    ' Return the value at the particular row and column of
    ' the current sheet.
    read = Worksheets(SheetName).Cells(Row, Column)
End Function

```

### A.1.3 writeTo Method

```
Public Sub writeTo(Arg As Variant, TheTarget As String,
Optional RowNum As Long, Optional SheetName As String)
    Dim Target As String
    Dim Row As Long
    Dim Column As Long

    If SheetName = "" Then
        SheetName = ActiveSheet.Name
    End If

    If RowNum > 0 Then
        ' Treat the String TheTarget as the Column name
        Target = getCell(TheTarget, RowNum)
    Else
        Target = TheTarget
    End If

    Call getAddress(Target, Column, Row)
    Worksheets(SheetName).Cells(Row, Column) = Arg
End Sub
```

### A.1.4 getCell Function

```
Public Function getCell(Column As String, Row As Long) As
String
    Dim Result As String
    Result = Str(Row)
    Result = Trim$(Column) + Trim$(Result)
    getCell = Result
End Function
```

### A.1.5 getAddress Method

```
Public Sub getAddress(Target As String, ColVal As Long,
RowVal As Long)
    Dim Entry As Long
    Dim ColNum As Long
    Dim Char
    Dim Check As Boolean
    Dim Length As Long

    ' Initialize variables
    Length = Len(Target)
```

```

Check = True
Entry = 1

' Loop through the first characters of the string which
' contain the column identifier (in letters), and
' change that to a number.
Do While Check = True
    Char = Mid(Target, Entry, 1)
    ColNum = (Asc(Char)) - 64
    If ColNum < 0 Then
        Check = False
        Entry = Entry - 1
    ElseIf Entry = 1 Then
        ColVal = ColNum
    ElseIf Entry > 1 Then
        ColVal = (26 * ColVal) + ColNum
    End If
    Entry = Entry + 1
Loop

' Read the remainder of the Target string to get the
' row identifier.
Length = Length - Entry + 1
Char = Mid(Target, Entry, Length)
RowVal = Val(Char)

'ColVal and RowVal are passed by reference back to the
'calling method.
End Sub

```

#### A.1.6 getCol Function

```

Public Function getCol(ColNum As Long) As String
    Dim Result As String
    Dim Char As String
    Dim Counter As Long
    Dim NewCol As Long

    ' Clear the result and initialize the loop variables.
    Result = ""
    NewCol = ColNum
    Counter = 0
    ' Loop until you get a 1-26 number corresponding to the
    ' right-most letter in the column name. At the same
    ' time, Counter is used to keep track of the left
    ' letters in the column name. If the loop executes once

```

```

' (NewCol > 26), then Counter will be 1. This
' corresponds to a column between AA and AZ. The left
' A is set by Counter, while the A-Z on the right is
' set by whatever NewCol finishes as.
Do While ((NewCol - 26) > 0)
    NewCol = NewCol - 26
    Counter = Counter + 1
Loop

' Should Counter be greater than 26 that means the
' specified column is a three or more letter one.
' getCol is called recursively to handle that.
If Counter > 26 Then
    Result = getCol(Counter)
End If

' This block handles the case where the column name has
' two letters in it. 64 is added to Counter to convert
' it to the ASCII number format, which is then used in
' the Chr function. This is then appended to the end of
' the Result string, allowing for the possibility of
' earlier recursion.
If Counter > 0 Then
    Result = Result + Chr(Counter + 64)
End If

' Like the previous block, this adds on the rightmost
' letter of the column name (or the column name itself
' if only a one-letter name).
Result = Result + Chr(NewCol + 64)

getCol = Result
End Function

```

## A.2 Analysis Spreadsheets

### A.2.1 Mass Project

	A	B	C	D	E	F	G	H	I
1	Inputs			Outputs			Masses		
2									
3	Chamber Pressure (psia)	1015		Mass Flow Rate (lbm/sec)	177.7707		Top Disk	679.7058301	
4									
5	Thrust (lbf)	1740162		Throat Area (ft^2)	0.416682		Barrel	4078.23498	
6									
7	Isp (sec)	304		Exit Area (ft^2)	6.666915		Converging Nozzle	1305.035194	
8									
9	Chamber Temperature (deg R)	6606		Throat Diameter (ft)	0.728564		Diverging Nozzle	4678.551169	
10									
11	Gamma	1.24		Barrel Diameter (ft)	3.64282		Total	10741.52717	
12									
13	Specific Gas Constant (ft-lbf/lbm deg R)	7.65		Barrel Length (ft)	5.464229				
14							Percent Error	-41.93455228	
15	Expansion Ratio	16		Wall Thickness (ft)	0.130499				
16									
17	Wall Material Density (lbm/ft^3)	500					Calculated Mass	10741.52717	
18									
19	Wall Material Tensile Strength (psia)	85000							
20									
21	Wall Material Safety Factor	1.5		Auto Calculate					
22									
23	Target Mass (lbm)	18499							
24									

Figure A.1 Mass Project Spreadsheet

Table A.1 Mass Project Cell Formulas

Cell	Formula
E3	$B5/(B7*32.2)$
E5	$(E3/(B3*144))*SQRT((B13*B9/B11)*((2/(B11+1))^{((-B11-1)/(B11-1))}))$
E7	$E5*B15$
E9	$SQRT(4*E5/3.14)$
E11	$5*E9$
E13	$1.5*E11$
E15	$2*B3*E11*B21/B19$
H3	$B17*E15*3.14*25*E9*E9/4$
H5	$B17*E15*5*3.14*E13*E9$
H7	$B17*E15*12*3.14*E9*E9$
H9	$B17*E15*3.14*E9*E9*((2.868*B15)-2.868)$
H11	$H3+H5+H7+H9$
H14	$100*((H11-B23)/B23)$
H17	$2*(40*B21*B17/((B19*144)*(SQRT(3.14*B3*144))))* ((B5/(B7*32.2))^{(3/2)})* (((B13*B9/B11)*((2/(B11+1))^{((-B11-1)/(B11-1))}))^{(3/4)})* (52.882+(2.868*B15))$

Table A.2 Mass Projects Inputs

Name	Propellants	Thrust (lbf)	Chamber Pressure (psi)	Mass(lb)
17D11	LOX/Kerosene	19400	1152	507
F-1	LOX/Kerosene	1740162	1015	18499
H-1	LOX/Kerosene	213065	580	1400
LR-79-7	LOX/Kerosene	170565	595	1418
RS-56-OBA	LOX/Kerosene	235343	696	1774
RS-56-OSA	LOX/Kerosene	86862	696	1014
RD-861	N2O4/UDMH	17694	1288	271
Viking 2	N2O4/UDMH	155789	798	1711
Viking 2B	N2O4/UDMH	161861	856	1711
Viking 4	N2O4/UDMH	162079	783	1874
Viking 4B	N2O4/UDMH	180971	848	1874
Viking 5C	N2O4/UDMH	169057	798	1710
HM-10	LOX/LH2	13889	435	320
HM7-A	LOX/LH2	13864	435	328
J-2	LOX/LH2	232261	435	3170
LE-5	LOX/LH2	23148	522	540
LE-7	LOX/LH2	242343	1842	3779
M-1	LOX/LH2	1199574	986.3	19991
MA-5A	LOX/LH2	472106	696	3550
RL-10	LOX/LH2	15000	348	289
RL-10A-1	LOX/LH2	15000	348	289
RL-10A-3A	LOX/LH2	16501	469	311
RL-10A-4	LOX/LH2	20796	567	370
RL-10A-5KA	LOX/LH2	22590	592	320
RL-10B-X	LOX/LH2	20997	1480	699
RL-10C	LOX/LH2	34994	1479	699
RS-68	LOX/LH2	744737	1391	14544
STME	LOX/LH2	649708	2220	7937
Vulcain	LOX/LH2	241668	1479	2866
Vulcain 2	LOX/LH2	292251	1682	3968
Bell 8081	Nitric Acid/UDMH	16000	145	287
LR-91-11	N2O4/Aerozine-50	103302	849	1298
LR-91-7	N2O4/Aerozine-50	100000	725	1246
LR-91-5	N2O4/Aerozine-50	100034	815	1102
Aestus	N2O4/MMH	6500	145	300

Table A.3 Mass Project Inputs

Name	Area Ratio	Thrust/Weight Ratio	Isp (s)	Temp (deg R)	Gamma	R
17D11	189	38.26	362	6606	1.24	7.65
F-1	16	94.07	304	6606	1.24	7.65
H-1	8	152.19	289	6606	1.24	7.65
LR-79-7	8	120.29	282	6606	1.24	7.65
RS-56-OBA	8	132.66	299	6606	1.24	7.65
RS-56-OSA	25	85.66	316	6606	1.24	7.65
RD-861	112.4	65.29	317	6093	1.25	9.91
Viking 2	11	91.05	280	6093	1.25	9.91
Viking 2B	11	94.60	278	6093	1.25	9.91
Viking 4	31	86.49	296	6093	1.25	9.91
Viking 4B	30.8	96.57	296	6093	1.25	9.91
Viking 5C	11	98.86	278	6093	1.25	9.91
HM-10	61	43.40	443	5373	1.26	45.45
HM7-A	62.5	42.27	443	5373	1.26	45.45
J-2	28	73.27	421	5373	1.26	45.45
LE-5	140	42.87	450	5373	1.26	45.45
LE-7	52	64.13	446	5373	1.26	45.45
M-1	28	60.01	428	5373	1.26	45.45
MA-5A	8	132.99	296	5373	1.26	45.45
RL-10	40	51.90	410	5373	1.26	45.45
RL-10A-1	47	51.90	425	5373	1.26	45.45
RL-10A-3A	61	53.06	444	5373	1.26	45.45
RL-10A-4	84	56.21	449	5373	1.26	45.45
RL-10A-5KA	8.2	70.59	398	5373	1.26	45.45
RL-10B-X	250	30.04	470	5373	1.26	45.45
RL-10C	190	50.06	450	5373	1.26	45.45
RS-68	21.5	51.21	420	5373	1.26	45.45
STME	45	81.86	431	5373	1.26	45.45
Vulcain	45	84.32	431	5373	1.26	45.45
Vulcain 2	61.5	73.65	434	5373	1.26	45.45
Bell 8081	45	55.75	240	5805	1.23	9.97
LR-91-11	49.2	79.59	316	5805	1.2	7.8
LR-91-7	45	80.26	316	5805	1.2	7.8
LR-91-5	49.2	90.77	315	5805	1.2	7.8
Aestus	83	21.67	324	6093	1.2	14.58

Table A.4 Mass Project Outputs

Name	Percent Error	Total Mass	Disk Mass	Barrel Mass	CN Mass	DN Mass	Mdot
17D11	-89.148	55.01513	0.577954	3.4677248	1.109672	49.85978	1.664322
F-1	-41.934	10741.527	679.7058	4078.235	1305.035	4678.551	177.7707
H-1	-63.98	504.2262	41.56112	249.36674	79.79736	133.501	22.89594
LR-79-7	-73.911	369.931	30.49177	182.95061	58.54419	97.94444	18.78386
RS-56-OBA	-71.377	507.76005	41.8524	251.11442	80.35661	134.4366	24.44411
RS-56-OSA	-83.020	172.17284	8.637526	51.825156	16.58405	95.12611	8.536638
RD-861	-85.352	39.693825	0.661131	3.9667888	1.269372	33.79653	1.733448
Viking 2	-79.129	357.09114	26.43396	158.60379	50.75321	121.3002	17.27917
Viking 2B	-78.428	369.08096	27.32152	163.92912	52.45732	125.373	18.0818
Viking 4	-68.459	591.06178	26.05357	156.32144	50.02286	358.6639	17.0051
Viking 4B	-64.386	667.3885	29.53749	177.22495	56.71198	403.9141	18.98722
Viking 5C	-76.138	408.0313	30.20485	181.22911	57.99332	138.604	18.88567
HM-10	-84.507	49.576159	1.36001	8.1600577	2.611218	37.44487	0.973669
HM7-A	-84.641	50.37596	1.356339	8.1380357	2.604171	38.27741	0.971916
J-2	-32.515	2139.2471	100.3881	602.32882	192.7452	1243.785	17.13319
LE-5	-64.870	189.69773	2.609167	15.655004	5.009601	166.424	1.597516
LE-7	-59.213	1541.3256	47.68528	286.11169	91.55574	1115.973	16.87484
M-1	-18.623	16268.02	763.407	4580.4419	1465.741	9458.429	87.04171
MA-5A	33.325	4733.0485	390.1241	2340.7448	749.0383	1253.141	49.53269
RL-10	-82.214	51.399204	1.916714	11.500281	3.68009	34.30212	1.136191
RL-10A-1	-81.129	54.535959	1.816141	10.896847	3.486991	38.33598	1.096091
RL-10A-3A	-80.186	61.619922	1.690403	10.142418	3.245574	46.54153	1.154174
RL-10A-4	-72.825	100.54427	2.138919	12.833516	4.106725	81.46511	1.438393
RL-10A-5KA	-89.152	34.712483	2.839714	17.038285	5.452251	9.382234	1.762695
RL-10B-X	-77.899	154.48564	1.254134	7.5248043	2.407937	143.2988	1.387406
RL-10C	-60.574	275.58184	2.881199	17.287194	5.531902	249.8815	2.415045
RS-68	-59.237	5928.5018	323.4839	1940.9032	621.089	3043.026	55.06781
STME	-26.384	5842.8348	200.7108	1204.2646	385.3647	4052.495	46.815
Vulcain	-43.338	1623.9304	55.78462	334.70771	107.1065	1126.332	17.4135
Vulcain 2	-36.356	2525.3873	68.84496	413.06976	132.1823	1911.29	20.91271
Bell 8081	-74.509	73.158827	2.513123	15.07874	4.825197	50.74177	2.070393
LR-91-11	-77.267	295.06581	9.506594	57.039563	18.25266	210.267	10.15233
LR-91-7	-77.108	285.23291	9.798209	58.789252	18.81256	197.8329	9.827817
LR-91-5	-73.834	288.34809	9.290159	55.740952	17.8371	205.4799	9.862368
Aestus	-91.004	26.985859	0.579741	3.4784438	1.113102	21.81457	0.623035



Table A.5 Mass Project Outputs

Name	A*	Ae	D*	Db	Lb	t
17D11	0.003437	0.649617	0.06617	0.330851	0.496277	0.013452
F-1	0.416682	6.666915	0.728564	3.64282	5.464229	0.130499
H-1	0.093916	0.751331	0.345888	1.729441	2.594162	0.035403
LR-79-7	0.075107	0.600854	0.309318	1.546588	2.319882	0.032478
RS-56-OBA	0.083556	0.668445	0.326252	1.63126	2.446889	0.040071
RS-56-OSA	0.02918	0.729505	0.192801	0.964005	1.446008	0.023681
RD-861	0.00349	0.392268	0.066677	0.333383	0.500074	0.015155
Viking 2	0.056149	0.617638	0.267446	1.337229	2.005844	0.037663
Viking 2B	0.054776	0.602534	0.264156	1.320778	1.981166	0.039903
Viking 4	0.056317	1.745824	0.267846	1.339228	2.008842	0.03701
Viking 4B	0.058061	1.788288	0.271962	1.359811	2.039717	0.040698
Viking 5C	0.061369	0.675062	0.279602	1.398011	2.097017	0.039375
HM-10	0.01164	0.710014	0.121768	0.608841	0.913261	0.009347
HM7-A	0.011619	0.726163	0.121658	0.608292	0.912439	0.009339
J-2	0.204816	5.734846	0.510795	2.553977	3.830966	0.039211
LE-5	0.015914	2.228011	0.142384	0.711918	1.067877	0.013116
LE-7	0.047639	2.47724	0.246347	1.231736	1.847604	0.080077
M-1	0.458916	12.84964	0.764595	3.822977	5.734466	0.13308
MA-5A	0.370082	2.960652	0.686616	3.43308	5.149619	0.084333
RL-10	0.016978	0.679121	0.147065	0.735324	1.102986	0.009032
RL-10A-1	0.016379	0.769803	0.144446	0.722231	1.083346	0.008871
RL-10A-3A	0.012797	0.780626	0.12768	0.638398	0.957598	0.010567
RL-10A-4	0.013192	1.108124	0.129634	0.648171	0.972257	0.012971
RL-10A-5KA	0.015484	0.126965	0.140443	0.702215	1.053323	0.014672
RL-10B-X	0.004875	1.218699	0.078803	0.394016	0.591023	0.020582
RL-10C	0.008491	1.613338	0.104004	0.520021	0.780032	0.027145
RS-68	0.205866	4.426126	0.512104	2.560518	3.840777	0.125706
STME	0.10966	4.934683	0.373756	1.868782	2.803173	0.146425
Vulcain	0.061226	2.755149	0.279275	1.396373	2.094559	0.072891
Vulcain 2	0.064655	3.976255	0.286989	1.434943	2.152415	0.085185
Bell 8081	0.036459	1.640633	0.215509	1.077543	1.616314	0.005514
LR-91-11	0.027247	1.340535	0.186304	0.931519	1.397279	0.027913
LR-91-7	0.030887	1.38991	0.198359	0.991796	1.487694	0.025378
LR-91-5	0.027573	1.356575	0.187415	0.937075	1.405613	0.026955
Aestus	0.013713	1.138213	0.132172	0.660858	0.991286	0.003382

The Auto Calculate button in Figure A.1 is used to run through a set of calculations automatically. The data in Table A.2 and Table A.3 are located on the

spreadsheet starting at row 30. When the Auto Calculate button is pressed, the program copies the data from those rows into the input section shown in Figure A.1. The cell formulas then calculate the answers, and then the Auto Calculate button code copies that output data back into the appropriate column on the same row that the input data came from. The VBA code is contained entirely in Sheet1, and is shown below.

```
Private Sub CommandButton1_Click()
Dim Counter As Integer
Counter = 31
Do While Counter < 66
    'Inputs
    'Chamber Pressure  B3 (3, 2)
    Worksheets("Sheet1").Cells(3, 2).Value =
Worksheets("Sheet1").Cells(Counter, 4).Value

    'Thrust  B5 (5, 2)
    Worksheets("Sheet1").Cells(5, 2).Value =
Worksheets("Sheet1").Cells(Counter, 3).Value

    'Isp  B7 (7, 2)
    Worksheets("Sheet1").Cells(7, 2).Value =
Worksheets("Sheet1").Cells(Counter, 8).Value

    'Chamber Temperature  B9 (9, 2)
    Worksheets("Sheet1").Cells(9, 2).Value =
Worksheets("Sheet1").Cells(Counter, 9).Value

    'Gamma  B11 (11, 2)
    Worksheets("Sheet1").Cells(11, 2).Value =
Worksheets("Sheet1").Cells(Counter, 10).Value

    'Specific Gas Constant  B13 (13, 2)
    Worksheets("Sheet1").Cells(13, 2).Value =
Worksheets("Sheet1").Cells(Counter, 11).Value

    'Expansion Ratio  B15 (15, 2)
    Worksheets("Sheet1").Cells(15, 2).Value =
Worksheets("Sheet1").Cells(Counter, 6).Value

    'Target Mass  B23 (23, 2)
```

```

Worksheets("Sheet1").Cells(23, 2).Value =
Worksheets("Sheet1").Cells(Counter, 5).Value

'Outputs
'Percent Error  H14 (14, 8)
Worksheets("Sheet1").Cells(Counter, 13).Value =
Worksheets("Sheet1").Cells(14, 8).Value

'Total Mass  H11 (11, 8)
Worksheets("Sheet1").Cells(Counter, 14).Value =
Worksheets("Sheet1").Cells(11, 8).Value

'Disk Mass  H3 (3, 8)
Worksheets("Sheet1").Cells(Counter, 15).Value =
Worksheets("Sheet1").Cells(3, 8).Value

'Barrel Mass  H5 (5, 8)
Worksheets("Sheet1").Cells(Counter, 16).Value =
Worksheets("Sheet1").Cells(5, 8).Value

'Converging Nozzle Mass  H7 (7, 8)
Worksheets("Sheet1").Cells(Counter, 17).Value =
Worksheets("Sheet1").Cells(7, 8).Value

'Diverging Nozzle Mass  H9 (9, 8)
Worksheets("Sheet1").Cells(Counter, 18).Value =
Worksheets("Sheet1").Cells(9, 8).Value

'Mass Flow Rate  E3 (3, 5)
Worksheets("Sheet1").Cells(Counter, 19).Value =
Worksheets("Sheet1").Cells(3, 5).Value

'Throat Area  E5 (5, 5)
Worksheets("Sheet1").Cells(Counter, 20).Value =
Worksheets("Sheet1").Cells(5, 5).Value

'Exit Area  E7 (7, 5)
Worksheets("Sheet1").Cells(Counter, 21).Value =
Worksheets("Sheet1").Cells(7, 5).Value

'Throat Diameter  E9 (9, 5)
Worksheets("Sheet1").Cells(Counter, 22).Value =
Worksheets("Sheet1").Cells(9, 5).Value

'Barrel Diameter  E11 (11, 5)

```

```
Worksheets("Sheet1").Cells(Counter, 23).Value =  
Worksheets("Sheet1").Cells(11, 5).Value
```

```
'Barrel Length E13 (13, 5)  
Worksheets("Sheet1").Cells(Counter, 24).Value =  
Worksheets("Sheet1").Cells(13, 5).Value
```

```
'Thickness E15 (15, 5)  
Worksheets("Sheet1").Cells(Counter, 25).Value =  
Worksheets("Sheet1").Cells(15, 5).Value
```

```
Counter = Counter + 1  
Loop  
End Sub
```

## A.2.2 DeltaV

	A	B	C	D	E	F	G	H	I
1	<b>Input</b>			<b>Time (s)</b>	<b>Velocity (ft/s)</b>	<b>Mass (lbm)</b>	<b>Height (ft)</b>	<b>Density (lbm/ft^3)</b>	<b>Cd</b>
2	Empty Weight (lbm)	1722000		0	0	6115000	0	0.075	0.065
3	Propellant Weight (lbm)	4393000		0.15	1.093957482	6110754.717	0.164093622	0.074999931	0.065
4	Inert Weight (lbm)	23400		0.3	2.192045087	6106509.434	0.492900103	0.074999754	0.065
5	Propulsion System Weight (lbm)	210600		0.45	3.29429963	6102264.151	0.987043345	0.074999453	0.065
6	Specific Impulse (s)	265		0.6	4.400757916	6098018.868	1.64715191	0.074999015	0.065
7	Burn Time (s)	150		0.75	5.511456748	6093773.585	2.473859004	0.074998427	0.065
8	Initial Velocity (ft/s)	0		0.9	6.626432924	6089528.302	3.467802463	0.074997681	0.065
9	Initial Altitude (ft)	0		1.05	7.745723235	6085283.019	4.629624748	0.074996767	0.065
10	Cross Section Area (ft^2)	845.4		1.2	8.869364466	6081037.736	5.959972924	0.074995677	0.065
11	Target Velocity (ft/s)	7600		1.35	9.997393396	6076792.453	7.459498653	0.074994404	0.065
12	Initial Flight Angle (degrees)	90		1.5	11.1298468	6072547.17	9.128858177	0.074992941	0.065
13	Final Flight Angle (degrees)	20		1.65	12.26676143	6068301.887	10.96871231	0.074991281	0.065
14	Time to Start Turn (s)	0		1.8	13.40817406	6064056.604	12.97972642	0.074989419	0.065
15	Time to End Turn (s)	70		1.95	14.55412142	6059811.321	15.1625704	0.074987348	0.065
16	Initial Thrust (lbf)	7500000		2.1	15.70464026	6055566.038	17.51791871	0.074985063	0.065
17	Second Phase Thrust (lbf)	7800000		2.25	16.85976731	6051320.755	20.04645027	0.074982558	0.065
18	Third Phase Thrust (lbf)	8100000		2.4	18.01953929	6047075.472	22.74884854	0.074979829	0.065
19	Time to Start Second Phase (s)	50		2.55	19.1839929	6042830.189	25.62580145	0.074976868	0.065
20	Time to Start Third Phase (s)	100		2.7	20.35316486	6038584.906	28.67800138	0.074973673	0.065
21				2.85	21.52709184	6034339.623	31.90614518	0.074970238	0.065
22				3	22.70581052	6030094.34	35.31093414	0.074966558	0.065
23	<b>Output</b>			3.15	23.88935758	6025849.057	38.89307396	0.074962629	0.065
24	Payload Weight to target velocity (lbm)	1488000		3.3	25.07776967	6021603.774	42.65327475	0.074958446	0.065
25	Final Velocity (ft/s)	7953.470977		3.45	26.27108342	6017358.491	46.59225099	0.074954004	0.065
26	Final Altitude (ft)	191835.1813		3.6	27.46933548	6013113.208	50.71072155	0.0749493	0.065
27	Final Mass (lbm)	1723000		3.75	28.67256244	6008867.925	55.00940966	0.074944328	0.065
28	Final Flight Angle (degrees)	20.1		3.9	29.88080093	6004622.642	59.48904286	0.074939086	0.065
29	Propellant Remaining (%)	0.022763487		4.05	31.09408752	6000377.358	64.15035303	0.074933568	0.065
30				4.2	32.3124588	5996132.075	68.99407634	0.07492777	0.065
31	Calculate			4.35	33.53595131	5991886.792	74.02095326	0.074921689	0.065
32				4.5	34.76460161	5987641.509	79.23172849	0.07491532	0.065
33				4.65	35.99844623	5983396.226	84.62715101	0.074908659	0.065

Figure A.2 DeltaV Spreadsheet

The code for this spreadsheet is contained in the Sheet1 object. The Calculate command button's only function is to call the deltaV method, which runs the calculations to generate the data in columns D through I.

```
Private Sub CommandButton1_Click()  
    deltaV  
End Sub  
  
Sub deltaV()  
    'Initialize Variables  
    'Vehicle weight minus propellant  
    Dim EmptyWeight As Double  
    'Weight of the used propellant  
    Dim PropellantWeight As Double  
    'Weight of vehicle structure, not propulsion system  
    Dim InertWeight As Double  
    'Weight of tanks, engine, unused propellant, etc...  
    Dim SystemWeight As Double  
    Dim Thrust As Double  
    Dim ThrustOne As Double  
    Dim ThrustTwo As Double  
    Dim ThrustThree As Double  
    Dim Isp As Double  
    Dim BurnTime As Double  
    Dim PhaseTwo As Double  
    Dim PhaseThree As Double  
    'Initial mass of the propellant  
    Dim InitialMass As Double  
    'Propellant mass during the simulation  
    Dim Mass As Double  
    'Vehicle velocity at the start of the simulation  
    Dim InitialVelocity As Double  
    'Vehicle velocity during the simulation  
    Dim Velocity As Double  
    'Vehicle height during the simulation  
    Dim Height As Double  
    'Cross sectional area of the vehicle  
    Dim Area As Double  
    'Mass flow rate of the engine  
    Dim MDot As Double  
    'Desired final velocity  
    Dim TargetVelocity As Double  
    Dim dt As Double                'Time step
```

```

'VeLOCITY increment for each time step
Dim dv As Double
'Atmospheric density at a given height
Dim Density As Double
Dim Cd As Double 'Drag coefficient
'Keeps track of time elapsed in the simulation
Dim Time As Double
Dim Counter As Integer
'Drag force on the vehicle
Dim Drag As Double
Dim NumIterations As Integer
Dim Theta As Double 'Flight Angle
Dim dTheta As Double
Dim FinalAngle As Double
Dim StartTurn As Double
Dim EndTurn As Double
Dim Vx As Double
Dim Vy As Double
Dim ErrorOut As Variant
Dim Pressure As Double
Dim Mach As Double

```

```

'Clear Old Data
Worksheets("Sheet1").Range("B24:B30").ClearContents
Worksheets("Sheet1").Range("D:I").ClearContents
Worksheets("Sheet2").Range("A:D").ClearContents
writeVar "D1", "Time (s)"
writeVar "E1", "Velocity (ft/s)"
writeVar "F1", "Mass (lbm)"
writeVar "G1", "Height (ft)"
writeVar "H1", "Density (lbm/ft^3)"
writeVar "I1", "Cd"

```

```

'Get Input Data
EmptyWeight = read("B2")
PropellantWeight = read("B3")
InertWeight = read("B4")
SystemWeight = read("B5")
Isp = read("B6")
BurnTime = read("B7")
InitialVelocity = read("B8")
Height = read("B9")
Area = read("B10")
TargetVelocity = read("B11")
Theta = read("B12")
FinalAngle = read("B13")

```

```

StartTurn = read("B14")
EndTurn = read("B15")
ThrustOne = read("B16")
ThrustTwo = read("B17")
ThrustThree = read("B18")
PhaseTwo = read("B19")
PhaseThree = read("B20")

'First check to see if thrust is enough to lift off the
'ground
If ThrustOne < (EmptyWeight + PropellantWeight) Then
    'If so, set all output to zero. Crash and burn.
    ErrorOut = MsgBox("Insufficient Thrust for Lift-
Off.", vbOKOnly, "Error!")
Else
    'If not, continue with the simulation.
    'Set up first iteration
    NumIterations = 1000
    Mass = EmptyWeight + PropellantWeight
    InitialMass = Mass
    MDot = ThrustOne / Isp
    Thrust = ThrustOne
    dt = BurnTime / NumIterations
    ' Keep track of the x and y components of velocity
    Vx = InitialVelocity * Cos(Theta)
    Vy = InitialVelocity * Sin(Theta)
    Velocity = Sqr((Vx ^ 2) + (Vy ^ 2))
    Time = 0
    Density = 0.075 * Exp((-0.0000074) * (Height ^
1.15))
    Counter = 3

    'Output first iteration
    writeVar "D2", Time
    writeVar "E2", Velocity
    writeVar "F2", Mass
    writeVar "G2", Height
    writeVar "H2", Density
    writeS "Sheet2", "A1", Time
    writeS "Sheet2", "B1", Velocity
    writeS "Sheet2", "C1", Mass
    writeS "Sheet2", "D1", Height

    ' Flight turn calculation. At this point Theta is
    ' the initial flight angle.
    dTheta = -(Theta - FinalAngle) / (EndTurn -
StartTurn)

```

```

Theta = Theta * (3.14159265359 / 180)
' Convert Theta and dTheta to radians
dTheta = dTheta * (3.14159265359 / 180)

'Run a loop to calculate velocity at each time
'increment
Do While Time <= BurnTime
    ' Cd calculation
    Pressure = (2395.3 * Exp((-0.00005) * Height))
    Mach = Velocity / Sqr(1.4 * Pressure / Density)
    If Mach < 0.5 Then
        Cd = 0.065
    ElseIf ((Mach >= 0.5) And (Mach < 1)) Then
        Cd = 0.0825
    ElseIf ((Mach >= 1) And (Mach < 1.4)) Then
        Cd = 0.125
    ElseIf ((Mach >= 1.4) And (Mach < 2)) Then
        Cd = 0.14
    ElseIf ((Mach >= 2) And (Mach < 3)) Then
        Cd = 0.12
    ElseIf (Mach >= 3) Then
        Cd = 0.1
    End If
    writeTo "I", (Counter - 1), Cd

    Drag = Cd * Density * Velocity * Velocity / 2
    dv = ((32.2 * Thrust) / Mass) - (Drag * Area /
Mass) - (32.2 * Sin(Theta))

    Time = Time + dt
    Vx = Vx + (dv * dt * Cos(Theta))
    Vy = Vy + (dv * dt * Sin(Theta))
    Velocity = Sqr((Vx ^ 2) + (Vy ^ 2))
    If (Time >= StartTurn) And (Time <= EndTurn)
Then
        Theta = Theta + (dTheta * dt)
    End If

    Height = Height + (Vy * dt)
    'Catch the error condition where the height
    'becomes negative, e.g. the rocket runs
    'itself into the ground
    If Height < 0 Then
        ErrorOut = MsgBox("Negative value for
height. Simulation Terminated.", vbOKOnly, "Error!")
        Exit Do
    End If

```



```

Density = 0.075 * Exp((-0.0000074) * (Height ^
1.15))

Mass = Mass - (MDot * dt)
If Mass < EmptyWeight Then
    ' Undo calculations because the rocket
    ' can't expel more propellant than it has.
    Mass = Mass + (MDot * dt)
    Height = Height - (Vy * dt)
    If (Time >= StartTurn) And (Time <=
EndTurn) Then
        Theta = Theta - (dTheta * dt)
    End If
    Vx = Vx - (dv * dt * Cos(Theta))
    Vy = Vy - (dv * dt * Sin(Theta))
    Velocity = Sqr((Vx ^ 2) + (Vy ^ 2))
    Exit Do
End If

writeTo "D", Counter, Time
writeTo "E", Counter, Velocity
writeTo "F", Counter, Mass
writeTo "G", Counter, Height
writeTo "H", Counter, Density
writeSTo "Sheet2", "A", (Counter - 1), Time
writeSTo "Sheet2", "B", (Counter - 1), Velocity
writeSTo "Sheet2", "C", (Counter - 1), Mass
writeSTo "Sheet2", "D", (Counter - 1), Height
Counter = Counter + 1

' There are at most three phases of thrust
' levels during the simulation. These
' two if blocks catch when the simulation has
' progressed to the times denoted by
' PhaseTwo and PhaseThree and changes the mass
' flow rate MDot to correspond to the
' new thrust.
If (Time < PhaseThree) And (Time >= PhaseTwo)
Then
    MDot = ThrustTwo / Isp
    Thrust = ThrustTwo
End If
If Time >= PhaseThree Then
    MDot = ThrustThree / Isp
    Thrust = ThrustThree
End If
Loop

```

```

'Output Final Results.  Payload depends on whether
'target velocity was reached
If Velocity < TargetVelocity Then
    writeVar "B24", 0
Else
    writeVar "B24", (EmptyWeight - (InertWeight +
SystemWeight))
End If
writeVar "B25", Velocity
writeVar "B26", Height
writeVar "B27", Mass
Theta = Theta * (180 / 3.14159265359)
writeVar "B28", Theta
writeVar "B29", (100 * ((Mass - EmptyWeight) /
PropellantWeight))
End If
End Sub

```

### A.2.3 Reliability

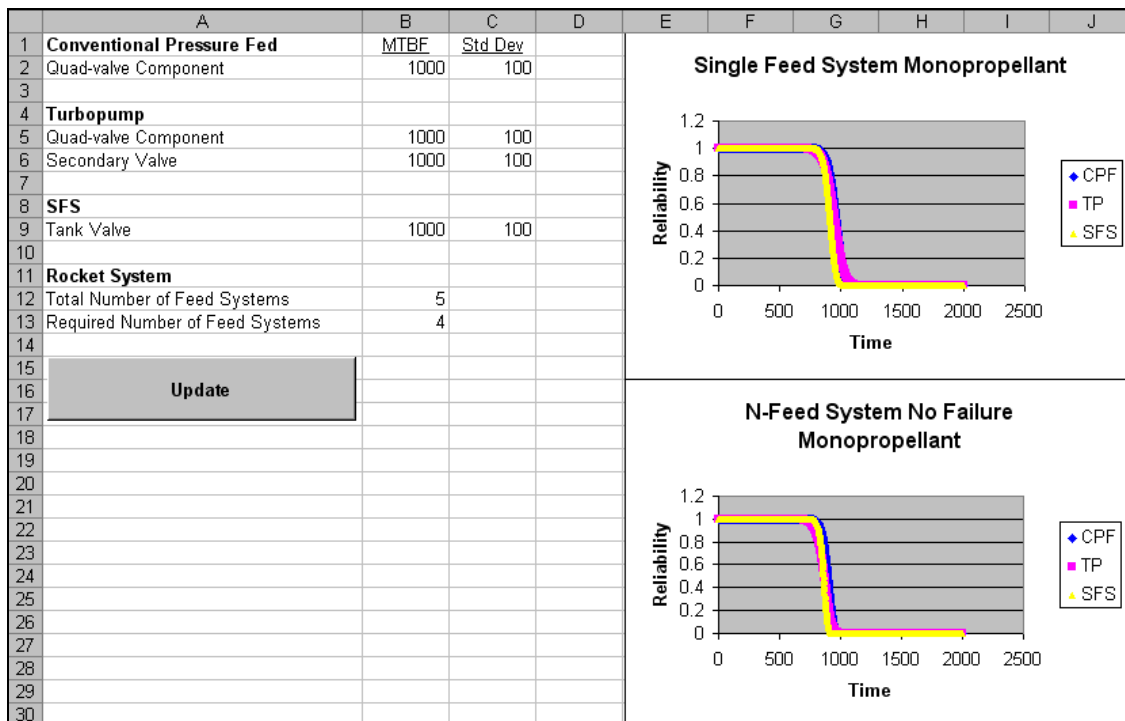


Figure A.3 Reliability Spreadsheet

The code for the reliability spreadsheet is contained in the Sheet1 object and the Module1 object. The Update button code is contained in Sheet1, and the algorithm for that code requires some helper methods: calcProb, factorial, and interProb. Those helper methods are stored in Module1 so they can be accessed from anywhere in the spreadsheet.

```
Private Sub Updater_Click()
    Dim Temp As Double
    Dim Result As Double
    Dim Row As Long
    Dim Zed As Integer
    Dim N As Integer
    Dim M As Integer
    Dim Mono As Double
    Dim Bi As Double
    Dim Answer As Double
    Dim Check As Double
    Dim Col As String

    'Test Code
    'Temp = read("H2", , "Sheet2")
    'writeTo Temp, "D14", , "Sheet2"
    'writeTo 1, "G2", , "Sheet2"
    'Temp = read("H2", , "Sheet2")
    'writeTo Temp, "D15", , "Sheet2"

    ' Put in the time = 0 Prob Tau values for all three
    ' systems
    ' Conventional Pressure Fed
    Temp = read("D", 5, "Sheet2")
    Result = interProb(Temp)
    writeTo Result, "D", 6, "Sheet2"

    ' Turbopump
    ' Quad Valve
    Temp = read("U", 6, "Sheet2")
    Result = interProb(Temp)
    writeTo Result, "U", 7, "Sheet2"
    ' Secondary Valve
    Temp = read("U", 14, "Sheet2")
    Result = interProb(Temp)
    writeTo Result, "U", 15, "Sheet2"
```

```

' SFS
Temp = read("AP", 5, "Sheet2")
Result = interProb(Temp)
writeTo Result, "AP", 6, "Sheet2"

' Set up calculations for N-feed system engine with
' allowed failures.
N = read("B12", , "Sheet1")
M = read("B13", , "Sheet1")
Zed = factorial(N) / ((factorial(M) * factorial(N -
M)))

' Output Status
writeTo "Working", "A19", , "Sheet1"
writeTo "Current Row", "A20", , "Sheet1"

Row = 2
Do While (Row < 2003)
    ' Calculate the Prob Tau values and the probability
    ' of success for an N-feed system engine with
    ' allowed failures.
    ' Conventional Pressure Fed
    Temp = read("F", Row, "Sheet2")
    Result = interProb(Temp)
    writeTo Result, "G", Row, "Sheet2"

    ' Read in P,FS and calculate the probability of
    ' success for monopropellant and bipropellant.
    Mono = read("L", Row, "Sheet2")
    Mono = Mono ^ M
    Bi = Mono ^ 2

    ' Use recursive solution to fill in the data in
    ' each row.
    Answer = calcProb(Zed, Mono)
    writeTo Answer, "P", Row, "Sheet2"
    Answer = calcProb(Zed, Bi)
    writeTo Answer, "Q", Row, "Sheet2"

    ' Turbopump
    ' Quad Valve
    Temp = read("W", Row, "Sheet2")
    Result = interProb(Temp)
    writeTo Result, "X", Row, "Sheet2"
    ' Secondary Valve

```

```

Temp = read("AA", Row, "Sheet2")
Result = interProb(Temp)
writeTo Result, "AB", Row, "Sheet2"

' Follow the same procedure for the other two feed
' system types.
Mono = read("AG", Row, "Sheet2")
Mono = Mono ^ M
Bi = Mono ^ 2

Answer = calcProb(Zed, Mono)
writeTo Answer, "AK", Row, "Sheet2"
Answer = calcProb(Zed, Bi)
writeTo Answer, "AL", Row, "Sheet2"

' SFS
Temp = read("AR", Row, "Sheet2")
Result = interProb(Temp)
writeTo Result, "AS", Row, "Sheet2"

Mono = read("AX", Row, "Sheet2")
Mono = Mono ^ M
Bi = Mono ^ 2

Answer = calcProb(Zed, Mono)
writeTo Answer, "BB", Row, "Sheet2"
Answer = calcProb(Zed, Bi)
writeTo Answer, "BC", Row, "Sheet2"

' Advance to next row
Row = Row + 1
writeTo Row, "B20", , "Sheet1"
Temp = 100 * (Row - 1) / 2002
writeTo Str(Temp) & "%", "B19", , "Sheet1"
Loop
Range("A19:B20").ClearContents

' Go through the results and fill in the Results
' Summary Tables
Check = read("S3", , "Sheet1")

' CPF Single Mono
Row = 2
Col = "L"

```

```

Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "S7", , "Sheet1"

' CPF Single Bi
Row = 2
Col = "M"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "S8", , "Sheet1"

' CPF N Mono
Row = 2
Col = "N"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "S9", , "Sheet1"

' CPF N Bi
Row = 2
Col = "O"
Result = read(Col, Row, "Sheet2")
Do While Result > Check

```

```

    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "S10", , "Sheet1"

' CPF N Mono Failure
Row = 2
Col = "P"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "S11", , "Sheet1"

' CPF N Bi Failure
Row = 2
Col = "Q"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "S12", , "Sheet1"

' TP Single Mono
Row = 2
Col = "AG"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

```

```

        If Row > 2003 Then
            Exit Do
        End If
    Loop
    Result = read("A", Row, "Sheet2")
    writeTo Result, "T7", , "Sheet1"

    ' TP Single Bi
    Row = 2
    Col = "AH"
    Result = read(Col, Row, "Sheet2")
    Do While Result > Check
        Row = Row + 1
        Result = read(Col, Row, "Sheet2")

        If Row > 2003 Then
            Exit Do
        End If
    Loop
    Result = read("A", Row, "Sheet2")
    writeTo Result, "T8", , "Sheet1"

    ' TP N Mono
    Row = 2
    Col = "AI"
    Result = read(Col, Row, "Sheet2")
    Do While Result > Check
        Row = Row + 1
        Result = read(Col, Row, "Sheet2")

        If Row > 2003 Then
            Exit Do
        End If
    Loop
    Result = read("A", Row, "Sheet2")
    writeTo Result, "T9", , "Sheet1"

    ' TP N Bi
    Row = 2
    Col = "AJ"
    Result = read(Col, Row, "Sheet2")
    Do While Result > Check
        Row = Row + 1
        Result = read(Col, Row, "Sheet2")

        If Row > 2003 Then

```



```

        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "T10", , "Sheet1"

' TP N Mono Failure
Row = 2
Col = "AK"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "T11", , "Sheet1"

' TP N Bi Failure
Row = 2
Col = "AL"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "T12", , "Sheet1"

' SFS Single Mono
Row = 2
Col = "AX"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If

```

```

Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "U7", , "Sheet1"

' SFS Single Bi
Row = 2
Col = "AY"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "U8", , "Sheet1"

' SFS N Mono
Row = 2
Col = "AZ"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "U9", , "Sheet1"

' SFS N Bi
Row = 2
Col = "BA"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")

```

```

writeTo Result, "U10", , "Sheet1"

' SFS N Mono Failure
Row = 2
Col = "BB"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "U11", , "Sheet1"

' SFS N Bi Failure
Row = 2
Col = "BC"
Result = read(Col, Row, "Sheet2")
Do While Result > Check
    Row = Row + 1
    Result = read(Col, Row, "Sheet2")

    If Row > 2003 Then
        Exit Do
    End If
Loop
Result = read("A", Row, "Sheet2")
writeTo Result, "U12", , "Sheet1"
End Sub

```

Module1 helper methods.

```

Function calcProb(Zed As Integer, P As Double) As Double
    Dim Z As Integer
    Dim Prob As Double
    Z = Zed

    If (Z > 2) Then
        Prob = calcProb((Z - 1), P)
    Else
        Prob = P
    End If

```

```

    Prob = P + Prob - (P * Prob)

    calcProb = Prob
End Function

Function factorial(Num As Integer) As Integer
    Dim Result As Integer
    Dim i As Integer
    i = Num
    Result = Num
    i = i - 1
    Do While i > 1
        Result = Result * i
        i = i - 1
    Loop
    factorial = Result
End Function

Function interProb(TauIn As Double) As Double
    Dim Check As Double
    Dim Row As Long
    Dim TauOne As Double
    Dim TauTwo As Double
    Dim ProbOne As Double
    Dim ProbTwo As Double

    Row = 2
    TauIn = Abs(TauIn)

    If TauIn > 5 Then
        interProb = 1
    Else
        Check = read("A", Row, "Sheet3")
        Do While (Check < TauIn)
            Row = Row + 1
            Check = read("A", Row, "Sheet3")

            If ((Check = 0) And (Row > 3)) Then
                Row = Row - 1
                Check = TauIn
            End If
        Loop

        ' At this point, unless Check is exactly equal to
        ' TauIn (meaning the loop found the right
        ' Probability value), Row is at the Tau entry that
        ' is greater than the input Tau value. This will

```

```

' be the TauTwo data point. The probability value
' for this Tau value will be the ProbTwo data
' point.
If (Check <> TauIn) Then
    TauTwo = read("A", Row, "Sheet3")
    ProbTwo = read("B", Row, "Sheet3")

    Row = Row - 1
    TauOne = read("A", Row, "Sheet3")
    ProbOne = read("B", Row, "Sheet3")

    Check = ProbOne + ((TauIn - TauOne) * (ProbTwo
- ProbOne) / (TauTwo - TauOne))
Else
    Check = read("B", Row, "Sheet3")
End If

interProb = Check
End If
End Function

```

#### A.2.4 Blowdown

The Blowdown spreadsheet is used to model the vent process of an SFS tank. The spreadsheet is divided into two worksheets, Sim and Data. The Sim worksheet is used to run iterative and analytical models of the vent process, and then compares the results. The Data worksheet is used to run those same models but uses an experimental data set as the basis for the initial conditions. Other inputs are taken from the Sim worksheet, and when the calculations on the data sheet are complete, the results are compared to the actual data.

	A	B	C	D	E	F	G	H	I	J
1	<b>Inputs</b>			Time	Pressure	Temp	Mass			
2	Tank Volume (cubic feet)	1.34		0	1000	500	7.493592		Iterative Solution	
3	Line Inner Diameter (ft)	0.08		0.01	948.3485	492.3386	7.217124		Pressure in psia	
4	Ratio of Specific Heats	1.4		0.02	899.7417	484.8547	6.952904		Temperature in deg R	
5	Specific Gas Constant (ft-lbf/lbm-deg R)	51.5		0.03	853.978	477.543	6.700299		Mass in lbm	
6	Initial Pressure (psia)	1000		0.04	810.8707	470.3981	6.458714			
7	Initial Temperature (deg R)	500		0.05	770.2467	463.4149	6.227589		Vent Time:	
8	External Pressure (psia)	14		0.06	731.9455	456.5885	6.006394		0.91	
9	Final Pressure (psia)	20		0.07	695.8179	449.9143	5.794633			
10	Time Step (s)	0.01		0.08	661.7255	443.3875	5.591836		Final Pressure:	
11				0.09	629.5395	437.004	5.397562		26.04719989	
12				0.1	599.1402	430.7594	5.211392			
13	Go to Iterative Solution			0.11	570.4163	424.6498	5.032932		Final Temperature:	
14				0.12	543.2641	418.6712	4.86181		173.6403239	
15	Go to Analytical Solution			0.13	517.587	412.8199	4.697674			
16				0.14	493.2951	407.0922	4.540191		Final Mass:	
17	Go to Comparison			0.15	470.3045	401.4846	4.389047		0.56204426	
18				0.16	448.5369	395.9939	4.243945			
19				0.17	427.9192	390.6166	4.104602			
20	Area	0.0050265		0.18	408.3832	385.3497	3.970753			
21	Alpha	15.152002		0.19	389.8653	380.1902	3.842145			
22	Initial Mass	7.4935922		0.2	372.3057	375.1351	3.718537			
23				0.21	355.6489	370.1815	3.599705			

Figure A.4 Blowdown Spreadsheet, Sim Sheet Iterative Solution

	A	B	J	K	L	M	N	O	P	Q
1	<b>Inputs</b>			Time	Pressure	Temp	Mass			
2	Tank Volume (cubic feet)	1.34		0	1000	500	7.493592		Analytical Solution	
3	Line Inner Diameter (ft)	0.08		0.01	949.4779	492.6484	7.221174		Pressure in psia	
4	Ratio of Specific Heats	1.4		0.02	901.1596	485.3513	6.956737		Temperature in deg R	
5	Specific Gas Constant (ft-lbf/lbm-deg R)	51.5		0.03	854.9645	478.1086	6.700104		Mass in lbm	
6	Initial Pressure (psia)	1000		0.04	810.8142	470.9204	6.451101			
7	Initial Temperature (deg R)	500		0.05	768.6326	463.7866	6.209557		Time to Unchoke:	
8	External Pressure (psia)	14		0.06	728.3462	456.7073	5.975303		0.921240582	
9	Final Pressure (psia)	20		0.07	689.8834	449.6824	5.748173			
10	Time Step (s)	0.01		0.08	653.1749	442.712	5.528003		Time to Final Pressure:	
11				0.09	618.1535	435.796	5.314631		1.014636385	
12				0.1	584.754	428.9345	5.1079			
13	Go to Iterative Solution			0.11	552.9135	422.1274	4.907652			
14				0.12	522.5708	415.3747	4.713735			
15	Go to Analytical Solution			0.13	493.6666	408.6765	4.525996			
16				0.14	466.1438	402.0327	4.344288			
17	Go to Comparison			0.15	439.9469	395.4434	4.168463			
18				0.16	415.0221	388.9086	3.998377			
19				0.17	391.3176	382.4281	3.83389			
20	Area	0.0050265		0.18	368.7832	376.0022	3.674861			
21	Alpha	15.152002		0.19	347.3704	369.6306	3.521153			
22	Initial Mass	7.4935922		0.2	327.0322	363.3136	3.372633			
23				0.21	307.7234	357.0509	3.229166			

Figure A.5 Blowdown Spreadsheet, Sim Sheet Analytical Solution

	A	B	Q	R	S	T	U	V	W	X	Y	Z
1	<b>Inputs</b>		Time	Pressure	Temp	Mass						
2	Tank Volume (cubic feet)	1.34	0	0	0	0			Comparison of the			
3	Line Inner Diameter (ft)	0.08	0.01	0.119089	0.06294	0.056113			two methods			
4	Ratio of Specific Heats	1.4	0.02	0.157597	0.102418	0.056122			Pressure, Temp, and			
5	Specific Gas Constant (ft-lbf/lbm-deg R)	51.5	0.03	0.115523	0.118444	0.002917			Mass columns all			
6	Initial Pressure (psia)	1000	0.04	0.00697	0.111034	0.117874			represent the			
7	Initial Temperature (deg R)	500	0.05	0.209557	0.080213	0.289537			percent error between			
8	External Pressure (psia)	14	0.06	0.491746	0.026013	0.517624			the analytical results			
9	Final Pressure (psia)	20	0.07	0.852887	0.051526	0.801774			and the iterative			
10	Time Step (s)	0.01	0.08	1.292168	0.152357	1.141551						
11			0.09	1.808623	0.276424	1.536447						
12			0.1	2.401131	0.423664	1.985881						
13			0.11	3.068418	0.594006	2.489198						
14	Go to Iterative Solution		0.12	3.809068	0.787373	3.045675						
15	Go to Analytical Solution		0.13	4.621518	1.003678	3.654519						
16			0.14	5.504069	1.242828	4.314867						
17	Go to Comparison		0.15	6.45489	1.504722	5.025793						
18			0.16	7.472021	1.78925	5.786303						
19			0.17	8.55338	2.096296	6.595342						
20	Area	0.0050265	0.18	9.696768	2.425736	7.451793						
21	Alpha	15.152002	0.19	10.89988	2.777439	8.35448						
22	Initial Mass	7.4935922	0.2	12.1603	3.151265	9.302169						
23			0.21	13.47552	3.547067	10.29357						

Chris Morton:  
% Error = Analytical -  
Simulation / Simulation \*  
100

Figure A.6 Blowdown Spreadsheet, Sim Sheet Comparison of Solutions

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1				Time	Press	T deg F	T deg R		Sim P	Sim P Error	Sim T	Sim T Error		Calc P	Calc P Error	Calc T
2	New Data			0	492.8329	100.3883	560.0583		492.8329	0	560.0583	0		492.8329	0	560.0583
3			0.01	478.6089	100.0911	559.7611		465.8919	2.58039106	550.9554	1.57228482		476.0872	0.51167084	554.5539	
4			0.02	458.033	100.2717	559.9417		440.6314	3.53092146	542.0765	3.18987271		459.9886	0.39682485	549.1302	
5			0.03	457.4638	99.3096	558.9796		416.9339	8.22388111	533.4142	4.56476674		444.5092	2.62860753	543.7857	
6	Clear Data		0.04	446.8073	97.31934	556.9893		394.6906	10.5749143	524.9615	5.71866919		429.6222	3.48699741	538.5189	
7			0.05	444.718	97.04154	556.7115		373.8016	14.3895424	516.7114	7.14214064		415.3023	5.96888842	533.3282	
8			0.06	436.2921	97.37868	557.0487		354.1742	16.662418	508.6575	8.64037577		401.5253	7.05446314	528.2122	
9			0.07	433.1638	97.76831	557.4383		335.7228	19.7716083	500.7936	10.1140658		388.2682	9.10970909	523.1695	
10			0.08	432.0944	96.16655	555.8365		318.3685	23.0759485	493.1137	11.1993402		375.5089	11.4816661	518.1986	
11			0.09	421.1895	95.06205	554.732		302.0379	24.176879	485.612	12.3415847		363.2266	11.7611675	513.2983	
12			0.1	418.81	94.78638	554.4564		286.6633	26.8136823	478.2829	13.6009904		351.4013	13.6777896	508.4671	
13			0.11	414.4862	94.27884	553.9488		272.1818	28.8747723	471.1212	14.7891153		340.0139	15.1110625	503.7038	
14			0.12	409.1045	94.13299	553.803		258.5353	30.5517739	464.1217	16.0128542		329.0463	16.2444771	499.0071	
15			0.13	405.3919	93.4107	553.0807		245.6696	32.4090159	457.2795	17.1055832		318.4812	17.6349137	494.3759	
16			0.14	398.2684	91.87419	551.5442		233.5345	33.4259009	450.5898	18.0256874		308.302	18.2549524	489.8087	
17			0.15	395.5145	92.37175	552.0417		222.0836	35.1905976	444.0482	19.2625535		298.4929	19.6865111	485.3046	
18			0.16	392.1724	91.11788	550.7879		211.2735	36.7059159	437.6502	20.2010512		289.0388	20.9266725	480.8624	
19			0.17	383.6051	89.99069	549.6607		201.064	37.0391503	431.3916	21.1172771		279.9255	21.037482	476.4808	
20			0.18	377.9369	90.42793	550.0979		191.4175	37.8463696	425.2684	22.2886689		271.139	21.6701976	472.1589	
21			0.19	374.4764	88.90668	548.5767		182.2993	38.9943715	419.2766	23.0869028		262.6664	22.6872067	467.8955	
22			0.2	369.5148	86.76274	546.4327		173.6768	39.737201	413.4124	23.7511622		254.4949	23.3385114	463.6896	
23			0.21	364.4119	87.10887	546.7789		165.5198	40.3569006	407.6722	24.8378878		246.6127	23.9024478	459.5401	

Figure A.7 Blowdown Spreadsheet, Data Sheet

This spreadsheet makes use of VBA code and cell formulas to evaluate iterative and analytical simulations of an SFS tank venting process. The results of the iterative solution, shown in Figure A.4, are filled in by the VBA code. The analytical solution, shown in Figure A.5, is generated by the cell formulas. The data in columns L, M, and N

are dependent on time and the inputs on the left side of the worksheet. Lastly, the comparison section, Figure A.6, and the Data sheet, Figure A.7, use the cell formulas to evaluate the percent difference between the analytical and iterative models. The cell formulas used are summarized in Table A.6.

Table A.6 Blowdown Spreadsheet Cell Formulas

Cell	Equation
Sim, B20	$\text{PI}() * ((B3/2)^2)$
Sim, B21	$(B2 / ((B4 - 1) * B20)) * \text{SQRT}((B4 + 1) / (2 * B4 * B5 * 32.2))$
Sim, B22	$B6 * B2 * 144 / (B5 * B7)$
Sim, Column L	$\$B\$6 / ((1 - (D2 * \text{SQRT}(\$B\$7)) / (2 * \$B\$21))^{(-2 * \$B\$4 / (\$B\$4 - 1))})$
Sim, Column M	$\$B\$7 * ((1 - (D2 * \text{SQRT}(\$B\$7)) / (2 * \$B\$21))^{(-2)})$
Sim, Column N	$\$B\$22 / ((1 - (D2 * \text{SQRT}(\$B\$7)) / (2 * \$B\$21))^{(-2 / (\$B\$4 - 1))})$
Sim, P8	$(((((B4 + 1) / 2)^{(B4 / (B4 - 1))}) * (B8 / B6))^{((1 - B4) / (2 * B4))}) - 1) * ((2 * B21) / \text{SQRT}(B7))$
Sim, P11	$((((B9 / B6)^{(1 - B4) / (2 * B4)}) - 1) * ((2 * B21) / \text{SQRT}(B7)))$
Sim, Column S	$100 * \text{ABS}((L2 - E2) / E2)$
Sim, Column T	$100 * \text{ABS}((M2 - F2) / F2)$
Sim, Column U	$100 * \text{ABS}((N2 - G2) / G2)$
Data, Column G	$F2 + 459.67$
Data, Column J	$100 * \text{ABS}((I2 - E2) / E2)$
Data, Column L	$100 * \text{ABS}((K2 - G2) / G2)$
Data, Column O	$100 * \text{ABS}((N2 - E2) / E2)$
Data, Column Q	$100 * \text{ABS}((P2 - G2) / G2)$



The VBA code on the Sim worksheet is activated by a Worksheet\_Change method that starts the iterative model calculations whenever one of the cells in the input block is changed. There are also three “Go to” buttons on this worksheet that simply act as navigation to quickly get to different parts of the worksheet.

```
Private Sub Worksheet_Change(ByVal Target As Range)
    Dim CellAddress As String
    CellAddress = Target.Address

    If Mid(CellAddress, 2, 1) = "B" Then
        If Val(Mid(CellAddress, 4)) < 11 Then
            Call runSim
        End If
    End If
End Sub

Sub runSim()
    ' Read in variables
    Dim Volume As Double
    Dim D As Double
    Dim Area As Double
    Dim Gamma As Double
    Dim R As Double
    Dim PressI As Double
    Dim TempI As Double
    Dim PressE As Double
    Dim PressF As Double
    Dim dt As Double
    ' Program variables
    Dim Time As Double
    Dim Press As Double
    Dim Temp As Double
    Dim Mass As Double
    Dim TSonic As Double
    Dim dP As Double
    Dim mDot As Double
    Dim StopCondition As Double
    Dim ReadIn As Double
    Dim Row As Long
    Dim Continue As Boolean
    Dim Eqn As String
```

```

' Read In Data
Volume = read("B2")
D = read("B3")
Area = 3.14159 * ((D / 2) ^ 2)
Gamma = read("B4")
R = read("B5")
PressI = read("B6")
TempI = read("B7")
PressE = read("B8")
PressF = read("B9")
dt = read("B10")

' Run Iterative Solution
' Clear Old Data from all sections
Range("D:G").ClearContents
writeTo "Time", "D1"
writeTo "Pressure", "E1"
writeTo "Temp", "F1"
writeTo "Mass", "G1"
Range("K:N").ClearContents
writeTo "Time", "K1"
writeTo "Pressure", "L1"
writeTo "Temp", "M1"
writeTo "Mass", "N1"
Range("R:U").ClearContents
writeTo "Time", "R1"
writeTo "Pressure", "S1"
writeTo "Temp", "T1"
writeTo "Mass", "U1"

' Initialize first iteration and output
Row = 2
Time = 0
Press = PressI
Temp = TempI
Mass = (Press * 144 * Volume) / (R * Temp)
StopCondition = ((Gamma + 1) / 2) ^ (Gamma / (Gamma -
1))
writeTo Time, "D", Row
writeTo Press, "E", Row
writeTo Temp, "F", Row
writeTo Mass, "G", Row
Row = Row + 1

' Iterate through and output the results until either

```

```

' the simulation determines that the flow is unchoked,
' or the final pressure is reached.
Continue = True
Do While (Continue = True)
    ' Time
    Time = Time + dt

    ' Pressure
    Press = Press * 144 'Change psi to psf
    TSonic = Temp * (2 / (Gamma + 1))
    dP = -(Gamma * Press * Area / Volume * dt) *
(Sqr((Gamma * (R * 32.2) * TSonic)))
    Press = Press + dP
    Press = Press / 144 'Change back to psi

    ' Mass
    mDot = (Mass * Area / Volume) * (Sqr((Gamma * (R *
32.2) * TSonic)))
    Mass = Mass - (mDot * dt)

    ' Temperature
    Temp = (Press * 144 * Volume) / (Mass * R)

    ' Check for unchoked Flow
    If (Press / PressE) < StopCondition Then
        Continue = False
    End If

    ' Check for Final Pressure
    If Press <= PressF Then
        Continue = False
    End If

    ' Output Results
    writeTo Time, "D", Row
    writeTo Press, "E", Row
    writeTo Temp, "F", Row
    writeTo Mass, "G", Row
    Row = Row + 1
Loop

' Output the final results
Row = Row - 1
writeTo read("D", Row), "I8"
writeTo read("E", Row), "I11"
writeTo read("F", Row), "I14"
writeTo read("G", Row), "I17"

```

```

' Copy Time column to Analytical and Comparison
Sections
Application.ScreenUpdating = False
Columns("D:D").Select
Selection.Copy
Columns("K:K").Select
ActiveSheet.Paste
Columns("R:R").Select
ActiveSheet.Paste
Range("D1").Select
Application.CutCopyMode = False
Application.ScreenUpdating = True

' Fill in cell formulae for Analytical Section
' Pressure Formula
' "(1-(D2*SQR($B$7))/(2*$B$21))"
' Eqn = "=$B$6*((1-(D2*SQR($B$7))/(2*$B$21))^( -
2*$B$4/($B$4-1)))"
Eqn = "=$B$6/((1-(D2*SQR($B$7))/(2*$B$21))^( -
2*$B$4/($B$4-1)))"
Range("L2").Formula = Eqn

' Temperature Formula
' Eqn = "=$B$7/((1-(D2*SQR($B$7))/(2*$B$21))^2)"
Eqn = "=$B$7*((1-(D2*SQR($B$7))/(2*$B$21))^2)"
Range("M2").Formula = Eqn

' Mass Formula
' Eqn = "=$B$22*((1-(D2*SQR($B$7))/(2*$B$21))^( -
2/($B$4-1)))"
Eqn = "=$B$22/((1-(D2*SQR($B$7))/(2*$B$21))^( -2/($B$4-
1)))"
Range("N2").Formula = Eqn

' Copy Paste Equations
Application.ScreenUpdating = False
Range("L2:N2").Select
Selection.Copy
Eqn = "L3:N" + Trim(Str(Row))
Range(Eqn).Select
ActiveSheet.Paste
Range("D1").Select
Application.CutCopyMode = False
Application.ScreenUpdating = True

```

```

' Fill in cell formulae for Comparison Section
' Pressure Formula
Eqn = "=100*ABS((L2-E2)/E2)"
Range("S2").Formula = Eqn

' Temperature Formula
Eqn = "=100*ABS((M2-F2)/F2)"
Range("T2").Formula = Eqn

' Mass Formula
Eqn = "=100*ABS((N2-G2)/G2)"
Range("U2").Formula = Eqn

' Copy Paste Equations
Application.ScreenUpdating = False
Range("S2:U2").Select
Selection.Copy
Eqn = "S3:U" + Trim(Str(Row))
Range(Eqn).Select
ActiveSheet.Paste
Range("D1").Select
Application.CutCopyMode = False
Application.ScreenUpdating = True
End Sub

Private Sub GoAnalytical_Click()
    ActiveWindow.Panes(2).Activate
    Range("N1").Select
End Sub

Private Sub GoCompare_Click()
    ActiveWindow.Panes(2).Activate
    Range("U1").Select
End Sub

Private Sub GoIterative_Click()
    ActiveWindow.Panes(2).Activate
    Range("G1").Select
End Sub

```

The VBA code on the Data sheet is activated by one of the two buttons on the left side of the worksheet. It should be noted that the button with the caption “New Data” has the program name “RunData.” This part of the program expects that the user will have

copied and pasted a data set from another source into columns E and F. The expectation is that the pressure data in column E is in psia, and the temperature data in column F is in degrees Fahrenheit. Conversion of temperature to degrees Rankine is handled through the cell formulas in column G. Also, the time data in column D is set through the cell formulas, adding 0.01 seconds each row, because that was how the available SFS data files (2005 experiment) recorded data. If new data files record time differently, then this column will have to be modified.

```
Private Sub Clear_Click()
    Dim Row As Long
    Dim ToClear As String
    Dim Temp As String

    Row = 2
    Temp = read("D", Row)
    Do While (Temp <> "")
        Row = Row + 1
        Temp = read("D", Row)
    Loop

    ToClear = "D2:Q" + Trim(Str(Row))
    Range(ToClear).ClearContents
End Sub

Private Sub RunData_Click()
    ' Read in variables
    Dim Volume As Double
    Dim D As Double
    Dim Area As Double
    Dim Gamma As Double
    Dim R As Double
    Dim PressI As Double
    Dim TempI As Double
    Dim Presse As Double
    Dim PressF As Double
    Dim Alpha As Double
    ' Program variables
    Dim Check As String
    Dim Eqn As String
    Dim TheCell As String
    Dim Row As Long
```

```

Dim dt As Double
Dim Time As Double
Dim LastTime As Double
Dim Output As Double
Dim Calc As Double
Dim Mass As Double
Dim mDot As Double
Dim TSonic As Double
Dim dP As Double
Dim Press As Double
Dim Temp As Double

' Make sure temperature data is in deg R
Check = read("G2")
If Check = "" Then
    ' Transfer temperature in deg F to deg R
    Eqn = "=F2 + 459.67"
    Range("G2").Formula = Eqn
    Application.ScreenUpdating = False
    Range("G2").Copy

    Row = 3
    Check = read("F", Row)
    Do While Check <> ""
        Eqn = "G" + Trim(Str(Row))
        Range(Eqn).Select
        ActiveSheet.Paste
        Row = Row + 1
        Check = read("F", Row)
    Loop
    Application.CutCopyMode = False
    Application.ScreenUpdating = True
End If

' Read in data from Sim worksheet
Volume = read("B2", , "Sim")
D = read("B3", , "Sim")
Area = read("B20", , "Sim")
Gamma = read("B4", , "Sim")
R = read("B5", , "Sim")
PressI = read("E2")
TempI = read("G2")
PressE = read("B8", , "Sim")
PressF = read("B9", , "Sim")
Alpha = read("B21", , "Sim")

```

```

' Set up the calculations in row 2
Row = 2
' Simulation First Iteration
writeTo PressI, "I", Row
writeTo TempI, "K", Row
Eqn = "=100*ABS((I2-E2)/E2)" ' Press % error equation
Range("J2").Formula = Eqn
Eqn = "=100*ABS((K2-G2)/G2)" ' Temp % error equation
Range("L2").Formula = Eqn
Mass = PressI * 144 * Volume / (R * TempI)
Press = PressI
Temp = TempI

' Analytical Calculation First Row
Time = read("D", Row)
LastTime = Time
Calc = ((Time / (2 * Alpha)) * Sqr(TempI)) + 1
' Pressure Equation
Output = PressI * (Calc ^ ((-2 * Gamma) / (Gamma - 1)))
writeTo Output, "N", Row
' Temperature Equation
Output = TempI / (Calc ^ 2)
writeTo Output, "P", Row
' Percent Error
Eqn = "=100*ABS((N2-E2)/E2)" ' Press % error equation
Range("O2").Formula = Eqn
Eqn = "=100*ABS((P2-G2)/G2)" ' Temp % error equation
Range("Q2").Formula = Eqn

' Loop through each row where there is a data point
' (indicated by a non-null value in the time column)
' and fill in the other columns.
Row = Row + 1
Check = read("D", Row)
Do While Check <> ""
    ' Simulation
    ' Allow for variable dt in the data set,
    ' recalculate it for each point
    Time = read("D", Row)
    dt = Time - LastTime
    LastTime = Time

    ' Pressure
    Press = Press * 144 'Change psi to psf

```



```

    TSonic = Temp * (2 / (Gamma + 1))
    dP = -(Gamma * Press * Area / Volume * dt) *
(Sqr((Gamma * (R * 32.2) * TSonic)))
    Press = Press + dP
    Press = Press / 144 'Change back to psi

    ' Mass (kept track of for temperature calculation)
    mDot = (Mass * Area / Volume) * (Sqr((Gamma * (R *
32.2) * TSonic)))
    Mass = Mass - (mDot * dt)

    ' Temperature
    Temp = (Press * 144 * Volume) / (Mass * R)

    ' Output Results
    writeTo Press, "I", Row
    writeTo Temp, "K", Row

    ' Analytical Calculation
    Calc = ((Time / (2 * Alpha)) * Sqr(TempI)) + 1
    ' Pressure Equation
    Output = PressI * (Calc ^ ((-2 * Gamma) / (Gamma -
1)))
    writeTo Output, "N", Row
    ' Temperature Equation
    Output = TempI / (Calc ^ 2)
    writeTo Output, "P", Row

    ' Percent Errors
    ' Simulation
    ' Press % error equation
    Eqn = "=100*ABS((I" + Trim(Str(Row)) + "-E" +
Trim(Str(Row)) + ")/E2)"
    TheCell = "J" + Trim(Str(Row))
    Range(TheCell).Formula = Eqn
    ' Temp % error equation
    Eqn = "=100*ABS((K" + Trim(Str(Row)) + "-G" +
Trim(Str(Row)) + ")/G2)"
    TheCell = "L" + Trim(Str(Row))
    Range(TheCell).Formula = Eqn

    ' Calculation
    ' Press % error equation
    Eqn = "=100*ABS((N" + Trim(Str(Row)) + "-E" +
Trim(Str(Row)) + ")/E2)"

```

```

TheCell = "O" + Trim(Str(Row))
Range(TheCell).Formula = Eqn
' Temp % error equation
Eqn = "=100*ABS((P" + Trim(Str(Row)) + "-G" +
Trim(Str(Row)) + ")/G2)"
TheCell = "Q" + Trim(Str(Row))
Range(TheCell).Formula = Eqn

' Advance Loop Counter
Row = Row + 1
Check = read("D", Row)

Loop
End Sub

```

### A.2.5 Refill

	A	B	C	D	E	F	G	H	I
1	<b>Inputs</b>			Time (s)	Velocity (ft/s)	Flow Rate (ft <sup>3</sup> /s)	RFS Pressure (psia)	Volume Filled (ft <sup>3</sup> )	
2	Main Tank Pressure (psia)	50							
3	Initial RFS Pressure (psia)	30		0	31.04802517		0	30	0
4	External Pressure (psia)	14.7		0.14196615	30.90008053	2.696540722		32	0.382817498
5	Propellant Density (lbm/ft <sup>3</sup> )	51.5		0.27674193	29.4477471	2.569800721		34	0.729164414
6	RFS Tank Radius (ft)	1.34		0.40611139	27.91996851	2.436476888		36	1.044370093
7	Pipe Inner Diameter (ft)	0.333333		0.5317409	26.30360192	2.295422294		38	1.332742878
8	Change in Height	5		0.65530764	24.58117789	2.145112441		40	1.597807437
9	Acceleration (ft/s <sup>2</sup> )	32.2							
10	Pipe Length (ft)	5		0.77866413	22.72859738	1.983444293		42	1.842478162
11	Total K	3.07		0.90410028	20.71096259	1.807372442		44	2.069188001
12	Dynamic Viscosity (lbf-s/ft <sup>2</sup> )	4.49E-05		1.03485307	18.47427411	1.61218455		46	2.279985626
13				1.17632454	15.92650734	1.389849957		48	2.476609736
14				1.33991433	12.88442736	1.124378399		50	2.660546572
15	<b>Outputs</b>								
16	<i>Option One</i>								
17	Stage I Time	0.655308							
18	Stage II Time	1.991146							
19	Total Refill Time	2.646453							
20									
21	<i>Option Two</i>								
22	Stage I Time	1.339914							
23	Stage II Time	2.108529							
24	Total Refill Time	3.448443							
25									
26	f	0.002604							
27	Initial Reynolds #	368653							
28	Area (ft <sup>2</sup> )	0.087266							
29	Total Volume (ft <sup>3</sup> )	10.07866							
30	Diffuser Volume (ft <sup>3</sup> )	1.475711							
31	Stage II Flow Rate (ft <sup>3</sup> /s)	3.518149							
32	Stage II Velocity (ft/s)	40.31502							
33									

Figure A.8 Refill Spreadsheet

The Refill spreadsheet runs an iterative simulation of an SFS tank refill process.

Two refill options are evaluated: Option One denotes the case where the vent valve is

opened as soon as there is enough liquid in the tank to cover a six-inch long diffuser.

Option Two is the case where the pressure is allowed to equilibrate with the supply tank pressure before the vent valve is opened. These simulations are evaluated by the VBA code, which is activated with a Worksheet\_Change method. The simulation is run every time one of the inputs is changed by the user. All of the simulation code is contained in the Sheet1 object.

```
Private Sub Worksheet_Change(ByVal Target As Range)
    Dim CellAddress
    Dim Column As String
    Dim Temp As String
    Dim Row As Integer
    CellAddress = Target.Address
    Column = Mid(CellAddress, 2, 1)
    Temp = Mid(CellAddress, 4)
    Row = Val(Temp)

    If ((Column = "B") And (Row < 13)) Then
        Refill
    End If
End Sub

Sub Refill()
    ' Input Variables
    Dim PMain As Double
    Dim PRFS As Double
    Dim PExt As Double
    Dim rho As Double
    Dim r As Double
    Dim D As Double
    Dim DeltaZ As Double
    Dim g As Double
    Dim L As Double
    Dim KTot As Double
    Dim mu As Double
    'Program Variables
    Dim StageOneTime As Double
    Dim StageTwoTime As Double
    Dim f As Double
    Dim fTest As Double
    Dim Re As Double
```

```

Dim Velocity As Double
Dim VTot As Double
Dim VFilled As Double
Dim VDiff As Double
Dim VChange As Double
Dim Area As Double
Dim FlowRate As Double
Dim dt As Double
Const Pi As Double = 3.14159265359
Dim Continue As Boolean
Dim Row As Integer
Dim OptionOne As Boolean

' 1. Read in Input Variables
PMain = read("B2")
PRFS = read("B3")
PExt = read("B4")
rho = read("B5")
r = read("B6")
D = read("B7")
DeltaZ = read("B8")
g = read("B9")
L = read("B10")
KTot = read("B11")
mu = read("B12")

'Initialize other variables
StageOneTime = 0
VTot = (4 * Pi / 3) * (r ^ 3)
VFilled = 0
OptionOne = True
' Set up iteration outputs
Range("D:H").ClearContents
Range("B16:B25").ClearContents
writeVar "D1", "Time (s)"
writeVar "E1", "Velocity (ft/s)"
writeVar "F1", "Flow Rate (ft^3/s)"
writeVar "G1", "RFS Pressure (psia)"
writeVar "H1", "Volume Filled (ft^3)"

' 2. Find friction factor f for the system.
Velocity = Sqr((2 * (((PMain - PRFS) * 144 * 32.2) /
rho) + (g * DeltaZ))) / (1 + (f * L / D) + KTot))
Re = (Velocity * rho * D) / (mu * 32.2)
f = (-1.8 * (Log(6.9 / Re))) ^ -2
writeVar "B26", f
writeVar "B27", Re

```

```

writeVar "B29", VTot

' 3. Iterate using Stage I conditions until the filled
' volume is above the diffuser level. In this version
' the diffuser is always assumed to be six inches high.
VDiff = (Pi / 6) * ((3 * r * r) + (0.5 ^ 2)) * 0.5
Area = (Pi / 4) * D * D
Continue = True
writeVar "B28", Area
writeVar "B30", VDiff

' Output Initial Iteration
Range("D2:H2").Select
With Selection
    .HorizontalAlignment = xlCenter
    .MergeCells = True
End With
writeVar "D2", "Option One"
writeVar "D3", StageOneTime
writeVar "E3", Velocity
writeVar "F3", FlowRate
writeVar "G3", PRFS
writeVar "H3", VFilled
Row = 4
' Revision 3 of this code combines the processes of
' both revisions 1 and 2. That means the stop
' conditions on this loop are modified such that both
' options are properly calculated. Option 1 is the
' scenario where stage I lasts until the fluid fills
' above the diffuser level, and Option 2 is the
' scenario where stage I lasts until the pressure in
' the SFS tank equilibrates with the pressure in the
' main tank. Both options have a stage II that fills
' the remainder of the RFS tank up to an ullage volume
' equal to the diffuser volume (six inches deep from
' the edge of the sphere).
Do While (Continue)
    ' 3.1 Velocity calculation
    Velocity = Sqr((2 * (((PMain - PRFS) * 144 * 32.2)
/ rho) + (g * DeltaZ))) / (1 + (f * L / D) + KTot))
    ' 3.2 Flow Rate
    FlowRate = Velocity * Area
    ' 3.3 and 3.4 Time step calculation. Note that the
    ' gamma in this equation is the ratio of the
    ' specific heats of the pressurant gas, which in
    ' this case is assumed to be Helium. Thus gamma is
    ' 5/3. VChange represents the volume that will be

```

```

' filled when the pressure in the RFS tank is
' changed by 2 psi. This number is calculated by
' subtracting V2 (see lab notebook volume 2 pg 10)
' from VTot.
VChange = (VTot - VFilled) - ((VTot - VFilled) /
((1 + (2 / PRFS)) ^ (3 / 5)))
dt = VChange / FlowRate

' 3.5 To update the variables, VFilled has Q*dt
' added to it, and PRFS has 2 added to it. Note
' that all pressures are stored as psia, and
' conversion to psf is done when necessary.
VFilled = VFilled + (FlowRate * dt)
PRFS = PRFS + 2
StageOneTime = StageOneTime + dt

' Check and unmerge any merged rows of cells during
' the output.
If Range(getCell("D", Row) + ":" + getCell("H",
Row)).MergeCells Then
    Range(getCell("D", Row) + ":" + getCell("H",
Row)).Select
        With Selection
            .HorizontalAlignment = xlRight
            .MergeCells = False
        End With
End If
writeVar getCell("D", Row), StageOneTime
writeVar getCell("E", Row), Velocity
writeVar getCell("F", Row), FlowRate
writeVar getCell("G", Row), PRFS
writeVar getCell("H", Row), VFilled
Row = Row + 1

' Option One stop condition. Here the program
' outputs the stage I results, calculates stage II
' time, and then sets the loop to calculate option
' 2.
If ((VFilled > VDiff) And OptionOne) Then
    OptionOne = False
    writeVar "B17", StageOneTime
    ' Merge the cells from D to H for the Option 2
    ' label
    Range(getCell("D", Row) + ":" + getCell("H",
Row)).Select
        With Selection
            .HorizontalAlignment = xlCenter

```

```

        .MergeCells = True
    End With
    writeVar getCell("D", Row), "Option Two"
    Row = Row + 1

    ' Stage II calculation
    Velocity = Sqr((2 * (((PMain - PExt) * 144 *
32.2) / rho) + (g * DeltaZ))) / (1 + (f * L / D) + KTot))
    FlowRate = Velocity * Area
    ' Note that the stage II flow rate and velocity
    ' will be the same for both options because the
    ' pressure difference between the main tank and
    ' the external conditions are the same.
    writeVar "B31", FlowRate
    writeVar "B32", Velocity
    StageTwoTime = (VTot - VFilled - VDiff) /
FlowRate
    writeVar "B18", StageTwoTime
    writeVar "B19", (StageOneTime + StageTwoTime)
End If

    ' Continue loop with option 2
    If (((PRFS + 2) > PMain) And Not OptionOne) Then
        Continue = False
    End If

    If VFilled > VTot Then
        VFilled = VFilled - (FlowRate * dt)
        StageOneTime = StageOneTime - dt
        Continue = False
    End If

Loop

    ' 4. For Stage II it is assumed that the flow rate is
    ' constant because the pressure difference between the
    ' main tank and outside remains constant.
    ' 4.1 Flow velocity calculation using PExt
    Velocity = Sqr((2 * (((PMain - PExt) * 144 * 32.2) /
rho) + (g * DeltaZ))) / (1 + (f * L / D) + KTot))
    FlowRate = Velocity * Area
    ' 4.2 Calculate in one time step
    StageTwoTime = (VTot - VFilled) / FlowRate

    ' Output results
    writeVar "B22", StageOneTime
    writeVar "B23", StageTwoTime

```

```

writeVar "B24", (StageOneTime + StageTwoTime)
Range("A1").Select
End Sub

```

### A.3 Data Analysis Spreadsheets

#### A.3.1 Fluctuation

The Fluctuation spreadsheet was used to analyze the outflow pressure data from the Baseline tests. The spreadsheet went through two revisions. The first revision was a basic program that let the user select two time ranges by row number. The first time range specified the data used to calculate the average outflow pressure, and the second range was used to examine pressure trends at a certain section in the data.

The spreadsheet was used to examine the drain transitions from one run tank to the next for all of the baseline tests. For a given test, the same average outflow pressure was used, and the time range defining the fluctuation around a given drain transition was arrived at by examining a pressure vs time chart of the data.

The program assumes that the time (DAQ reference) and outflow data are copied to columns A and B, respectively, of Sheet 1, as shown in Figure A.9.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	DAQ Elaps	Water Outlet Press.			Press Avg	Fluct Max	Fluct Min	Fluct %	Duration	P Start	P End	Fluct Start	Fluct End
2	0.282	-2.64859			122.8815	124.0554	114.4457	6.865035	2.924	564	2096	1833	1846
3	0.499	-2.67643											
4	0.717	-2.67755			Time	393.312							
5	0.921	-2.68572			Row	1846							
6	1.137	-2.6968											
7	1.355	-2.67465											
8	1.558	-2.68656											
9	1.862	-2.69721											
10	2.043	-2.65816											

Figure A.9 Fluctuation Spreadsheet First Revision



The program runs every time one of the Start or End entries in row 2 is modified by using the following macro.

```
Private Sub Worksheet_Change(ByVal Target As Range)
    Dim CellAddress
    Dim Row As Long
    Dim ReadIn As Double
    Dim StartAt As Long
    Dim EndAt As Long
    Dim Duration As Double
    Dim Eqn As String
    Dim AvgP As Double
    Dim Max As Double
    Dim Min As Double
    Dim Percent As Double
    Dim RunSub As Boolean

    CellAddress = Target.Address
    RunSub = (CellAddress = "$J$2") Or (CellAddress =
"$K$2") Or (CellAddress = "$L$2") Or (CellAddress = "$M$2")

    If (RunSub = True) Then
        ' Find Pressure Average from inputs in E1 and E2
        StartAt = read("J2")
        EndAt = read("K2")
        Eqn = "=AVERAGE(B" + Trim(Str(StartAt)) + ":B" +
Trim(Str(EndAt)) + ")"
        Worksheets("Sheet1").Range("E2").Formula = Eqn

        ' Find Fluctuation Max and Min from inputs in H1
and H2
        StartAt = read("L2")
        EndAt = read("M2")
        Eqn = "=MAX(B" + Trim(Str(StartAt)) + ":B" +
Trim(Str(EndAt)) + ")"
        Worksheets("Sheet1").Range("F2").Formula = Eqn
        Eqn = "=MIN(B" + Trim(Str(StartAt)) + ":B" +
Trim(Str(EndAt)) + ")"
        Worksheets("Sheet1").Range("G2").Formula = Eqn

        ' Calculate Fluctuation % Based on the largest
deviation from the mean.
        AvgP = read("E2")
        Max = read("F2")
        Min = read("G2")
        Max = Abs(AvgP - Max)
```

```

Min = Abs(AvgP - Min)
If (Max > Min) Then
    Percent = (Max / AvgP) * 100#
Else
    Percent = (Min / AvgP) * 100#
End If
writeTo Percent, "H2"

' Find Fluctuation Duration from inputs in H1 and
H2
Duration = read("A", EndAt) - read("A", StartAt)
writeTo Duration, "I2"
End If

If (CellAddress = "$F$4") Then
    Max = read("F4")
    Row = 2
    ReadIn = read("A", Row)
    Do While (ReadIn < Max)
        Row = Row + 1
        ReadIn = read("A", Row)
    Loop
    writeTo Row, "F5"
End If
End Sub

```

The second revision of the code was made to be more thorough and to check each data point of outflow pressure against the calculated mean outflow pressure. The input section of this spreadsheet is shown in Figure A.10.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Sequence	Water Outlet Press.			Data Filename:		SFS 12-6-2007_6.txt			Overlap	0		
2	0	-0.90789			First Cycle Average Pressure:		119.8654 psig			Drain	55		
3	0.516	124.0859			Pressure Variation Threshold:		2 %						
4	0.725	120.3645											
5	0.946	120.0355											
6	1.163	120.6645											
7	1.385	120.4819											
8	1.605	120.3354			Pressure Variation Events that deviate by more than the threshold percentage from the mean outflow								
9	1.812	120.2115			Event	Start	End	Duration	Max Var	Max %			
10	2.022	120.4314			1	164.388	164.648	0.26	116.74	2.607438			
11	2.247	120.1834			2	273.543	273.799	0.256	110.9046	7.475713			
12	2.473	120.3532			3	328.295	0	-328.295	-2.56584	102.1406			
13	2.693	120.4369											
14	2.9	120.3769											

Figure A.10 Fluctuation Spreadsheet Second Revision

In this revision of the code, the data is still expected to be in columns A and B; only the time data is expected to be in reference to the Sequence. When the start button is clicked, the program calculates the average of the outflow pressure and compares each data point to that value. Any outflow pressure with a percent error greater than the threshold value (default of 2%) is logged as a pressure event. The way the program was set up caused an event to be recorded at the end of every Sequence, when outflow pressure dropped back to zero. Thus, the last event in every report was always omitted from analysis. The code for these calculations is as follows.

```
Private Sub Start_Click()  
    'Timing Inputs  
    Dim Drain As Double  
    Dim Overlap As Double  
    'Counters  
    Dim StartSequence As Long  
    Dim RowCounter As Long  
    Dim OutRow As Long  
    Dim EventCounter As Long  
    Dim EventStart As Long  
    'I/O Variables  
    Dim ReadIn As Double  
    'Calculation Variables  
    Dim Threshold As Double  
    Dim Eqn As String  
    Dim AverageP As Double  
    Dim FailTime As Double  
    Dim Time As Double  
    Dim MaxVar As Double  
    Dim MaxPercent As Double  
    Dim Temp As Double  
    Dim Continue As Boolean  
  
    'Read in data from the Input sheet  
    Threshold = read("H3")  
    ' Set Default Threshold value in case of null entry.
```

```

If Threshold <= 0 Then
    Threshold = 2#
End If
Drain = read("L2")
Overlap = read("L1")

' Clear old data import inputs to the Report sheet.
Worksheets("Report").Range("H2").ClearContents
Worksheets("Report").Range("E10:J1000").ClearContents

' Pause worksheet updating and set the active sheet to
Data. This is to simplify the
' code and not require the read function to always
reference the Data sheet.
Application.ScreenUpdating = False

' Assume that the data in columns A and B are formatted
to be at the start of the
' AutoSequence. Skipping the first two data points.
StartSequence = 4

' Now look for the row that corresponds to the end of
the drains (two cycles).
' This will be at the same place as the start of the
(hypothetical) seventh drain.
' Using the stage start time equation with N = 7.
Time = (7 - 1) * (Drain - Overlap)
RowCounter = StartSequence
ReadIn = read("A", RowCounter)
Do While (ReadIn < Time)
    RowCounter = RowCounter + 1
    ReadIn = read("A", RowCounter)
Loop
' Omit last two rows.
RowCounter = RowCounter - 2

' At this point RowCounter defines the row where the
drain data ends. A formula is
' constructed to average the outflow pressure data for
all cycles, and that formula
' is placed on the Report sheet. Because the outflow
pressure does not instantly
' rise to its nominal value the first two data points
after SequenceStart are omitted,

```

' and likewise at the end because the pressure does not fall off instantly.

```
Eqn = "=AVERAGE(B" + Trim(Str(StartSequence)) + ":B" + Trim(Str(RowCounter)) + ")"
```

```
StartSequence = StartSequence - 2
```

```
Worksheets("Report").Range("H2").Formula = Eqn
```

```
AverageP = read("H2")
```

' Reset RowCounter to the StartSequence and go through the entire data set of the

' Auto Sequence looking for pressure variation events in the outflow pressure. An

' "event" is defined as a pressure reading that is outside the threshold range

' (default of plus or minus 2% of the average pressure). For each event the following

' data are displayed on the Report sheet:

' 1) Max variation intensity in pressure and % of mean (the data point furthest removed

' from the mean pressure).

' 2) The start time of the event (in Sequence Time).

' 3) The end time of the event (in Sequence Time).

' 4) The duration of the event in seconds.

' 5) A note if the event is within plus or minus 5 seconds of the failure injection.

' 6) If 5 is yes, another note specifying if the even is before or after the failure

' injection.

```
RowCounter = StartSequence + 2
```

```
OutRow = 10
```

```
EventCounter = 1
```

```
Continue = True
```

```
Do While (Continue = True)
```

```
    ReadIn = read("B", RowCounter) 'Outflow Pressure Data
```

```
    Temp = perError(AverageP, ReadIn)
```

' Event Condition: If the percent error between an outflow pressure data point

' and the average pressure calculated from cycle 1 is greater than the threshold

' percent value (default of 2%) then a variation event exists.

```
    If Temp > Threshold Then
```

```
        EventStart = RowCounter
```

```
        MaxVar = ReadIn
```

```
        MaxPercent = Temp
```

```

        ' Need to advance RowCounter to the end of the
event (or the end of the
        ' Sequence), updating MaxVar as necessary.
RowCounter = RowCounter + 1
ReadIn = read("B", RowCounter)
Temp = perError(AverageP, ReadIn)
Do While ((Temp > Threshold) And (read("A",
RowCounter) <> 0))
    If Temp > MaxPercent Then
        MaxVar = ReadIn
        MaxPercent = Temp
    End If
    RowCounter = RowCounter + 1
    ReadIn = read("B", RowCounter)
    Temp = perError(AverageP, ReadIn)
Loop

```

```

        ' Output the data to the Report sheet
        ' Event Number
writeTo EventCounter, "E", OutRow, "Report"
        ' Start Time
Time = read("A", EventStart)
writeTo Time, "F", OutRow, "Report"
        ' End Time
Temp = read("A", RowCounter)
writeTo Temp, "G", OutRow, "Report"
        ' Duration
writeTo (Temp - Time), "H", OutRow, "Report"
        ' Max Variation (Pressure)
writeTo MaxVar, "I", OutRow, "Report"
        ' Max Variation (Percent of mean)
writeTo MaxPercent, "J", OutRow, "Report"

EventCounter = EventCounter + 1
OutRow = OutRow + 1
End If

```

```

        ' Advance RowCounter to the next row and check to
see if the Sequence has ended
        ' (Sequence Time will reset to zero). Also,
because of tank 3 refills the
        ' outflow pressure will drop to zero before the
Sequence ends. To prevent an
        ' event from being recorded in this part of the
data the loop will also end if

```

```

        ' the time is greater or equal to the time when a
seventh drain would start.
        ' The program will assume at this point that the
three tanks only run for a
        ' maximum of 2 cycles.
        RowCounter = RowCounter + 1
        ReadIn = read("A", RowCounter)
        Time = (7 - 1) * (Drain - Overlap)
        If (ReadIn = 0) Or (ReadIn >= Time) Then
            Continue = False
        End If
    Loop

    Worksheets("Report").Activate
    Application.ScreenUpdating = True
End Sub

' ArgOne should be the accepted value and ArgTwo the test
value.
Function perError(ArgOne As Double, ArgTwo As Double)
    Dim Result As Double
    Result = (ArgOne - ArgTwo) / ArgOne
    Result = (Abs(Result)) * 100#
    perError = Result
End Function

```

### A.3.2 Variations

The Variations spreadsheet was used to analyze the Fail-Operational test data.

The code is very similar to that of Fluctuations rev 2, as that spreadsheet was a derivative of Variations. The spreadsheet is broken into three worksheets. The “Input” worksheet is shown in Figure A.11. When the program finishes running, the output data is sent to the “Report” worksheet in the same format as shown in the figures of Section C.2.

Lastly, the “Data” worksheet contains all of the data from the test data file, excluding all header lines except column descriptions, put into Excel format.

	A	B	C	D	E	F	G	H
1	<u>Timing Inputs</u>			<u>Failure Settings</u>			<u>Filename</u>	
2	Drain	55		Type	Stop		SFS 12-6-2007_6.txt	
3	Vent	22		Valve	Vent			
4	Fill	25		Tank	2		<u>Variation Threshold</u>	
5	Fill/Vent	15		Cycle	2		2 %	
6	Press	7.5		Stage	Fill			
7	Overlap	0						
8								
9								
10								
11	Start							
12								
13								

Figure A.11 Variations Spreadsheet

```

Private Sub Worksheet_Change(ByVal Target As Range)
    Dim CellAddress As String
    Dim CheckInput As String
    Dim CheckNum As Long
    Dim Response
    Dim ErrorSet As Boolean

    CellAddress = Target.Address
    ErrorSet = False

    If Mid(CellAddress, 2, 1) = "E" Then
        Select Case Val(Mid(CellAddress, 4))
            Case 2
                CheckInput = read("E2")
                If (CheckInput <> "Start" And CheckInput <>
"Stop") Then
                    ErrorSet = True
                End If
            Case 3
                CheckInput = read("E3")
                If (CheckInput <> "Drain" And CheckInput <>
"Vent" And CheckInput <> "Fill" And CheckInput <> "Press")
Then
                    ErrorSet = True
                End If
        End Select
    End If

```



```

Case 4
    CheckNum = read("E4")
    If (CheckNum < 1 And CheckNum > 3) Then
        ErrorSet = True
    End If
Case 5
    CheckNum = read("E5")
    If (CheckNum < 1) Then
        ErrorSet = True
    End If
Case 6
    CheckInput = read("E6")
    If (CheckInput <> "Drain" And CheckInput <>
"Vent" And CheckInput <> "Fill" And CheckInput <> "Press")
Then
        ErrorSet = True
    End If
Case Else
    ErrorSet = False
End Select

If ErrorSet = True Then
    Response = MsgBox("Invalid Input.  Macros may not
work correctly.", vbOKOnly, "Error!")
End If
    'If Val(Mid(CellAddress, 4)) < 2 Then
    '    Call runSim
    'End If
End If
End Sub

```

```

Private Sub Start_Click()
    'Timing Inputs
    Dim Drain As Double
    Dim Vent As Double
    Dim Fill As Double
    Dim FillVent As Double
    Dim Press As Double
    Dim Overlap As Double
    'Failure Settings
    Dim FailType As String
    Dim FailValve As String
    Dim FailTank As Long
    Dim FailCycle As Long
    Dim FailStage As String
    'Counters

```

```

Dim StartSequence As Long
Dim RowCounter As Long
Dim OutRow As Long
Dim EventCounter As Long
Dim EventStart As Long
'I/O Variables
Dim ReadIn As Double
'Calculation Variables
Dim Threshold As Double
Dim Eqn As String
Dim AverageP As Double
Dim FailTime As Double
Dim Time As Double
Dim MaxVar As Double
Dim MaxPercent As Double
Dim Temp As Double
Dim Continue As Boolean

'Read in data from the Input sheet
Drain = read("B2", , "Input")
Vent = read("B3", , "Input")
Fill = read("B4", , "Input")
FillVent = read("B5", , "Input")
Press = read("B6", , "Input")
Overlap = read("B7", , "Input")
FailType = read("E2", , "Input")
FailValve = read("E3", , "Input")
FailTank = read("E4", , "Input")
FailCycle = read("E5", , "Input")
FailStage = read("E6", , "Input")
Threshold = read("G5", , "Input")

' Clear old data import inputs to the Report sheet.
Worksheets("Report").Range("C2").ClearContents
Worksheets("Report").Range("D2:D4").ClearContents
Worksheets("Report").Range("A12:H1000").ClearContents
Worksheets("Report").Range("H2:H7").ClearContents
Worksheets("Report").Range("K2:K7").ClearContents
' Set Default Threshold value in case of null entry.
If Threshold <= 0 Then
    Threshold = 2#
End If
writeTo Threshold, "D3", , "Report"
writeTo Drain, "H2", , "Report"
writeTo Vent, "H3", , "Report"

```

```

writeTo Fill, "H4", , "Report"
writeTo FillVent, "H5", , "Report"
writeTo Press, "H6", , "Report"
writeTo Overlap, "H7", , "Report"
writeTo FailType, "K2", , "Report"
writeTo FailValve, "K3", , "Report"
writeTo FailTank, "K4", , "Report"
writeTo FailCycle, "K5", , "Report"
writeTo FailStage, "K6", , "Report"
writeTo read("G2", , "Input"), "D1", , "Report"

```

'Use the findTime function to calculate the time at which the failure occurs.

```

FailTime = findTime(Drain, Vent, Fill, FillVent, Press,
Overlap, FailType, FailValve, FailTank, FailCycle,
FailStage)

```

```

writeTo FailTime, "D4", , "Report"

```

' Pause worksheet updating and set the active sheet to Data. This is to simplify the

' code and not require the read function to always reference the Data sheet.

```

Application.ScreenUpdating = False
Worksheets("Data").Activate

```

' First find the row on the Data sheet where the Auto Sequence begins. The program

' expects a File Sequence to be before the Auto Sequence, when the System Refill was

' run.

```

StartSequence = 2

```

```

ReadIn = read("B", StartSequence)

```

```

Do While (ReadIn = 0)

```

```

    StartSequence = StartSequence + 1

```

```

    ReadIn = read("B", StartSequence)

```

```

Loop

```

' At this point StartSequence has reached the row where the File Sequence data is

' expected to be.

```

Do While (ReadIn > 0)

```

```

    StartSequence = StartSequence + 1

```

```

    ReadIn = read("B", StartSequence)

```

```

Loop

```

' At this point StartSequence has reached the row where the File Sequence data is

```

' expected to end. The data in the Sequence Time
column will be zero from here until
' the start of the Auto Sequence.
Do While (ReadIn = 0)
    StartSequence = StartSequence + 1
    ReadIn = read("B", StartSequence)
Loop
StartSequence = StartSequence - 1

' Now look for the row that corresponds to the end of
the first three drains (cycle 1).
' This will be at the same place as the start of the
fourth drain. Using the stage
' start time equation from the findTime function with N
= 4.
Time = (4 - 1) * (Drain - Overlap)
RowCounter = StartSequence
ReadIn = read("B", RowCounter)
Do While (ReadIn < Time)
    RowCounter = RowCounter + 1
    ReadIn = read("B", RowCounter)
Loop

' At this point RowCounter defines the row where the
first datum from the fourth
' drain is recorded, and the end of the data from cycle
1. A formula is constructed
' to average the outflow pressure data for cycle 1, and
that formula is placed on the
' Report sheet. Because the outflow pressure does not
instantly rise to its nominal
' value the first two data points after SequenceStart
are omitted.
StartSequence = StartSequence + 2
Eqn = "=AVERAGE(Data!Y" + Trim(Str(StartSequence)) +
":Y" + Trim(Str(RowCounter)) + ")"
StartSequence = StartSequence - 2
Worksheets("Report").Range("D2").Formula = Eqn
AverageP = read("D2", , "Report")

' Reset RowCounter to the StartSequence and go through
the entire data set of the
' Auto Sequence looking for pressure variation events
in the outflow pressure. An
' "event" is defined as a pressure reading that is
outside the threshold range

```

```

' (default of plus or minus 2% of the average
pressure). For each event the following
' data are displayed on the Report sheet:
' 1) Max variation intensity in pressure and % of mean
(the data point furthest removed
' from the mean pressure).
' 2) The start time of the event (in Sequence Time).
' 3) The end time of the event (in Sequence Time).
' 4) The duration of the event in seconds.
' 5) A note if the event is within plus or minus 5
seconds of the failure injection.
' 6) If 5 is yes, another note specifying if the even
is before or after the failure
' injection.
RowCounter = StartSequence + 2
OutRow = 12
EventCounter = 1
Continue = True

Do While (Continue = True)
    ReadIn = read("Y", RowCounter) 'Outflow Pressure
Data
    Temp = perError(AverageP, ReadIn)
    ' Event Condition: If the percent error between an
outflow pressure data point
    ' and the average pressure calculated from cycle 1
is greater than the threshold
    ' percent value (default of 2%) then a variation
event exists.
    If Temp > Threshold Then
        EventStart = RowCounter
        MaxVar = ReadIn
        MaxPercent = Temp

        ' Need to advance RowCounter to the end of the
event (or the end of the
        ' Sequence), updating MaxVar as necessary.
        RowCounter = RowCounter + 1
        ReadIn = read("Y", RowCounter)
        Temp = perError(AverageP, ReadIn)
        Do While ((Temp > Threshold) And (read("B",
RowCounter) <> 0))
            If Temp > MaxPercent Then
                MaxVar = ReadIn
                MaxPercent = Temp
            End If
            RowCounter = RowCounter + 1

```

```

        ReadIn = read("Y", RowCounter)
        Temp = perError(AverageP, ReadIn)
Loop

    ' Output the data to the Report sheet
    ' Event Number
    writeTo EventCounter, "A", OutRow, "Report"
    ' Start Time
    Time = read("B", EventStart)
    writeTo Time, "B", OutRow, "Report"
    ' End Time
    Temp = read("B", RowCounter)
    writeTo Temp, "C", OutRow, "Report"
    ' Duration
    writeTo (Temp - Time), "D", OutRow, "Report"
    ' Max Variation (Pressure)
    writeTo MaxVar, "E", OutRow, "Report"
    ' Max Variation (Percent of mean)
    writeTo MaxPercent, "F", OutRow, "Report"
    ' Near Failure
    If ((Abs(Time - FailTime)) <= 5#) Then
        writeTo "YES", "G", OutRow, "Report"
    ' Before or After
        If (Time < FailTime) Then
            writeTo "Before", "H", OutRow, "Report"
        Else
            writeTo "After", "H", OutRow, "Report"
        End If
    End If

    EventCounter = EventCounter + 1
    OutRow = OutRow + 1
End If

    ' Advance RowCounter to the next row and check to
    see if the Sequence has ended
    ' (Sequence Time will reset to zero). Also,
    because of tank 3 refills the
    ' outflow pressure will drop to zero before the
    Sequence ends. To prevent an
    ' event from being recorded in this part of the
    data the loop will also end if
    ' the time is greater or equal to the time when a
    seventh drain would start.
    ' The program will assume at this point that the
    three tanks only run for a

```

```

        ' maximum of 2 cycles.
        RowCounter = RowCounter + 1
        ReadIn = read("B", RowCounter)
        Time = (7 - 1) * (Drain - Overlap)
        If (ReadIn = 0) Or (ReadIn >= Time) Then
            Continue = False
        End If
    Loop

    Worksheets("Report").Activate
    Application.ScreenUpdating = True
End Sub

Function findTime(D As Double, V As Double, F As Double, FV
As Double, P As Double, O As Double, FType As String, Valve
As String, Tank As Long, Cycle As Long, Stage As String)
    Dim N As Long
    Dim TimeStart As Double
    Dim TimeEnd As Double

    ' Need to find N, the drain number that the failure
    occurs in.
    N = ((Cycle - 1) * 3) + Tank

    ' Time for the start and end of a stage depends on N
    and the timing inputs, per the following equations.
    Select Case Stage
        Case "Drain"
            TimeStart = (N - 1) * (D - O)
            TimeEnd = (N * D) - ((N - 1) * O)
        Case "Vent"
            TimeStart = (N * D) - ((N - 1) * O)
            TimeEnd = (N * D) - ((N - 1) * O) + V
        Case "Fill"
            If Valve = "Fill" Then
                TimeStart = (N * D) - ((N - 1) * O) + V
            Else
                TimeStart = (N * D) - ((N - 1) * O) + V +
(F - FV)
            End If
            ' End time of the Fill Stage is the same no
            matter if it is the Fill or Vent
            ' valve that has the Stop failure
            TimeEnd = (N * D) - ((N - 1) * O) + V + F
    End Select
End Function

```

```

        Case "Press"
            TimeStart = (N * D) - ((N - 1) * O) + V + F
            TimeEnd = (N * D) - ((N - 1) * O) + V + F + P
        Case Else
    End Select

    If FType = "Start" Then
        findTime = TimeStart
    Else
        findTime = TimeEnd
    End If

End Function

' ArgOne should be the accepted value and ArgTwo the test
value.
Function perError(ArgOne As Double, ArgTwo As Double)
    Dim Result As Double
    Result = (ArgOne - ArgTwo) / ArgOne
    Result = (Abs(Result)) * 100#
    perError = Result
End Function

```



## APPENDIX B

### TEST PROCEDURE

#### B.1 Test Operation

This section describes how the 2007 experiment was run and how the hardware and software components of the test bed affected the operation of the tests. The safety concerns that were identified for the testing are discussed first, followed by the procedures used to run the test bed and conduct the various testing.

##### B.1.1 Safety Concerns

When the SFS test bed was located at NASA MSFC, a requirement of the testing was that a formal Test Readiness Review (TRR) be conducted. It was decided to hold an informal TRR and invite UAH professors and MSFC employees to inspect the test bed and provide an outside opinion of any problems that should be addressed. Several issues were found prior to and discussed during the TRR, many of which were the same as in the 2005 experiment. The solutions implemented for these issues are as follows.

Air System over-pressurization. The main air supply tank at the PRC was rated at 3500 psig, and the compressor that supplied the tank was capable of pressurizing it that high. In order to avoid system over-pressurization, the constraints on the test bed had to be well defined. The primary constraint, as far as the Air subsystem was concerned, was

the COPV loan agreement with UAH that the SFS run tanks were not to be pressurized significantly higher than 120 psig. This limitation was put in place because of the more limited safety equipment available at the UAH PRC. When the test bed was located at MSFC, there were concrete blast barriers placed around the test bed. It was not possible to place such barriers at the PRC, so it was decided to run the experiment at a lower operating pressure.

To ensure low pressure operation, a relief valve was obtained with a set pressure of 150 psig. This relief valve was chosen because it was readily available, had the right set pressure range, and came with a 1" tube size that could be easily installed on the test bed. The capacity of the relief valve was 440 scfm. In order to keep the flow rate through the Air subsystem below the relief valve's capacity, calculations were done for various supply tank pressures to find an orifice that would reduce the flow rate. Assuming a tank pressure of 2000 psig, it was found that an orifice with a Cv of approximately 0.34 would suffice to bring the flow rate through the Air subsystem to below the relief valve's capacity.<sup>32</sup> While an orifice was not available, a needle valve with the appropriate Cv was found and installed upstream of the dome regulator. Using a valve also provided an extra way of manually controlling air flow to the test bed.

Water System over-pressurization. The accumulator tank was the component of the water system that had the most sensitive pressure constraints. This tank was a commercial off-the-shelf product typically used for well water systems and was only rated to 100 psig. The tank was adapted for use with the SFS. As water flowed into the tank, an air filled bladder was compressed. The pressure from this bladder could then push water out of the tank when required. Since the tank is at least partially filled with

air, over-pressurization is a significant safety concern as a rupture of the accumulator tank would most likely be catastrophic.

To resolve this issue, two actions were taken. First, the specifications for the main supply pump were obtained and calculations were done to find the pressure that would be produced at the pumps maximum output and zero flow rate (the case where the pump is set to run as fast as possible while all the Fill valves were closed), the dead-head pressure. These calculations<sup>30</sup> showed that the pump was incapable of producing water pressure anywhere near 100 psig, and testing of the water system verified this conclusion.

The second corrective action was to modify the ICM to create a programmatic “relief valve” for the water system. Obtaining a real relief valve was abandoned since none were readily available and the pump calculations showed it wasn’t critical to have. The programmatic relief valve checks the data read from the water inlet pressure transducer. If this value is ever over 75 psig, the program sets a special case of the emergency stop. In the special case, all of the Drain and Fill valves are left open while the program halts operation. The idea behind this was that if the pump ever produced a pressure near the point that the accumulator tank was in danger, the program would stop what it was doing and open all of the valves between the accumulator and the catch tank. Excess pressure would be allowed to drain through the system, and the pump could be shut down to prevent further increases in pressure. As stated before, the current pump hardware is in no danger of producing over-pressurization conditions in the Water System, so the programmatic relief valve exists in the ICM code as a redundant safety check that can be modified in the future should different pump hardware be used.

COPV rupture. While this safety concern was first identified for the 2005 experiment, it was determined to not be a very significant issue for the 2007 experiment. The key factor in this assessment is that the 2007 experiment was run at a much lower pressure. The COPVs are rated for approximately 2800 psig, and the aluminium liner underneath the composite is rated for 500 psig. The 2005 experiment ran at 500 psig, so tank rupture was a concern if the composite were to fail. Even if that were to happen in the 2007 experiment, the operating pressure of 120 psig is well below the liner's rating, and thus tank rupture was not deemed a significant risk. In order to avoid injury to personnel in the extremely rare case that the COPVs were damaged during testing, area control procedures were implemented in the laboratory space. All personnel were evacuated during SFS testing, and the ICM was configured to control the experiment from a remote computer terminal.

### B.1.2 Test Procedures

In keeping with the process of the 2005 experiment, a semi formal procedure document (see Section B.2) was authored to detail how to run the SFS test bed and associated hardware. Procedures for handling safety concerns was also included in this document. The next sections summarize these procedures and explain how the testing caused certain changes to be made.

#### B.1.2.1 Pre and Post-Test Procedures

At the start of any test, the first thing that was done was a check on the electrical system controlling the valves. This could be done safely in the lab area since a normal

shutdown of the full SFS test bed would result in zero pressure in both the Air and Water Systems. When the ICM and the valve power supplies were activated, the manual valve controls were used to check for correct valve operation. The Vent and Pressurization valves made an audible click when actuated, while the Fill and Drain valves did not actuate unless there was air pressure going to the Ross valve controller. When there was only power to the valve controller, the LEDs on the connectors would light up when the valves were set to be open. These LEDs were checked before the Air System was activated, and after that the valves could be rechecked by actuating them. Lastly, the Emergency Stop feature of the ICM was tested by setting the valves to some non-normal position and clicking the Stop button. When this was done, the click of the valves could be heard to verify that the Stop feature was working correctly.

Starting the Air System consisted of opening the valves between the Air Supply Tank and the dome regulator, and then using the hand regulator to set the air inlet pressure. Normally the Air Supply Tank was kept at or near 2000 psig, because that was the pressure assumed in the flow calculations for the dome regulator. Between the Supply Tank and the dome regulator were four valves: the Supply Tank Main Valve, the PID Valve, the Shutoff Valve, and the Flow Control Valve. Before the Air System was activated, the Supply Tank and Flow Control valves were verified to be closed, and the other two opened. The PID valve was a part of the infrastructure at the PRC, and was left open at all times. The Shutoff valve was also part of the infrastructure, but it was included in the test procedures as a quick way to halt air flow through the system in the case of an emergency. The control switch for the Shutoff valve was wired independently of the ICM and the other SFS hardware, and was controlled from the PRC control room.

This was where the remote computer terminal, used to control the SFS test bed during actual experiments, was located as well.

In order to avoid slamming the components downstream of the Supply Tank with 2000 psig all at once, the Supply Tank valve was opened first slowly to let the air through to the Flow Control Valve. Since the Flow Control valve was a needle valve, it was very easy to open in slowly and in a controlled fashion. Once all the valves to the dome regulator were opened, the ICM was consulted to make sure that the air inlet pressure was at its expected value. If not, the pressure could be adjusted through the hand regulator mounted on the SFS frame.

With the Air System activated, the Fill and Drain valves would actuate properly and the Water System could be started. This consisted of turning on the main pump (set to run at 50Hz), filling the water supply tank, and flushing out the test bed. Activating the water supply to the test bed was then just a simple matter of turning on the pump to this setting and opening the gate valve at the pump's outlet. If necessary, the return pump was activated for a short time to fill the supply tank with water from the catch tank.

During assembly and testing of the Water System, it was noticed that there was usually a small amount of rust and discoloration in the water when the system had been left alone for a few days. Cleanliness of the water was handled by a filter on the return pump line, but some impurities cropped up from the 4" pipe on the pump outlet. In order to get rid of anything floating in the water that was left in the test bed, the Fill and Drain valves were all opened and the main pump was allowed to run. This pushed new water into the test bed and pushed the old water out into the catch tank (where it would get filtered again). After doing this for a few seconds, the water in the test bed could be

checked by closing the Fill valves and opening the manual drain valves to make sure the water ran clear. Any remaining water in the tanks was flushed into the catch tank by closing the Vent valves, opening the Drain valve on the tank to be flushed, and then briefly opening that tank's Pressurization valve. When the pressure in the flushed tank returned to 0 psig, all of the water had been removed, and the process could be repeated for the other tanks.

Post-test procedures consist of the same actions done pre-test, only in reverse order. First, the main pump was turned off and the gate valve between the pump outlet and the test bed was closed. Next, the Air System was vented by closing the Air Supply Tank Main Valve and opening all of the Pressurization and Vent valves. The Flow Control valve was left open for this stage so that everything downstream of the air Supply Tank would vent. When that was done, the Flow Control valve was closed so it would be ready for the next time the Air System was started. Lastly, the power supplies to the valve control system were turned off and the ICM was stopped.

#### B.1.2.2 Calibration Tests

The purpose of the calibration tests was to determine the timing inputs that would go into making an Auto Sequence for the ICM to run. Due to small discrepancies in hardware, such as slightly different tubing lengths and different valve characteristics, each tank behaved differently during the different SFS Cycle Stages. By doing the calibration tests and looking at the results, a timing input for the Auto Sequence could be found that would work well enough for all three tanks.

The calibration tests were all run through the ICM, but since they did not follow the regular SFS Auto Sequence, a File Sequence had to be generated and used for each calibration test. The TestGenerator VI was programmed to quickly generate Sequence files for the various calibration tests. Once the Sequences were generated, the ICM could be set to run the File Sequence and note what kind of calibration test was being run on the data file.

TestGenerator was programmed such that each later calibration test built upon the last. An example of this is the way the Fill-1 and Fill-2 Stage calibration tests were done. A Fill-1 calibration assumes that the Vent valve is closed throughout the Fill Stage, so only the Fill Time input is used. The results of this calibration give a general idea of how long the Fill Stage should last. During actual testing, it was found that with the Vent closed the pressure in the COPV would build up to the point where water flow into the tank was stopped completely. That led immediately to the Fill-2 calibration, which attempts to determine how long the Vent should be open during the Fill Stage. The timing results of the Fill-1 calibration were used in the Fill-2 calibration's Fill Time input, while the goal of the tests was to find the right Fill/Vent Time input to achieve correct fill rates.

Similarly, the Pressurization Stage calibration test built on the Fill-1 and Fill-2 calibration tests, resulting in a value for Press Time. Next, the Vent Stage calibration was done to find the Vent Time input. This calibration was done differently from the others in that it did not use a File Sequence. Instead, the manual valve controls were used to pressurize each tank, and then vent them one at a time to atmospheric pressure. Data was recorded during this test, and by examining the data file, the time necessary for the Vent



Stage could be determined. With this piece of data, the Drain Stage timing calibration could be done. This test mimicked a full Tank Cycle. The tanks were filled, pressurized, and then a Drain Stage was attempted using the specified timing inputs. Determining the correct Drain Time was important because otherwise there might be too much water left in the tanks after the Drain (causing problems when a refill was attempted) or all of the water might have been expelled along with some pressurant gas. To simulate a real engine, the latter case is something that was to be avoided.

During calibration of the Drain Stage, it was found that the system could drain very rapidly, but too fast to allow for a refill of the tanks. To correct this, the venturi tube used in the drain line was replaced with a smaller version. This change lengthened the Drain time so that the Stage was long enough to allow for tank refill. In an actual system, the lines of the rest of the system would be sized more appropriately to allow for refill to occur in the right amount of time.

The final calibration test was the Refill Verification test. It was important to know that the results from the other calibration tests were repeatable over multiple cycles, specifically that the amount of water left in the COPVs after two (or more) fills was acceptable and that another Drain Stage would not leave too much or too little water in the COPV. Using the timing inputs from the previous calibration tests, TestGenerator was used to make a one-Cycle Three-Tank test, which behaved like a normal Auto Sequence (without any overlap in the Drain Stage, which was handled in later testing).

While the test program only included operation for two cycles, enough to get the fail-operational test data, these calibration tests were included as a way to verify that the timing inputs used for later tests could reasonably be used for long duration test of the

system. The calibration tests themselves start at very specific conditions, while those conditions may not be easily replicable when the system is actually running. Verification that the actual results were within expected limits served to show that the system was functioning correctly.

#### B.1.2.3 Baseline Tests

Before the fail-operational tests could be conducted, the behavior of the test bed under normal operating conditions needed to be characterized. The purpose of the baseline tests was to run the test bed with a series of full-Cycle Auto Sequences and record the data to observe its performance.

The primary measure for the baseline tests was the intensity and duration of pressure fluctuations at the water outlet. In the 2005 experiment, various methods were used to try and minimize the fluctuations at the water outlet. For the 2007 experiment, it was decided to use the most successful method from the 2005 experiment: varying the length of time that Drain Stages were overlapped.

The very first baseline test was just an ICM generated Auto Sequence with the Overlap time set to zero. Later baseline tests varied the Overlap time until a setting was found that minimized the outflow pressure variations. This test was then used as the standard data set that the fail-operational data could be compared to.

It should be noted that to use the Auto Sequence as it was implemented in the ICM, the SFS run tanks had to be filled and pressurized prior to the start of the Sequence. In order to do this, a special “System Refill” File Sequence was made and run before each baseline and fail-operational test. The System Refill used the results from the

calibration tests to run Fill and Pressurize Stages on all three tanks, putting them into the expected state for the start of the Auto Sequence. Similarly, each Auto Sequence would end with the run tanks filled and pressurized. This was not a desirable starting point for later tests because the amount of water in each tank would not be known precisely (due to variations in the individual tanks). Thus a “System Purge” file was created to open the drains and use air pressure to force any water out of the tanks. This File Sequence was run at the end of each baseline and fail-operational test.

#### B.1.2.4 Fail-Operational Tests

The fail-operational tests were the final test runs done for the SFS test bed in the 2007 experiment. While the three-tank SFS concept has an intrinsic fail-operational mode, the purpose of this testing was to quantify the specific effects various failure modes would have on a real system.

The failure modes investigated in the 2005 experiment were the Start and Stop failures: mechanical problems with the valves that would show up during operation as a failure to actuate properly at the beginning or end of an SFS Cycle Stage. These failure modes were investigated by setting the test bed to run a standard two-cycle Auto Sequence through the ICM using the timing results from the calibration tests, and the overlap results from the baseline tests. In each test, one failure was injected into the Sequence using the ICM’s interface. Data was recorded throughout the test to monitor the effect of outflow pressure (and elsewhere in the system) during the failure, as well as during the automatic response to two-tank operation.

The actual operation of the fail-operational test was very similar to that of the baseline tests. First, timing data from the calibration was input into the ICM to generate the Auto Sequence. Next, the “System Refill” File Sequence was run to put the tanks in the expected state for the Auto Sequence. Failure settings were input to the program (this step had to be done after the refill Sequence to avoid program errors) and the Auto Sequence was initiated. At the end, the “System Purge” File Sequence was run to empty the tanks and ready them for the next test.

## B.2 Test Procedure Document

### **Purpose**

This document describes the procedural steps required for operating the Sequential Feed System (SFS) located at the Johnson Research Center (JRC).

### **General**

The Test Conductor is in charge of test operations conducted for this program and is responsible for ensuring that the appropriate procedures and policies are followed. All activities pertaining to tests that are performed will be conducted through this Test Conductor.

### **Safety**

#### **1) Hazards**

##### **High Pressure Air**

- Over-pressure (greater than 120 psig) of system is prevented by using properly sized relief valves.
- Hearing protecting will be utilized as required.
- Area control is implemented in this test procedure.

##### **Damage due to Excessive Water Pressure**

- Over-pressure (greater than 100 psig) of system is prevented by using software control to vent excess water pressure into a catch tank.
- Additionally, water pump hardware is designed so that it cannot exceed safe operating conditions when operating normally.

##### **COPV Rupture**

- Composite Over-wrapped Pressure Vessels (COPVs) used in this experiment are rated for approximately 500 psig, well above the operating pressure.
- Area control is implemented in this test procedure to prevent injury to personnel if COPVs fail.

#### **2) Emergency Control Center**

The JRC will be the emergency control center if required.

#### **3) Injuries**

In the event of an injury, do not move the injured personnel unless failure to do so may result in further injury.

#### **4) Emergency Telephone Numbers**

UAH Police	824-6911
Ambulance	911

## **Pretest Procedures**

### **1) Verify and Implement Area Control**

- a) ( ) Ensure that all personnel have vacated the test area prior to the start of testing.
- b) ( ) Install warning signs on all doors leading to the laboratory area of the JRC.
- c) ( ) Ensure that the Plasmoid Thruster Experiment (PTX) area of the JRC is vacated or that the doors to this area of building have warning signs posted.

### **2) Verify Initial Facility Configuration**

- a) ( ) Visually inspect facility fluid systems, verify all systems that affect the execution of this procedure are fully assembled and that no visually obvious problems are present. If a problem is discovered, discontinue this procedure immediately and take corrective action.

- b) ( ) Verify that the facility system is isolated and vented by confirming the position of the following facility hand valves, hand pressure regulators, and pressure gauges:

- 1) ( ) SFS Air Inlet Pressure Gauge.....0 PSIG
- 2) ( ) Air Supply Tank Main Valve.....CLOSED
- 3) ( ) Air Supply Tank Inlet Valve.....CLOSED
- 4) ( ) PID Valve.....OPEN
- 5) ( ) Shutoff Valve.....CLOSED
- 6) ( ) Flow Control Valve.....CLOSED
- 7) ( ) PID Valve and Shutoff Valve Air Supply.....CLOSED
- 8) ( ) Shutoff Valve Air Supply Hand Regulator.....0 PSIG
- 9) ( ) Air Supply Pressure Gauge.....1500-2000 PSIG

Note: The Air Supply Tank should be close to 2000 psig before testing begins. The pressure will drop as air is removed from the tank.

Acceptable operating limits are within 1000 to 2000 psig. The Air Supply Tank pressure MUST NOT be higher than 2000psig, as the relief valve may not be able to accommodate the flow rates at higher supply pressure.

- 10) ( ) Water System Catch Tank Valve.....OPEN
- 11) ( ) Water System Gate Valve.....CLOSED
- 12) ( ) Hand Valve, Tank 1 Drain.....CLOSED
- 13) ( ) Hand Valve, Tank 2 Drain.....CLOSED
- 14) ( ) Hand Valve, Tank 3 Drain.....CLOSED

### 3) Set up/verify Control and Data Acquisition Systems

- a) ( ) Start Integrated Control and Monitoring (ICM) LabVIEW program on the lab computer. Verify that transducer power supply is on (set to 28 volt DC) and that measurements are being properly displayed on the Data tab of the ICM.
- b) ( ) On the ICM front panel, verify all buttons for manual control of valves are properly initialized to the de-energized position (OPEN for vent valves, and CLOSED for other valves).
- c) ( ) Turn on 5 volt DC power supply for the relay panel.
- d) ( ) Turn on 28 volt DC power supply system for the solenoid valves.

### 4) Verify Operation of Solenoid Valves

- a) ( ) Verify that the following SOVs cycle properly when commanded, by listening for the solenoid to click open and closed.
  - 1) ( ) Tank 1 Vent.....CLOSED/OPEN/CLOSED
  - 2) ( ) Tank 2 Vent.....CLOSED/OPEN/CLOSED
  - 3) ( ) Tank 3 Vent.....CLOSED/OPEN/CLOSED
  - 4) ( ) Tank 1 Pressurization.....OPEN/CLOSED/OPEN
  - 5) ( ) Tank 2 Pressurization.....OPEN/CLOSED/OPEN
  - 6) ( ) Tank 3 Pressurization.....OPEN/CLOSED/OPEN
- b) ( ) Set all Fill and Drain valves to OPEN and confirm that the LED on each module of the Ross Valve Controller is lit. Reset valves to CLOSED when done.

- c) ( ) Proceed to step 5 of the Pretest Procedures, and verify the Emergency Safe-Out Functionality. This should be done prior to the activation of the Air System, as it requires multiple ICM restarts and the valve relays tend to briefly skip to the energized setting when the program activates. After the completion of that procedure, continue with step 4-d of the Pretest Procedures.
- d) ( ) Proceed to step 6 of the Pretest Procedures, and activate the Air System. After the completion of that procedure, continue with step 4-e of the Pretest Procedures.
- e) ( ) Verify that the following ROVs cycle properly when commanded, by visually confirming that the ball valve stems cycle. This can be done visually for each valve, or by listening for the pop sound when the valve actuates.
  - 1) ( ) Tank 1 Drain.....OPEN/CLOSED/OPEN
  - 2) ( ) Tank 2 Drain.....OPEN/CLOSED/OPEN
  - 3) ( ) Tank 3 Drain .....OPEN/CLOSED/OPEN
  - 4) ( ) Tank 1 Fill.....OPEN/CLOSED/OPEN
  - 5) ( ) Tank 2 Fill.....OPEN/CLOSED/OPEN
  - 6) ( ) Tank 3 Fill.....OPEN/CLOSED/OPEN
- f) ( ) Remove plastic caps from the vent valves and the manual drain valves.

## 5) Verify Emergency Safe-Out Functionality

- a) ( ) Click on the Emergency Shutdown Icon on the ICM front panel.
- b) ( ) Verify visually, audibly, or by feeling (with your hand) that the following valves cycle to the following positions. This step may require repeating several times to check all the following valves. This can be done by pressing the start button (labeled with an arrow) on the ICM front panel and pressing the labeled buttons to place the following valves in the ON position (energized state) before pressing the Emergency Shutdown Icon again.
  - 1) ( ) Tank 1 Drain.....CLOSED
  - 2) ( ) Tank 2 Drain.....CLOSED
  - 3) ( ) Tank 3 Drain.....CLOSED



- 4) ( ) Tank 1 Fill.....CLOSED
- 5) ( ) Tank 2 Fill.....CLOSED
- 6) ( ) Tank 3 Fill.....CLOSED
- 7) ( ) Tank 1 Pressurization.....CLOSED
- 8) ( ) Tank 2 Pressurization .....CLOSED
- 9) ( ) Tank 3 Pressurization .....CLOSED
- 10) ( ) Tank 1 Vent.....OPEN
- 11) ( ) Tank 2 Vent.....OPEN
- 12) ( ) Tank 3 Vent.....OPEN

#### **6) Activate Air System**

- a) ( ) Open the Shutoff Valve's Air Supply Valve and confirm that the hand regulator is set to 100 psig.
- b) ( ) Verify that the PID Valve is open.
  - 1) ( ) If the PID Valve is not open, start the Digital PID Control VI on the control room computer.
  - 2) ( ) Set PID Voltage to 5 and run the VI. PID Valve should open.
  - 3) ( ) If the PID Valve does not open, discontinue this procedure and take corrective action.
- c) ( ) In the control room, toggle the switch to open the Shutoff Valve.
- d) ( ) On the control room computer, open remote access to the laboratory computer and make sure that the ICM front panel is visible.
- e) ( ) Unlock Air Supply Tank Main Valve and slowly open it.
- f) ( ) Manually open the Flow Control Valve.

- g) ( ) Confirm through ICM that the pressure at the air inlet is at the desired setting, adjust hand regulator on the SFS frame (connected to the dome regulator) as necessary.
- h) ( ) If necessary, return to step 4-d of the Pretest Procedures and test the actuation of the Fill and Drain ROVs.

## **7) Activate Water System**

- a) ( ) Unlock return pump activation switch and main pump controller power supply.
- b) ( ) Turn on main pump controller.
- c) ( ) Verify that the catch tank valve on the drain line into the catch tank is open, and that all manual drain valves are closed.
- d) ( ) Activate the return pump and allow the water to fill the pump supply tank. Do not overfill the supply tank or run the return pump when the catch tank is empty.
- e) ( ) Shut off the return pump and open the gate valve on the 4" line.
- f) ( ) Using the ICM's manual controls, open all drain and fill valves.
- g) ( ) Press the SHIFT key on the main pump controller's keypad and set the pump to 50 Hz.
- h) ( ) Press the RUN key on the main pump controller to activate the pump.
- i) ( ) Confirm through the ICM that the water inlet pressure is less than 30 psig, and that it does not get higher.
- j) ( ) Allow the water to flow through all of the fill and drain valves for a few seconds, flushing the system into the catch tank.
- k) ( ) Close the fill valves and keep the drain valves open to allow the water to drain out of the tanks.
- l) ( ) After a few seconds close the drain valves and open the manual drain valves on each tank. Capture the water and make sure it runs clear.

- m) ( ) If there is still water left in the tanks, close the vent valves and, for each tank one at a time, open the drain valve and briefly open the pressurization valve. When the pressure in the tank goes back to zero psig (after venting through the drain line), close the drain valve and reopen the vent.
- n) ( ) With the tanks empty of water and all fill valves closed, confirm through the ICM that the water inlet pressure is approximately 30 psig, and that it does not get higher.

## **Safety Procedures**

### **1) Air System Over-pressurization**

- a) ( ) If the air inlet pressure readout on the ICM front panel exceeds 150 psig during a test the system is over-pressurized.
- b) ( ) Verify that the air inlet relief valve is functioning (will be audible when air is escaping through the relief valve).
- c) ( ) Click the Emergency Stop Icon on the ICM front panel and toggle the switch in the control room to close the Shutoff Valve.
- d) ( ) Confirm that the system responds to the Emergency Stop correctly (all vents open, all other valves closed) and that the pressure drops below 150 psig.
- e) ( ) When the area is deemed safe to enter, close the Air Supply Tank Main Valve.
- f) ( ) Restart the ICM and use the manual valve controls to open all vent and pressurization valves (this will vent any remaining gas in the pressurant line). Toggle the switch to open the Shutoff valve to vent any air between it and the Air Supply Tank Main Valve.
- g) ( ) Follow Air System Shutdown procedure and make sure the system is safe.
- h) ( ) If the over-pressurization was due to a malfunction, take corrective action.
- i) ( ) Even if no malfunction is present, do not simply restart the ICM and continue. The Air system should be vented and step 6 of the Pretest Procedures should be redone.

### **2) Water System Over-pressurization**

- a) ( ) If the water inlet pressure readout on the ICM front panel exceeds 75 psig during a test the system is over-pressurized.
- b) ( ) The ICM is coded to open all vent, fill, and drain valves automatically during a Water System over-pressurization.
- c) ( ) Confirm that the running Sequence has halted, and that the valves are in their expected positions, allowing excess water pressure to flow into the catch tank.
- d) ( ) When the water inlet pressure drops, enter the lab area and shut down the main pump.

- e) ( ) Follow Water System Shutdown procedure and make sure the system is safe.
- f) ( ) If the over-pressurization was due to a malfunction, take corrective action.

### 3) Power Supply Failure

- a) ( ) If either of the 28 volt DC power supplies to the valves or the DAQ sensors fails, or if the 5 volt DC power supply to the relay panel fails, the SOVs should all revert to their de-energized position: OPEN for the vent valves and CLOSED for the other valves.
- b) ( ) In the event of a power failure, click the Emergency Stop Icon immediately, and toggle the control room switch to close the Shutoff Valve.
- c) ( ) Continue with step 1-d of the Safety Procedures.
- d) ( ) Follow Water System Shutdown procedure and make sure the system is safe.
- e) ( ) Take corrective action to restore the power supplies.

### 4) COPV Rupture

- a) ( ) Click the Emergency Stop Icon immediately, and toggle the control room switch to close the Shutoff Valve.
- b) ( ) Continue with step 1-d of the Safety Procedures.
- c) ( ) Follow Water System Shutdown procedure and make sure the system is safe.

**Post-test Procedures****1) Water System Shutdown**

- a) ( ) Verify that the test Sequence has completed and that all Fill and Drain valves are closed. This should be done before the air system is shut down.
- b) ( ) Enter the lab area and press the STOP key on the main pump controller.
- c) ( ) Turn off and lock the return pump and the power supply to the main pump.
- d) ( ) Remove any residual water from the COPVs using the manual drain valves and return that water to the catch tank.
- e) ( ) Replace the plastic caps on the outlets of the manual drain valves.

**2) Air System Shut Down**

- a) ( ) Verify through the ICM in the control room that the test Sequence has completed, and that no malfunctions in the system have occurred.
- b) ( ) Toggle the control room switch to close the Shutoff Valve.
- c) ( ) Enter the laboratory area and close the Air Supply Tank Main Valve.
- d) ( ) Return to the control room and toggle the switch to reopen the Shutoff Valve.
- e) ( ) Using the ICM manual valve controls, open all vent valves, and then open all pressurization valves.
- f) ( ) Allow the system to vent to zero psig. Note that the air inlet may not reach zero psig because of the check valves in place on the pressurant line. A final air inlet pressure no higher than 10 psig is acceptable.
- g) ( ) Toggle the control room switch to close the Shutoff Valve and manually close the Flow Control Valve.
- h) ( ) Stop the ICM program.
- i) ( ) Turn off the 5 volt DC power supply to the relay panel and the 28 volt DC power supply to the valves.
- j) ( ) Remove area control warning signs from laboratory doors.
- k) ( ) Replace the plastic caps on the outlets of all vent valves.

### **Calibration Test Procedures**

#### **1) Fill-1 Stage (Vent Closed) Calibration**

The purpose of this calibration test is to determine how long the fill valves should be open in order to fill the COPVs with 10 gallons of water.

- a) ( ) Load the TestGenerator LabVIEW program and run it.
- b) ( ) Select the “Fill/Vent Calibration” option from the “Test Type” Menu Ring.
- c) ( ) Input the desired fill time in the “Fill Time” input on the VI front panel.
- d) ( ) Click the “Generate Sequence” button and verify that the output data is in the expected form.
- e) ( ) For each calibration test, print out a copy of the “Calibration Test Summary” page (see end of this document) and record the following information.
  - 1) ( ) Date of the Test.
  - 2) ( ) The type of calibration test, Fill-1.
  - 3) ( ) The timing inputs used in TestGenerator.
- f) ( ) When satisfied with the Sequence data generated by TestGenerator, click the “Save File” button and select a text file to save the data to. Record the name of the text file under the “Sequence File Name” part of the summary page.
- g) ( ) If not done already, follow the Pretest Procedures to start the Air and Water systems.
- h) ( ) On the ICM’s “Settings” Tab, select the “File” option from the “Sequence Type” Menu Ring.
- i) ( ) In the “Filename” input, select the text file that contains the Sequence information from TestGenerator.
- j) ( ) Fill in the “Run”, “Test Type”, and “Comments” fields as appropriate.
- k) ( ) On the “Data” Tab, click the “Acquire Data” button.
- l) ( ) On the “Settings” Tab, click the “Begin Sequence” button.
- m) ( ) After the Sequence is completed, press the “Acquire Data” button again to halt data acquisition and note the file name in the “Last Saved File” output on the “Settings” Tab under “Data Record File” on the summary page.

- n) ( ) Verify through the ICM that the pressurization, drain, and fill valves are closed.
- o) ( ) Use the ICM manual controls to open all vent valves.
- p) ( ) Enter the laboratory area and capture the water in the SFS tanks by using the manual drain valves and a graduated container.
- q) ( ) Record the volume of water in each tank on the summary page.
- r) ( ) Return water from the SFS tanks to the catch tank after measurement.
- s) ( ) Repeat this procedure, varying the “Fill Time” input on TestGenerator for each test, enough times so that the volume of water in each SFS tank as a function of fill time can be sufficiently characterized.
- t) ( ) Determine the amount of time that the Fill valves should be open to allow the SFS tanks to fill to approximately 10 gallons. Use this as the “Fill Time” input in future tests.

## 2) Fill-2 Stage (Vent Open) Calibration

The purpose of this calibration test is to determine how long the vent valve should be open during the Fill Stage of the SFS Cycle. With the vent closed during the entire Stage, the air inside the COPV will be compressed and in principle can hinder the Fill process. Opening the vent part-way through the Fill process can facilitate the flow of water into the COPV.

- a) ( ) Load the TestGenerator LabVIEW program and run it.
- b) ( ) Select the “Fill Calibration” option from the “Test Type” Menu Ring.
- c) ( ) Input the result from the Fill-1 Stage calibration in the “Fill Time” input and the desired Fill-2 time (how long during the Fill Stage the vent should be open) in the “Fill/Vent Time” input on the VI front panel.  
NOTE: The ICM and associated VIs interpret “Fill/Vent” time as how long *at the end* of the Fill Stage that the Vent should be open. For example, if the “Fill Time” were set to 3 seconds and the “Fill/Vent Time” were set to 1 second, the Fill Valve would be open for 2 seconds before the Vent valve opened.
- d) ( ) Click the “Generate Sequence” button and verify that the output data is in the expected form.



- e) ( ) For each calibration test, print out a copy of the “Calibration Test Summary” page (see end of this document) and record the following information.
- 1) ( ) Date of the Test.
  - 2) ( ) The type of calibration test, Fill-2.
  - 3) ( ) The timing inputs used in TestGenerator.
- f) ( ) When satisfied with the Sequence data generated by TestGenerator, click the “Save File” button and select a text file to save the data to. Record the name of the text file under the “Sequence File Name” part of the summary page.
- g) ( ) If not done already, follow the Pretest Procedures to start the Air and Water systems.
- h) ( ) On the ICM’s “Settings” Tab, select the “File” option from the “Sequence Type” Menu Ring.
- i) ( ) In the “Filename” input, select the text file that contains the Sequence information from TestGenerator.
- j) ( ) Fill in the “Run”, “Test Type”, and “Comments” fields as appropriate.
- k) ( ) On the “Data” Tab, click the “Acquire Data” button.
- l) ( ) On the “Settings” Tab, click the “Begin Sequence” button.
- m) ( ) After the Sequence is completed, press the “Acquire Data” button again to halt data acquisition and note the file name in the “Last Saved File” output on the “Settings” Tab under “Data Record File” on the summary page.
- n) ( ) Verify through the ICM that the pressurization, drain, and fill valves are closed.
- o) ( ) Use the ICM manual controls to open all vent valves.
- p) ( ) Enter the laboratory area and capture the water in the SFS tanks by using the manual drain valves.
- q) ( ) Record the volume of water in each tank on the summary page.
- r) ( ) Return water from the SFS tanks to the catch tank after measurement.

- s) ( ) Repeat this procedure, varying the “Fill/Vent Time” and “Fill Time” inputs on TestGenerator, enough times so that the time needed for the vent to be open during the Fill Stage can be characterized.
- t) ( ) Determine the amount of time that the Fill valves should be open to allow the SFS tanks to fill to approximately 10 gallons, and how long the vent valve should be open during that time to facilitate the refill process. Use these results as the “Fill Time” and “Fill/Vent Time” inputs in future tests.

### 3) Pressurization Stage Calibration

The purpose of this calibration test is to determine how long the pressurization valve should be opened in order to pressurize a full COPV.

- a) ( ) Load the TestGenerator LabVIEW program and run it.
- b) ( ) Select the “Pressurize Calibration” option from the “Test Type” Menu Ring.
- c) ( ) Input the results from the Fill-1 and Fill-2 Stage calibrations in the “Fill Time” and “Fill/Vent Time” inputs, respectively
- d) ( ) Set the “Press Time” input to 10 seconds.
- e) ( ) Click the “Generate Sequence” button and verify that the output data is in the expected form.
- f) ( ) Print out a copy of the “Calibration Test Summary” page (see end of this document) and record the following information.
  - 1) ( ) Date of the Test.
  - 2) ( ) The type of calibration test, Pressurize.
  - 3) ( ) The timing inputs used in TestGenerator.
- g) ( ) When satisfied with the Sequence data generated by TestGenerator, click the “Save File” button and select a text file to save the data to. Record the name of the text file under the “Sequence File Name” part of the summary page.
- h) ( ) If not done already, follow the Pretest Procedures to start the Air and Water systems.
- i) ( ) On the ICM’s “Settings” Tab, select the “File” option from the “Sequence Type” Menu Ring.

- j) ( ) In the “Filename” input, select the text file that contains the Sequence information from TestGenerator.
- k) ( ) Fill in the “Run”, “Test Type”, and “Comments” fields as appropriate.
- l) ( ) On the “Data” Tab, click the “Acquire Data” button.
- m) ( ) On the “Settings” Tab, click the “Begin Sequence” button.
- n) ( ) After the Sequence is completed, press the “Acquire Data” button again to halt data acquisition and note the file name in the “Last Saved File” output on the “Settings” Tab under “Data Record File” on the summary page.
- o) ( ) Verify through the ICM that the pressurization, drain, and fill valves are closed.
- p) ( ) Use the ICM manual controls to open all vent valves.
- q) ( ) Examine the data file and determine the time necessary for the tanks to be pressurized to the desired air pressure (the normal air inlet pressure).
- r) ( ) If the tanks did not reach the desired pressure in 10 seconds, repeat the calibration with a larger value setting for the “Press Time” input.
- s) ( ) Use the result of this calibration as the “Press Time” input in future tests.

#### **4) Vent Stage Calibration**

The purpose of this calibration test is to determine how long the vent valve should be opened in order to fully vent an empty COPV.

- a) ( ) Print out a copy of the “Calibration Test Summary” page (see end of this document) and record the following information.
  - 1) ( ) Date of the Test.
  - 2) ( ) The type of calibration test, Pressurize.
- b) ( ) If not done already, follow the Pretest Procedures to start the Air and Water systems.
- c) ( ) Enter the laboratory area and ensure that the SFS tanks are completely empty using the manual drain valves.

- d) ( ) Return any water that was in the SFS tanks to the catch tank.
- e) ( ) On the ICM's "Settings" Tab, Fill in the "Run", "Test Type", and "Comments" fields as appropriate.
- f) ( ) On the "Data" Tab, click the "Acquire Data" button.
- g) ( ) Use the manual valve controls to close all vent valves.
- h) ( ) Use the manual valve controls to open all pressurization valves, and leave them open until the tank pressure equilibrates with the air inlet pressure.
- i) ( ) Verify through the ICM that the pressurization, drain, and fill valves are closed.
- j) ( ) Use the manual valve controls to open all vent valves.
- k) ( ) Once all of the tanks have vented to atmospheric pressure, click the "Acquire Data" button again to halt data acquisition and note the file name in the "Last Saved File" output on the "Settings" Tab under "Data Record File" on the summary page.
- l) ( ) Examine the data file and determine the time necessary for the tanks to vent to atmospheric pressure (0 psig).
- m) ( ) Use the result of this calibration as the "Vent Time" input in future tests.

### 5) Drain Stage Calibration

The purpose of this calibration test is to determine how long the Drain and Pressurization valves should be open in order to expel 10 gallons of water from the COPV into the Drain line.

- a) ( ) Load the TestGenerator LabVIEW program and run it.
- b) ( ) Select the "Drain Calibration" option from the "Test Type" Menu Ring.
- c) ( ) Input the results from the Fill-1, Fill-2, and Pressurization Stage calibrations in the "Fill Time", "Fill/Vent Time", and "Press Time" inputs, respectively. Input the desired drain time in the "Drain Time" input on the VI front panel.
- d) ( ) Click the "Generate Sequence" button and verify that the output data is in the expected form.

- e) ( ) For each calibration test, print out a copy of the “Calibration Test Summary” page (see end of this document) and record the following information.
- 1) ( ) Date of the Test.
  - 2) ( ) The type of calibration test, Drain.
  - 3) ( ) The timing inputs used in TestGenerator.
- f) ( ) When satisfied with the Sequence data generated by TestGenerator, click the “Save File” button and select a text file to save the data to. Record the name of the text file under the “Sequence File Name” part of the summary page.
- g) ( ) If not done already, follow the Pretest Procedures to start the Air and Water systems.
- h) ( ) On the ICM’s “Settings” Tab, select the “File” option from the “Sequence Type” Menu Ring.
- i) ( ) In the “Filename” input, select the text file that contains the Sequence information from TestGenerator.
- j) ( ) Fill in the “Run”, “Test Type”, and “Comments” fields as appropriate.
- k) ( ) On the “Data” Tab, click the “Acquire Data” button.
- l) ( ) On the “Settings” Tab, click the “Begin Sequence” button.
- m) ( ) After the Sequence is completed, press the “Acquire Data” button again to halt data acquisition and note the file name in the “Last Saved File” output on the “Settings” Tab under “Data Record File” on the summary page.
- n) ( ) Verify through the ICM that the pressurization, drain, and fill valves are closed.
- o) ( ) Use the ICM manual controls to open all vent valves.
- p) ( ) Enter the laboratory area and capture the water in the SFS tanks by using the manual drain valves.
- q) ( ) Record the volume of water in each tank on the summary page.
- r) ( ) Return water from the SFS tanks to the catch tank after measurement.

- s) ( ) Repeat this procedure, varying the “Drain Time” input on TestGenerator, enough times so that the volume of water remaining in the SFS tanks as a function of drain time can be characterized.
- t) ( ) Determine the amount of time that the Drain valves should be open to allow the SFS tanks to drain. The drain time needs to meet two criteria.
  - 1) ( ) At no point will the tanks drain so long that they are completely empty of water, allowing air to vent through the drain line into the catch tank.
  - 2) ( ) The amount of water remaining in the tank is not so much that when refilled the volume of the tank is much more than 10 gallons.
- u) ( ) Use the result of this calibration as the “Drain Time” input in future tests.

#### **6) Refill Verification – End of Drain**

The purpose of this calibration test is to verify that the results from the previous calibrations are repeatable for multiple tank cycles, specifically that the amount of water left in the COPVs after two drains is the expected amount and that another Fill Stage will not overfill the COPV.

- a) ( ) Enter the laboratory area and ensure that the SFS tanks are completely empty using the manual drain valves.
- b) ( ) Return any water that was in the SFS tanks to the catch tank.
- c) ( ) Load the TestGenerator LabVIEW program and run it.
- d) ( ) Select the “Three Tank” option from the “Test Type” Menu Ring.
- e) ( ) Input the results from the previous calibration tests in the appropriate VI input fields. Set “Number of Cycles” to 1.
- f) ( ) Click the “Generate Sequence” button and verify that the output data is in the expected form.
- g) ( ) For each test, print out a copy of the “Calibration Test Summary” page (see end of this document) and record the following information.
  - 1) ( ) Date of the Test.
  - 2) ( ) The type of calibration test, Refill Verification – End of Drain.

- 3) ( ) The timing inputs used in TestGenerator.
- h) ( ) When satisfied with the Sequence data generated by TestGenerator, click the “Save File” button and select a text file to save the data to. Record the name of the text file under the “Sequence File Name” part of the summary page.
- i) ( ) If not done already, follow the Pretest Procedures to start the Air and Water systems.
- j) ( ) On the ICM’s “Settings” Tab, select the “File” option from the “Sequence Type” Menu Ring.
- k) ( ) In the “Filename” input, select the text file that contains the Sequence information from TestGenerator.
- l) ( ) Fill in the “Run”, “Test Type”, and “Comments” fields as appropriate.
- m) ( ) On the “Data” Tab, click the “Acquire Data” button.
- n) ( ) On the “Settings” Tab, click the “Begin Sequence” button.
- o) ( ) After the Sequence is completed, continue to record data and confirm that the file settings are preserved in the ICM. Press the “Begin Sequence” button again and run the same Sequence file as before.
- p) ( ) After the end of the second Sequence, press the “Acquire Data” button again to halt data acquisition and note the file name in the “Last Saved File” output on the “Settings” Tab under “Data Record File” on the summary page.
- q) ( ) Verify through the ICM that the pressurization, drain, and fill valves are closed.
- r) ( ) Use the ICM manual controls to open all vent valves.
- s) ( ) Enter the laboratory area and capture the water in the SFS tanks by using the manual drain valves.
- t) ( ) Record the volume of water in each tank on the summary page.
- u) ( ) Return water from the SFS tanks to the catch tank after measurement.
- v) ( ) Verify that the amount of water remaining in each SFS tank is what should be expected after a Drain Stage.

- w) ( ) If the water amount is off by a significant amount, rerun the test to confirm the discrepancy. If it persists, rerun the fill and drain calibrations and adjust the timing inputs as necessary.

## **7) Refill Verification – End of Fill**

The purpose of this calibration test is to verify that the results from the previous calibrations are repeatable for multiple tank cycles, specifically that the amount of water left in the COPVs after two fills is the expected amount and that another Drain Stage will not leave too much or too little water in the COPV.

- a) ( ) Enter the laboratory area and ensure that the SFS tanks are completely empty using the manual drain valves.
- b) ( ) Return any water that was in the SFS tanks to the catch tank.
- c) ( ) Load the TestGenerator LabVIEW program and run it.
- d) ( ) Select the “Three Tank” option from the “Test Type” Menu Ring.
- e) ( ) Input the results from the previous calibration tests in the appropriate VI input fields. Set “Number of Cycles” to 1.
- f) ( ) Click the “Generate Sequence” button and verify that the output data is in the expected form.
- g) ( ) For each test, print out a copy of the “Calibration Test Summary” page (see end of this document) and record the following information.
  - 1) ( ) Date of the Test.
  - 2) ( ) The type of calibration test, Refill Verification – End of Drain.
  - 3) ( ) The timing inputs used in TestGenerator.
- h) ( ) When satisfied with the Sequence data generated by TestGenerator, click the “Save File” button and select a text file to save the data to. Record the name of the text file under the “Sequence File Name” part of the summary page.
- i) ( ) Use TestGenerator to generate another Sequence file. The “Fill Calibration” option should be selected for the “Test Type”, and the same timing inputs used for the “Three Tank” Sequence should be used.



- j) ( ) When the second Sequence is complete click the “Save File” button and select a text file to save the data to. Record the name of the text file under the “Sequence File Name” part of the summary page.
- k) ( ) If not done already, follow the Pretest Procedures to start the Air and Water systems.
- l) ( ) On the ICM’s “Settings” Tab, select the “File” option from the “Sequence Type” Menu Ring.
- m) ( ) In the “Filename” input, select the text file that contains the Sequence information from the “Three Tank” option of TestGenerator.
- n) ( ) Fill in the “Run”, “Test Type”, and “Comments” fields as appropriate.
- o) ( ) On the “Data” Tab, click the “Acquire Data” button.
- p) ( ) On the “Settings” Tab, click the “Begin Sequence” button.
- q) ( ) After the Sequence is completed, continue to record data and reset the “Filename” input to select the text file that contains the Sequence information from the “Fill Calibration” option of TestGenerator.
- r) ( ) Press the “Begin Sequence” button again and run the second Sequence file.
- s) ( ) After the end of the second Sequence, press the “Acquire Data” button again to halt data acquisition and note the file name in the “Last Saved File” output on the “Settings” Tab under “Data Record File” on the summary page.
- t) ( ) Verify through the ICM that the pressurization, drain, and fill valves are closed.
- u) ( ) Use the ICM manual controls to open all vent valves.
- v) ( ) Enter the laboratory area and capture the water in the SFS tanks by using the manual drain valves.
- w) ( ) Record the volume of water in each tank on the summary page.
- x) ( ) Return water from the SFS tanks to the catch tank after measurement.
- y) ( ) Verify that the amount of water remaining in each SFS tank is what should be expected after a Fill Stage. If the water amount is off by a significant amount, rerun the test to confirm the discrepancy. If it persists, rerun the fill and drain calibrations and adjust the timing inputs as necessary.

### **Baseline Test Procedures**

#### **1) Create System Refill File**

The ICM assumes that at the start of an Auto Sequence all three tanks will be filled and pressurized. In order to make sure the tanks are in this expected state before the start of a full cycle test, such as a Baseline or Fail-Operational test, a File Sequence must be generated using the calibration timing data. This File Sequence will have the instructions to get the tanks into the necessary state prior to testing.

- a) ( ) Load the TestGenerator LabVIEW program and run it.
- b) ( ) Select the “System Refill” option from the “Test Type” Menu Ring.
- c) ( ) Input the results from the previous calibration tests in the appropriate VI input fields, specifically “Fill Time”, “Fill/Vent Time”, and “Press Time.”
- d) ( ) Click the “Generate Sequence” button and verify that the output data is in the expected form.
- e) ( ) When satisfied with the Sequence data generated by TestGenerator, click the “Save File” button. Name the output file “SFS System Refill.txt.”

#### **2) No Overlap Test**

The purpose of this baseline test is to obtain a data file from a normal SFS Sequence using the timing inputs derived from the calibration tests. This baseline test will not include an overlap of the Drain stages during the Sequence.

- a) ( ) Load the ICM program and run it.
- b) ( ) On the “Settings” tab select the “File” option from the “Sequence Type” Menu Ring.
- c) ( ) In the “Filename” input, select the “SFS System Refill.txt” file from step 1 of the Baseline Test Procedures.
- d) ( ) On the “Settings” Tab, click the “Begin Sequence” button.
- e) ( ) After the System Refill File Sequence has finished, select the “Auto” option from the “Sequence Type” Menu Ring on the “Settings” tab.
- f) ( ) Input the results from the previous calibration tests in the appropriate VI input fields. Set “Overlap” to 0 and “Number of Cycles” to 2.
- g) ( ) Verify that the “Failure Settings” field is empty. If it is not, click the “Clear Settings” switch.

- h) ( ) For each baseline test, print out a copy of the “Baseline Test Summary” page (see end of this document) and record the following information.
  - 1) ( ) Date of the Test.
  - 2) ( ) The timing inputs used.
- i) ( ) If not done already, follow the Pretest Procedures to start the Air and Water systems.
- j) ( ) Fill in the “Run”, “Test Type”, and “Comments” fields as appropriate.
- k) ( ) On the “Data” Tab, click the “Acquire Data” button.
- l) ( ) On the “Settings” Tab, click the “Begin Sequence” button.
- m) ( ) After the Sequence is completed, press the “Acquire Data” button again to halt data acquisition and note the file name in the “Last Saved File” output on the “Settings” Tab under “Data Record File” on the summary page.
- n) ( ) Verify through the ICM that the pressurization, drain, and fill valves are closed.
- o) ( ) Use the ICM manual controls to open all vent valves.
- p) ( ) Examine the data file and quantify the pressure variation that occurs in the water outflow when the system transitions from draining in one SFS tank to draining in another.

### 3) Overlap Test

The purpose of this baseline test is to obtain a data file from a normal SFS Sequence using the timing inputs derived from the calibration tests and to calibrate the time needed to overlap the Drain Stages during the Sequence.

- a) ( ) Follow steps 2-a. through 2-e of the Baseline Test Procedures to run the System Refill File Sequence.
- b) ( ) Input the results from the previous calibration tests in the appropriate VI input fields. Set “Number of Cycles” to 2 and the “Overlap” input to the desired time during which SFS tank drains should overlap.
- c) ( ) Verify that the “Failure Settings” field is empty. If it is not, click the “Clear Settings” switch.

- d) ( ) For each baseline test, print out a copy of the “Baseline Test Summary” page (see end of this document) and record the following information.
- 1) ( ) Date of the Test.
  - 2) ( ) The timing inputs used.
- e) ( ) If not done already, follow the Pretest Procedures to start the Air and Water systems.
- f) ( ) Fill in the “Run”, “Test Type”, and “Comments” fields as appropriate.
- g) ( ) On the “Data” Tab, click the “Acquire Data” button.
- h) ( ) On the “Settings” Tab, click the “Begin Sequence” button.
- i) ( ) After the Sequence is completed, press the “Acquire Data” button again to halt data acquisition and note the file name in the “Last Saved File” output on the “Settings” Tab under “Data Record File” on the summary page.
- j) ( ) Verify through the ICM that the pressurization, drain, and fill valves are closed.
- k) ( ) Use the ICM manual controls to open all vent valves.
- l) ( ) Examine the data file and quantify the pressure variation that occurs in the water outflow when the system transitions from draining in one SFS tank to draining in another.
- u) ( ) Repeat this procedure, varying the “Overlap” input on the ICM, enough times so that the pressure fluctuation during the tank transition can be minimized.
- m) ( ) Use the value of “Overlap” that gives the smallest pressure fluctuation in future tests. The data file from that test will also be the baseline for comparison.

## **Fail-Operational Test Procedures**

### **1) Start/Stop Failure Tests**

The purpose of these tests is to quantify the effect that various Start and Stop failures have on the outflow pressure and flow rate. A Start failure is defined as a failure in which the valve fails to actuate as expected at the start of a Cycle Stage, and a Stop failure is defined as a failure in which the valve fails to actuate as expected at the end of a Cycle Stage. For these tests the timing inputs derived from the calibration tests will be used, the Sequences will run two cycles, and all failures will occur in the second cycle.

- a) ( ) Follow steps 2-a. through 2-e of the Baseline Test Procedures to run the System Refill File Sequence.
- b) ( ) Input the results from the previous calibration tests in the appropriate VI input fields. Set “Number of Cycles” to 2.
- c) ( ) Press the “Failure Sim?” button to activate the failureSetup sub-VI.
- d) ( ) In failureSetup set “During Cycle” to 2.
- e) ( ) Follow the procedure below for each of the following fail-operational tests. These tests should be done once with the “Start Failure” option selected from the “Failure Type” Menu Ring and once with the “Stop Failure” option selected.
  - 1) ( ) Tank 1 Vent valve during Vent Stage.
  - 2) ( ) Tank 2 Vent valve during Vent Stage.
  - 3) ( ) Tank 3 Vent valve during Vent Stage.
  - 4) ( ) Tank 1 Drain valve during Drain Stage.
  - 5) ( ) Tank 2 Drain valve during Drain Stage.
  - 6) ( ) Tank 3 Drain valve during Drain Stage.
  - 7) ( ) Tank 1 Pressurize valve during Pressurize Stage.
  - 8) ( ) Tank 2 Pressurize valve during Pressurize Stage.
  - 9) ( ) Tank 3 Pressurize valve during Pressurize Stage.
  - 10) ( ) Tank 1 Fill valve during Fill Stage.
  - 11) ( ) Tank 2 Fill valve during Fill Stage.

- 12) ( ) Tank 3 Fill valve during Fill Stage.
  - 13) ( ) Tank 1 Vent valve during Fill Stage (if “Fill/Vent Time” is greater than zero).
  - 14) ( ) Tank 2 Vent valve during Fill Stage (if “Fill/Vent Time” is greater than zero).
  - 15) ( ) Tank 3 Vent valve during Fill Stage (if “Fill/Vent Time” is greater than zero).
- f) ( ) For each fail-operational test, print out a copy of the “Fail-Operational Test Summary” page (see end of this document) and record the following information.
- 1) ( ) Date of the Test.
  - 2) ( ) The type of fail-operational test, Start Failure or Stop Failure.
  - 3) ( ) The timing inputs used.
  - 4) ( ) The failure inputs used.
- g) ( ) Make the appropriate settings in failureSetup for each fail-operational test and click the “Save” button.
- h) ( ) If not done already, follow the Pretest Procedures to start the Air and Water systems.
- i) ( ) Fill in the “Run”, “Test Type”, and “Comments” fields as appropriate.
- j) ( ) On the “Data” Tab, click the “Acquire Data” button.
- k) ( ) On the “Settings” Tab, click the “Begin Sequence” button.
- l) ( ) After the Sequence is completed, press the “Acquire Data” button again to halt data acquisition and note the file name in the “Last Saved File” output on the “Settings” Tab under “Data Record File” on the summary page.
- m) ( ) Verify through the ICM that the pressurization, drain, and fill valves are closed.
- n) ( ) Use the ICM manual controls to open all vent valves.

- o) ( ) Examine the data file and quantify the pressure variation that occurs in the water outflow when the system transitions from draining in one SFS tank to draining in another and any other significant variations that occur during the Sequence.

## **Calibration Test Summary**

Date of Test: \_\_\_\_\_

Test Type: \_\_\_\_\_

### **TestGenerator Timing Inputs**

Drain Time: \_\_\_\_\_

Vent Time: \_\_\_\_\_

Fill Time: \_\_\_\_\_

Fill/Vent Time: \_\_\_\_\_

Press Time: \_\_\_\_\_

Overlap: \_\_\_\_\_

Number of Cycles: \_\_\_\_\_

### **Data Files**

Sequence File Name: \_\_\_\_\_

Data Record File: \_\_\_\_\_

### **Notes**



## **Baseline Test Summary**

Date of Test: \_\_\_\_\_

### Timing Inputs

Drain Time: \_\_\_\_\_

Vent Time: \_\_\_\_\_

Fill Time: \_\_\_\_\_

Fill/Vent Time: \_\_\_\_\_

Press Time: \_\_\_\_\_

Overlap: \_\_\_\_\_

Number of Cycles: \_\_\_\_\_

Data Record File: \_\_\_\_\_

### Notes

## **Fail Operational Test Summary**

Date of Test: \_\_\_\_\_

Test Type: \_\_\_\_\_

### **Timing Inputs**

Drain Time: \_\_\_\_\_

Vent Time: \_\_\_\_\_

Fill Time: \_\_\_\_\_

Fill/Vent Time: \_\_\_\_\_

Press Time: \_\_\_\_\_

Overlap: \_\_\_\_\_

Number of Cycles: \_\_\_\_\_

### **Failure Inputs**

Failure Type: \_\_\_\_\_

Valve Type: \_\_\_\_\_

Tank #: \_\_\_\_\_

During Cycle: \_\_\_\_\_

Cycle Stage: \_\_\_\_\_

At Time: \_\_\_\_\_

Data Record File: \_\_\_\_\_

Note

## APPENDIX C

### ANALYSIS RESULTS OF TEST DATA

#### C.1 Baseline Test Analysis

The Baseline tests were analyzed using two methods. The first method used the Fluctuation spreadsheet to calculate average outflow pressure and then to calculate the maximum variation from that average during each tank transition. The time-frame for each transition was defined by the user. The results of this analysis are summarized in the tables below.

Table C.1 First Analysis Results, Zero Overlap Test

Transition	Press Avg	Fluct Max	Fluct Min	Fluct %	Duration	P Start	P End	Fluct Start	Fluct End
1	112.3546	113.9912	112.2554	1.456705	2.732	633	3126	909	922
2	112.3546	113.2901	110.8673	1.323692	2.437	633	3126	1184	1195
3	112.3546	113.398	111.6284	0.92873	3.578	633	3126	1463	1480
4	112.3546	114.3653	111.7996	1.789616	1.755	633	3126	1740	1748
5	112.3546	113.4677	111.2682	0.990722	1.698	633	3126	2017	2025
6	112.3546	113.627	111.6956	1.132507	2.349	633	3126	2295	2306
7	112.3546	113.6355	111.7987	1.140083	1.553	633	3126	2572	2579
8	112.3546	113.2866	111.1287	1.091097	1.498	633	3126	2848	2855
Averages				1.231644	2.2				

Table C.2 First Analysis Results, 0.1 Second Overlap Test

Transition	Press Avg	Fluct Max	Fluct Min	Fluct %	Duration	P Start	P End	Fluct Start	Fluct End
1	116.0283	117.0225	115.3897	0.856875	2.968	664	2184	925	938
2	116.0283	116.6127	112.6732	2.891637	2.001	664	2184	1183	1192
3	116.0283	117.1204	112.2728	3.236758	2.215	664	2184	1441	1451
4	116.0283	116.7629	115.3513	0.633083	3.277	664	2184	1700	1715
5	116.0283	116.5408	112.6894	2.877684	0.925	664	2184	1957	1961
Averages				2.099207	2.277				

Table C.3 First Analysis Results, 0.2 Second Overlap Test

Transition	Press Avg	Fluct Max	Fluct Min	Fluct %	Duration	P Start	P End	Fluct Start	Fluct End
1	117.7921	119.2718	116.8985	1.256263	2.05	618	2155	866	875
2	117.7921	118.0468	114.307	2.958687	1.876	618	2155	1122	1130
3	117.7921	118.8494	114.6813	2.640882	2.48	618	2155	1382	1393
4	117.7921	119.2887	116.1704	1.376682	1.785	618	2155	1639	1647
5	117.7921	118.4202	114.1809	3.065684	2.006	618	2155	1897	1906
Averages				2.25964	2.039				

Table C.4 First Analysis Results, 0.3 Second Overlap Test

Transition	Press Avg	Fluct Max	Fluct Min	Fluct %	Duration	P Start	P End	Fluct Start	Fluct End
1	118.5125	120.1597	117.7708	1.389911	1.678	616	2156	869	876
2	118.5125	119.4069	114.9788	2.981713	1.861	616	2156	1125	1133
3	118.5125	119.1376	115.085	2.892078	1.347	616	2156	1383	1389
4	118.5125	119.9917	117.8106	1.24817	1.573	616	2156	1641	1648
5	118.5125	119.5671	111.7941	5.668971	2.44	616	2156	1894	1905
Averages				2.836169	1.78				

Table C.5 First Analysis Results, 0.4 Second Overlap Test

Transition	Press Avg	Fluct Max	Fluct Min	Fluct %	Duration	P Start	P End	Fluct Start	Fluct End
1	120.2951	121.3068	119.4857	0.840979	1.582	630	2169	884	891
2	120.2951	121.3201	117.5354	2.2941	2.913	630	2169	1139	1152
3	120.2951	121.6053	116.74	2.955375	2.703	630	2169	1396	1408
4	120.2951	121.8378	119.657	1.282399	2.273	630	2169	1653	1663
5	120.2951	121.5494	110.9046	7.806258	3.507	630	2169	1900	1916
Averages				3.035822	2.596				

Table C.6 First Analysis Results, 0.5 Second Overlap Test

Transition	Press Avg	Fluct Max	Fluct Min	Fluct %	Duration	P Start	P End	Fluct Start	Fluct End
1	122.8815	124.106	122.1065	0.996482	2.339	564	2096	817	827
2	122.8815	123.7349	118.6925	3.408974	2.929	564	2096	1070	1083
3	122.8815	123.4513	119.0691	3.102521	1.811	564	2096	1328	1336
4	122.8815	124.6593	121.7077	1.446698	2.524	564	2096	1584	1595
5	122.8815	124.0554	114.4457	6.865035	2.924	564	2096	1833	1846
Averages				3.163942	2.505				

The second analysis used was with the Fluctuation rev2 spreadsheet, an adaptation of the Variations spreadsheet used for analysis of the Fail-Operational tests. This analysis was done with the pressure variation threshold at 1% and 2% of mean outflow pressure. The results are given in the tables below.

Table C.7 Second Analysis Results, Zero Overlap and 1% Threshold

Event	Start	End	Duration	Max Var	Max %
1	55.136	55.525	0.389	113.9912	1.422736
2	102.158	102.738	0.58	111.2201	1.042862
3	110.143	110.466	0.323	110.8673	1.35673
4	212.082	212.617	0.535	110.9068	1.321611
5	220.187	220.573	0.386	114.3653	1.755536
6	275.17	275.492	0.322	111.2682	1.000072
Averages			0.4225	112.1032	1.316591

Table C.8 Second Analysis Results, 0.1 Second Overlap and 1% Threshold

Event	Start	End	Duration	Max Var	Max %
1	102.431	102.86	0.429	114.7574	1.080893
2	110.164	110.417	0.253	112.6732	2.877437
3	164.996	165.244	0.248	112.2728	3.22261
4	211.835	212.517	0.682	114.5251	1.281172
5	274.775	275.027	0.252	112.6894	2.863483
Averages			0.3728	113.3836	2.265119

Table C.9 Second Analysis Results, 0.2 Second Overlap and 1% Threshold

Event	Start	End	Duration	Max Var	Max %
1	0.722	0.93	0.208	119.6299	1.696699
2	55.54	55.749	0.209	119.2718	1.392278
3	102.324	102.759	0.435	116.2463	1.179673
4	110.258	110.512	0.254	114.307	2.828334
5	165.025	165.498	0.473	114.6813	2.510103
6	211.668	212.351	0.683	116.2657	1.163239
7	219.599	220.079	0.48	119.2887	1.406631
8	274.575	274.823	0.248	114.1809	2.935475
Averages			0.37375	116.734	1.889054

Table C.10 Second Analysis Results, 0.3 Second Overlap and 1% Threshold

Event	Start	End	Duration	Max Var	Max %
1	54.893	55.181	0.288	120.1597	1.623803
2	55.486	55.712	0.226	119.6779	1.216279
3	102.515	102.728	0.213	116.9344	1.103999
4	103.161	103.369	0.208	119.4761	1.045604
5	110.047	110.298	0.251	114.9788	2.757906
6	164.61	164.858	0.248	115.085	2.668064
7	211.712	212.152	0.44	116.7967	1.220472
8	219.173	219.418	0.245	119.6663	1.206539
9	219.689	219.898	0.209	119.9917	1.481735
10	267.135	267.782	0.647	119.7406	1.269304
11	273.285	273.503	0.218	119.5671	1.122579
12	273.991	274.235	0.244	111.7941	5.451363
13	274.456	274.665	0.209	119.4428	1.017515
14	321.06	321.484	0.424	116.93	1.10767
Averages			0.290714	117.8744	1.735202

Table C.11 Second Analysis Results, 0.4 Second Overlap and 1% Threshold

Event	Start	End	Duration	Max Var	Max %
1	55.471	55.681	0.21	121.3068	1.202527
2	102.451	102.884	0.433	118.4988	1.140121
3	103.316	103.524	0.208	121.1144	1.041988
4	109.708	110.213	0.505	117.5354	1.943792
5	164.388	164.858	0.47	116.74	2.607438
6	211.656	212.073	0.417	118.4025	1.22046
7	218.675	219.14	0.465	121.8378	1.64553
8	219.426	219.635	0.209	121.7415	1.565191
9	223.056	223.266	0.21	121.0655	1.001192
10	241.148	241.371	0.223	121.2251	1.134348
11	267.111	267.546	0.435	121.6141	1.458908
12	271.357	271.786	0.429	121.217	1.127595
13	271.991	272.42	0.429	121.1478	1.069859
14	273.543	273.799	0.256	110.9046	7.475713
15	274.008	274.218	0.21	121.5494	1.404907
16	321.464	321.906	0.442	121.185	1.100908
Averages			0.346938	119.8178	1.75878

Table C.12 Second Analysis Results, 0.5 Second Overlap and 1% Threshold

Event	Start	End	Duration	Max Var	Max %
1	0.74	0.96	0.22	123.8634	1.371713
2	54.786	55.553	0.767	124.106	1.57027
3	55.774	55.996	0.222	123.5176	1.088698
4	103.223	103.663	0.44	123.9433	1.437071
5	109.446	110.187	0.741	118.6925	2.860215
6	164.18	164.657	0.477	119.0691	2.552021
7	212.004	212.421	0.417	123.5209	1.09143
8	218.367	218.842	0.475	124.6593	2.023043
9	219.12	219.343	0.223	124.4959	1.889345
10	219.564	220.212	0.648	123.6835	1.224492
11	220.419	220.64	0.221	123.5569	1.120888
12	220.864	221.074	0.21	123.5859	1.144587
13	221.281	221.701	0.42	123.5303	1.099079
14	221.909	225.314	3.405	123.761	1.287892
15	225.526	232.556	7.03	123.6544	1.200634
16	232.78	234.497	1.717	123.5335	1.10172
17	234.708	235.577	0.869	123.5822	1.141536
18	235.786	237.489	1.703	123.5299	1.098783
19	237.917	238.125	0.208	123.4261	1.013802
20	238.345	239.2	0.855	123.5052	1.078523
21	239.411	239.619	0.208	123.5014	1.075449
22	239.837	240.49	0.653	123.5667	1.128901
23	240.973	241.394	0.421	123.8703	1.377336
24	241.813	242.461	0.648	123.6441	1.192211
25	242.884	243.3	0.416	123.5979	1.154398
26	243.51	243.933	0.423	123.5166	1.087879
27	244.37	244.788	0.418	123.5225	1.092728
28	246.056	246.487	0.431	123.5564	1.12041
29	246.707	247.351	0.644	123.5892	1.147319
30	247.786	248.206	0.42	123.5529	1.117541
31	248.636	249.491	0.855	123.4599	1.041461
32	251.029	251.248	0.219	123.4235	1.011639
33	251.685	252.101	0.416	123.4855	1.062428
34	255.918	256.345	0.427	123.4712	1.050703
35	261.902	262.11	0.208	123.41	1.00062
36	262.747	262.964	0.217	123.4231	1.011343
37	266.481	268.414	1.933	124.2631	1.698801
38	268.839	269.914	1.075	123.4948	1.070008
39	270.121	270.339	0.218	123.4683	1.048358
40	270.556	270.776	0.22	123.4479	1.031672
41	270.983	271.198	0.215	123.4495	1.032947
42	271.405	272.483	1.078	124.0554	1.52886
43	273.211	273.462	0.251	114.4457	6.335911
44	273.683	273.892	0.209	123.7773	1.301232
45	274.101	276.443	2.342	123.5534	1.117951



Table C.12 Continued

46	276.659	280.68	4.021	123.5126	1.084623
47	280.89	281.513	0.623	123.4857	1.062587
48	281.721	282.165	0.444	123.4608	1.042212
49	282.373	282.583	0.21	123.412	1.002305
50	282.8	285.772	2.972	123.5083	1.081118
51	286.608	286.814	0.206	123.416	1.005515
52	287.465	288.096	0.631	123.4951	1.070236
53	288.512	288.722	0.21	123.4134	1.003375
54	288.939	289.154	0.215	123.4289	1.016078
55	289.374	289.813	0.439	123.4439	1.028394
56	292.11	292.525	0.415	123.4271	1.014598
57	292.964	293.181	0.217	123.4144	1.00424
58	293.387	294.018	0.631	123.4603	1.041825
59	294.226	294.87	0.644	123.4418	1.026618
60	295.312	295.743	0.431	123.6479	1.195352
61	296.192	296.4	0.208	123.4407	1.02573
62	320.902	321.33	0.428	123.7927	1.313844
63	322.195	322.403	0.208	123.4509	1.034062
Averages			0.777556	123.3082	1.290263

Table C.13 Second Analysis Results, Zero Overlap and 2% Threshold

Event	Start	End	Duration	Max Var	Max %
No Events above 2% Threshold measured for this test.					
Averages			NA	NA	<2%

Table C.14 Second Analysis Results, 0.1 Second Overlap and 2% Threshold

Event	Start	End	Duration	Max Var	Max %
1	110.164	110.417	0.253	112.6732	2.877437
2	164.996	165.244	0.248	112.2728	3.22261
3	274.775	275.027	0.252	112.6894	2.863483
Averages			0.251	112.5451	2.987843

Table C.15 Second Analysis Results, 0.2 Second Overlap and 2% Threshold

Event	Start	End	Duration	Max Var	Max %
1	110.258	110.512	0.254	114.307	2.828334
2	165.025	165.289	0.264	114.6813	2.510103
3	274.575	274.823	0.248	114.1809	2.935475
Averages			0.255333	114.3897	2.757971

Table C.16 Second Analysis Results, 0.3 Second Overlap and 2% Threshold

Event	Start	End	Duration	Max Var	Max %
1	110.047	110.298	0.251	114.9788	2.757906
2	164.61	164.858	0.248	115.085	2.668064
3	273.991	274.235	0.244	111.7941	5.451363
Averages			0.247667	113.9526	3.625778

Table C.17 Second Analysis Results, 0.4 Second Overlap and 2% Threshold

Event	Start	End	Duration	Max Var	Max %
1	164.388	164.648	0.26	116.74	2.607438
2	273.543	273.799	0.256	110.9046	7.475713
Averages			0.258	113.8223	5.041575

Table C.18 Second Analysis Results, 0.5 Second Overlap and 2% Threshold

Event	Start	End	Duration	Max Var	Max %
1	109.706	109.966	0.26	118.6925	2.860215
2	164.18	164.434	0.254	119.0691	2.552021
3	218.586	218.842	0.256	124.6593	2.023043
4	273.211	273.462	0.251	114.4457	6.335911
Averages			0.25525	119.2166	3.442798

## C.2 Fail-Operational Test Analysis

The Fail-Operational Test data were analyzed with the Variations spreadsheet. The average outflow pressure was calculated automatically for the first cycle (the program assumed a two-cycle test) and each data point was compared to that average. Any data point that was outside 2% of the average pressure triggered a pressure event report. The events were classified as follows. For the Duration value: Brief – one data point (<0.5 seconds), Short – a few data points (0.5 to 1 second), Long – 1 to 5 seconds, and Extended – >5 seconds. For the maximum variation as percentage of mean outflow pressure: Minor – <5%, Large – 5-90%, Major – >90%. A test was considered a success if there were only Minor pressure events. The results of each test are summarized in the tables below.

Table C.19 Start Failure in Vent Valve of Tank 1 During Vent Stage

Data Filename:	SFS 12-5-2007_18.txt					<u>Timing Inputs</u>		<u>Failure Settings</u>		
First Cycle Average Pressure:	108.6435	psig				Drain	55	Type	Start	
Pressure Variation Threshold:	2	%				Vent	22	Valve	Vent	
Failure Occurs at Time:	220	sec				Fill	25	Tank	1	
						Fill/Vent	15	Cycle	2	
						Press	7.5	Stage	Vent	
						Overlap	0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.146	110.429	0.283	105.3342	3.046088			Brief	Minor	Drain 3
2	165.199	165.461	0.262	104.0163	4.259055			Brief	Minor	Drain 4
3	275.156	275.413	0.257	103.9418	4.327697			Brief	Minor	Drain 6
								Test Overall: Success		

Table C.20 Start Failure in Drain Valve of Tank 1 During Drain Stage

Data Filename:		SFS 12-5-2007_19.txt	Timing Inputs				Failure Settings			
First Cycle Average Pressure:	109.3764	psig		Drain	55		Type	Start		
Pressure Variation Threshold:	2	%		Vent	22		Valve	Drain		
Failure Occurs at Time:	165	sec		Fill	25		Tank	1		
				Fill/Vent	15		Cycle	2		
				Press	7.5		Stage	Drain		
				Overlap	0					
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.105	110.36	0.255	104.7618	4.219009			Brief	Minor	Drain 3
2	165.03	165.591	0.561	-1.81155	101.6563	YES	After	Short	Major	Drain 4
3	220.109	220.349	0.24	105.8354	3.237406			Brief	Minor	Drain 5
4	275.042	275.285	0.243	104.8227	4.163314			Brief	Minor	Drain 6
									Test Overall: Error	

Table C.21 Start Failure in Press Valve of Tank 1 During Drain Stage

Data Filename:		SFS 12-5-2007_110.txt	Timing Inputs				Failure Settings			
First Cycle Average Pressure:	111.0025	psig		Drain	55		Type	Start		
Pressure Variation Threshold:	2	%		Vent	22		Valve	Press		
Failure Occurs at Time:	165	sec		Fill	25		Tank	1		
				Fill/Vent	15		Cycle	2		
				Press	7.5		Stage	Drain		
				Overlap	0					
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	55.099	55.409	0.31	108.4314	2.316195			Brief	Minor	Drain 2
2	110.015	110.282	0.267	106.2555	4.276452			Brief	Minor	Drain 3
3	165.197	165.753	0.556	105.847	4.644468	YES	After	Short	Minor	Drain 4
4	220.268	220.513	0.245	107.8071	2.878693			Brief	Minor	Drain 5
5	275.226	275.468	0.242	106.7317	3.847494			Brief	Minor	Drain 6
									Test Overall: Success	

Table C.22 Start Failure in Press Valve of Tank 1 During Press Stage

Data Filename:		SFS 12-5-2007_1.1.txt				Timing Inputs		Failure Settings		
First Cycle Average Pressure:		112.5705		psig		Drain	55	Type	Start	
Pressure Variation Threshold:		2		%		Vent	22	Valve	Press	
Failure Occurs at Time:		267		sec		Fill	25	Tank	1	
						Fill/Vent	15	Cycle	2	
						Press	7.5	Stage	Press	
						Overlap	0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.072	110.349	0.277	108.6242	3.505695			Brief	Minor	Drain 3
2	165.158	165.419	0.261	108.4523	3.658377			Brief	Minor	Drain 4
3	220.219	220.488	0.269	109.9528	2.325406			Brief	Minor	Drain 5
4	275.056	275.306	0.25	108.4848	3.629516			Brief	Minor	Drain 6
								Test Overall: Success		

Table C.23 Start Failure in Fill Valve of Tank 1 During Fill Stage

Data Filename:			SFS 12-5-2007_1.2.txt			Timing Inputs		Failure Settings		
First Cycle Average Pressure:			114.5259 psig			Drain	55	Type	Start	
Pressure Variation Threshold:			2 %			Vent	22	Valve	Fill	
Failure Occurs at Time:			242 sec			Fill	25	Tank	1	
						Fill/Vent	15	Cycle	2	
						Press	7.5	Stage	Fill	
						Overlap	0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.063	110.335	0.272	110.2884	3.69999			Brief	Minor	Drain 3
2	165.112	165.388	0.276	111.9053	2.28818			Brief	Minor	Drain 4
3	220.067	220.323	0.256	111.3115	2.806728			Brief	Minor	Drain 5
4	275.124	275.368	0.244	110.7132	3.329089			Brief	Minor	Drain 6
								Test Overall: Success		

Table C.24 Start Failure in Vent Valve of Tank 1 During Fill Stage

Data Filename:		SFS 12-5-2007_1.3.txt				Timing Inputs		Failure Settings		
First Cycle Average Pressure:		114.7878		psig		Drain	55	Type	Start	
Pressure Variation Threshold:		2		%		Vent	22	Valve	Vent	
Failure Occurs at Time:		252		sec		Fill	25	Tank	1	
						Fill/Vent	15	Cycle	2	
						Press	7.5	Stage	Fill	
						Overlap	0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.193	110.461	0.268	110.0346	4.140821			Brief	Minor	Drain 3
2	165.183	165.455	0.272	110.2943	3.914586			Brief	Minor	Drain 4
3	220.053	220.324	0.271	111.8857	2.528218			Brief	Minor	Drain 5
4	275.158	275.412	0.254	110.0576	4.120829			Brief	Minor	Drain 6
								Test Overall: Success		

Table C.25 Stop Failure in Vent Valve of Tank 1 During Vent Stage

Data Filename:		SFS 12-5-2007_1.4.txt				Timing Inputs		Failure Settings		
First Cycle Average Pressure:		115.1333 psig				Drain	55	Type	Stop	
Pressure Variation Threshold:		2 %				Vent	22	Valve	Vent	
Failure Occurs at Time:		242 sec				Fill	25	Tank	1	
						Fill/Vent	15	Cycle	2	
						Press	7.5	Stage	Vent	
						Overlap	0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.024	110.294	0.27	110.6875	3.861486			Brief	Minor	Drain 3
2	165.022	165.286	0.264	110.405	4.106853			Brief	Minor	Drain 4
3	220.065	220.326	0.261	111.7819	2.910894			Brief	Minor	Drain 5
4	275.133	275.382	0.249	110.7086	3.843125			Brief	Minor	Drain 6
								Test Overall: Success		



Table C.28 Stop Failure in Press Valve of Tank 1 During Press Stage

Data Filename:			SFS 12-5-2007_1.7.txt			Timing Inputs		Failure Settings		
First Cycle Average Pressure:			117.239	psig		Drain	55	Type	Stop	
Pressure Variation Threshold:			2	%		Vent	22	Valve	Press	
Failure Occurs at Time:			274.5	sec		Fill	25	Tank	1	
						Fill/Vent	15	Cycle	2	
						Press	7.5	Stage	Press	
						Overlap	0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	55.133	55.436	0.303	114.572	2.274863			Brief	Minor	Drain 2
2	110.077	110.355	0.278	113.7179	3.003389			Brief	Minor	Drain 3
3	165.091	165.352	0.261	112.906	3.695924			Brief	Minor	Drain 4
4	220.16	220.427	0.267	114.2427	2.55573			Brief	Minor	Drain 5
5	275.434	275.696	0.262	112.3692	4.153786	YES	After	Brief	Minor	Drain 6
									Test Overall: Success	

Table C.29 Stop Failure in Fill Valve of Tank 1 During Fill Stage

Data Filename:			SFS 12-5-2007_1.8.txt			Timing Inputs		Failure Settings		
First Cycle Average Pressure:			117.0999 psig			Drain	55	Type	Stop	
Pressure Variation Threshold:			2 %			Vent	22	Valve	Fill	
Failure Occurs at Time:			267 sec			Fill	25	Tank	1	
						Fill/Vent	15	Cycle	2	
						Press	7.5	Stage	Fill	
						Overlap	0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.134	110.389	0.255	112.4188	3.997594			Brief	Minor	Drain 3
2	165.046	165.318	0.272	114.2899	2.399682			Brief	Minor	Drain 4
3	220.007	220.278	0.271	114.0569	2.59867			Brief	Minor	Drain 5
4	275.092	275.347	0.255	112.3168	4.084652			Brief	Minor	Drain 6
								Test Overall: Success		



Table C.30 Stop Failure in Vent Valve of Tank 1 During Fill Stage

Data Filename:		SFS 12-5-2007_1.9.txt				Timing Inputs		Failure Settings		
First Cycle Average Pressure:		117.3688		psig		Drain	55	Type	Stop	
Pressure Variation Threshold:		2		%		Vent	22	Valve	Vent	
Failure Occurs at Time:		267		sec		Fill	25	Tank	1	
						Fill/Vent	15	Cycle	2	
						Press	7.5	Stage	Fill	
						Overlap	0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.014	110.281	0.267	112.897	3.810005			Brief	Minor	Drain 3
2	165.12	165.379	0.259	112.9471	3.767346			Brief	Minor	Drain 4
3	220.091	220.349	0.258	113.7621	3.072971			Brief	Minor	Drain 5
4	275.004	275.274	0.27	112.8481	3.851693			Brief	Minor	Drain 6
								Test Overall: Success		

Table C.31 Start Failure in Vent Valve of Tank 2 During Vent Stage

Data Filename:			SFS 12-5-2007_1.10.txt			Timing Inputs		Failure Settings		
First Cycle Average Pressure:			117.5529	psig		Drain	55	Type	Start	
Pressure Variation Threshold:			2	%		Vent	22	Valve	Vent	
Failure Occurs at Time:			275	sec		Fill	25	Tank	2	
						Fill/Vent	15	Cycle	2	
						Press	7.5	Stage	Vent	
						Overlap	0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.147	110.421	0.274	113.7665	3.221032			Brief	Minor	Drain 3
2	165.21	165.473	0.263	113.3657	3.562008			Brief	Minor	Drain 4
3	220.046	220.313	0.267	114.6838	2.440736			Brief	Minor	Drain 5
								Test Overall: Success		

Table C.32 Start Failure in Drain Valve of Tank 2 During Drain Stage

Data Filename:		SFS 12-5-2007_1.t1.txt				Timing Inputs		Failure Settings			
First Cycle Average Pressure:		117.7549		psig		Drain		55		Type	Start
Pressure Variation Threshold:		2		%		Vent		22		Valve	Drain
Failure Occurs at Time:		220		sec		Fill		25		Tank	2
						Fill/Vent		15		Cycle	2
						Press		7.5		Stage	Drain
						Overlap		0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.											
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location	
1	110.152	110.422	0.27	113.4093	3.690432			Brief	Minor	Drain 3	
2	165.062	165.331	0.269	112.9673	4.065737			Brief	Minor	Drain 4	
3	220.147	220.748	0.601	19.69357	83.2758	YES	After	Short	Large	Drain 5	
4	275.243	275.511	0.268	113.8595	3.308111			Brief	Minor	Drain 6	
Test Overall: Error											

Table C.33 Start Failure in Press Valve of Tank 2 During Drain Stage

Data Filename:			SFS 12-5-2007_1.t2.txt			Timing Inputs		Failure Settings			
First Cycle Average Pressure:			117.7734	psig		Drain	55	Type	Start		
Pressure Variation Threshold:			2	%		Vent	22	Valve	Press		
Failure Occurs at Time:			220	sec		Fill	25	Tank	2		
						Fill/Vent	15	Cycle	2		
						Press	7.5	Stage	Drain		
						Overlap	0				
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.											
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location	
1	110.162	110.434	0.272	113.6977	3.460638			Brief	Minor	Drain 3	
2	220.027	220.634	0.607	110.7268	5.983159	YES	After	Short	Large	Drain 5	
3	275.108	275.37	0.262	114.2119	3.024032			Brief	Minor	Drain 6	
								Test Overall: Error			

Table C.34 Start Failure in Press Valve of Tank 2 During Press Stage

Data Filename:		SFS 12-5-2007_1.t3.txt				Timing Inputs		Failure Settings		
First Cycle Average Pressure:		118.2007		psig		Drain	55	Type	Start	
Pressure Variation Threshold:		2		%		Vent	22	Valve	Press	
Failure Occurs at Time:		322		sec		Fill	25	Tank	2	
						Fill/Vent	15	Cycle	2	
						Press	7.5	Stage	Press	
						Overlap	0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.062	110.324	0.262	113.5614	3.924916			Brief	Minor	Drain 3
2	165.125	165.393	0.268	113.8588	3.673302			Brief	Minor	Drain 4
3	220.209	220.469	0.26	115.1683	2.56547			Brief	Minor	Drain 5
4	275.165	275.412	0.247	113.9299	3.613199			Brief	Minor	Drain 6
								Test Overall: Success		

Table C.35 Start Failure in Fill Valve of Tank 2 During Fill Stage

Data Filename:		SFS 12-5-2007_1.t4.txt				Timing Inputs		Failure Settings		
First Cycle Average Pressure:		118.3031		psig		Drain	55	Type	Start	
Pressure Variation Threshold:		2		%		Vent	22	Valve	Fill	
Failure Occurs at Time:		297		sec		Fill	25	Tank	2	
						Fill/Vent	15	Cycle	2	
						Press	7.5	Stage	Fill	
						Overlap	0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.164	110.446	0.282	115.1855	2.635222			Brief	Minor	Drain 3
2	165.003	165.274	0.271	115.2252	2.601716			Brief	Minor	Drain 4
3	220.13	220.387	0.257	115.1908	2.630755			Brief	Minor	Drain 5
4	275.087	275.323	0.236	114.375	3.320376			Brief	Minor	Drain 6
								Test Overall: Success		

Table C.36 Start Failure in Vent Valve of Tank 2 During Fill Stage

Data Filename:		SFS 12-5-2007_1.t5.txt				Timing Inputs		Failure Settings		
First Cycle Average Pressure:		118.7825		psig		Drain		55	Type	Start
Pressure Variation Threshold:		2		%		Vent		22	Valve	Vent
Failure Occurs at Time:		307		sec		Fill		25	Tank	2
						Fill/Vent		15	Cycle	2
						Press		7.5	Stage	Fill
						Overlap		0		
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.024	110.287	0.263	113.7215	4.260717			Brief	Minor	Drain 3
2	165.086	165.354	0.268	114.7999	3.352794			Brief	Minor	Drain 4
3	220.128	220.378	0.25	114.3933	3.695159			Brief	Minor	Drain 5
4	275.083	275.325	0.242	114.425	3.66844			Brief	Minor	Drain 6
								Test Overall: Success		

Table C.37 Start Failure in Vent Valve of Tank 3 During Vent Stage

Data Filename:		SFS 12-5-2007_1.t6.txt				Timing Inputs		Failure Settings		
First Cycle Average Pressure:		119.3441 psig				Drain	55	Type	Start	
Pressure Variation Threshold:		2 %				Vent	22	Valve	Vent	
Failure Occurs at Time:		330 sec				Fill	25	Tank	3	
						Fill/Vent	15	Cycle	2	
						Press	7.5	Stage	Vent	
						Overlap	0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.102	110.378	0.276	115.4436	3.268225			Brief	Minor	Drain 3
2	165.196	165.467	0.271	115.6166	3.123324			Brief	Minor	Drain 4
3	220.165	220.419	0.254	115.7351	3.023964			Brief	Minor	Drain 5
4	275.065	275.319	0.254	114.1781	4.328619			Brief	Minor	Drain 6
								Test Overall: Success		

Table C.38 Start Failure in Drain Valve of Tank 3 During Drain Stage

Data Filename:		SFS 12-5-2007_1.t7.txt	Timing Inputs				Failure Settings			
First Cycle Average Pressure:	119.1094	psig		Drain	55		Type	Start		
Pressure Variation Threshold:	2	%		Vent	22		Valve	Drain		
Failure Occurs at Time:	275	sec		Fill	25		Tank	3		
				Fill/Vent	15		Cycle	2		
				Press	7.5		Stage	Drain		
				Overlap	0					
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.14	110.411	0.271	114.7274	3.678976			Brief	Minor	Drain 3
2	165.193	165.449	0.256	114.5575	3.821618			Brief	Minor	Drain 4
3	220.147	220.419	0.272	115.5812	2.962147			Brief	Minor	Drain 5
4	275.142	275.721	0.579	21.64698	81.82597	YES	After	Short	Large	Drain 6
								Test Overall: Error		

Table C.39 Start Failure in Press Valve of Tank 3 During Drain Stage

Data Filename:		SFS 12-5-2007_1.t8.txt	Timing Inputs				Failure Settings			
First Cycle Average Pressure:	111.6452	psig		Drain	55		Type	Start		
Pressure Variation Threshold:	2	%		Vent	22		Valve	Press		
Failure Occurs at Time:	275	sec		Fill	25		Tank	3		
				Fill/Vent	15		Cycle	2		
				Press	7.5		Stage	Drain		
				Overlap	0					
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.										
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location
1	110.093	110.357	0.264	114.0339	4.321642			Brief	Minor	Drain 3
2	220.068	220.336	0.268	115.4258	3.153804			Brief	Minor	Drain 5
3	275.082	275.671	0.589	111.9756	6.048672	YES	After	Short	Large	Drain 6
								Test Overall: Error		



Data Filename:			SFS 12-6-2007_4.txt			Timing Inputs		Failure Settings			
First Cycle Average Pressure:			109.5945 psig			Drain		55		Type	Stop
Pressure Variation Threshold:			2 %			Vent		22		Valve	Press
Failure Occurs at Time:			329.5 sec			Fill		25		Tank	2
						Fill/Vent		15		Cycle	2
						Press		7.5		Stage	Press
						Overlap		0			
Pressure Variation Events that deviate by more that the threshold percentage from the mean outflow pressure.											
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location	
1	110.081	110.343	0.262	104.7625	4.408949			Brief	Minor	Drain 3	
2	165.008	165.285	0.277	107.3173	2.077847			Brief	Minor	Drain 4	
3	220.041	220.313	0.272	107.381	2.019674			Brief	Minor	Drain 5	
4	275.174	275.425	0.251	105.429	3.800774			Brief	Minor	Drain 6	
								Test Overall: Success			

Data Filename:			SFS 12-6-2007_6.txt			Timing Inputs		Failure Settings			
First Cycle Average Pressure:			111.4599 psig			Drain		55		Type	Stop
Pressure Variation Threshold:			2 %			Vent		22		Valve	Vent
Failure Occurs at Time:			322 sec			Fill		25		Tank	2
						Fill/Vent		15		Cycle	2
						Press		7.5		Stage	Fill
						Overlap		0			
Pressure Variation Events that deviate by more that the threshold						percentage from the mean outflow pressure.					
Event	Start	End	Duration	Max Var	Max %	Near Fail?	Began	Time	Pressure	Location	
1	110.185	110.465	0.28	108.5114	2.64536			Brief	Minor	Drain 3	
2	165.174	165.452	0.278	107.3362	3.699678			Brief	Minor	Drain 4	
3	220.126	220.407	0.281	108.752	2.429465			Brief	Minor	Drain 5	
4	275.165	275.428	0.263	106.7125	4.259268			Brief	Minor	Drain 6	
								Test Overall: Success			



## REFERENCES

- <sup>1</sup>Ring, E., *Rocket Propellant and Pressurization Systems*, Prentice-Hall, Inc., Englewood Cliffs, 1964.
- <sup>2</sup>Bailey, M., and Ramirez, P., “Pressurization Systems for Liquid Rockets,” NASA SP-8112, October 1975.
- <sup>3</sup>Sutton, G., and Biblarz, O., *Rocket Propulsion Elements*, 7th Ed., J. Wiley & Sons, New York, 2001.
- <sup>4</sup>Huether, Jacob E., Spears, James M., McCleskey, Carey M., and Rhodes, Russel E., “Space Shuttle to Reusable Launch Vehicle,” Presented at the Thirty-Second Space Congress Canaveral Council of Technical Societies, 1995.  
<http://science.ksc.nasa.gov/shuttle/nexgen/rlvhq10.htm>
- <sup>5</sup>Harrington, Steven M., “Dual Chamber Pump and Method,” United States Patent 7,007,456, granted March 7<sup>th</sup>, 2006.
- <sup>6</sup>Blackmon, J., and Lanning, M., “Reciprocating Feed System for Fluids,” United States Patent 6,314,978, granted November 13<sup>th</sup>, 2001.
- <sup>7</sup>Harrington, S., “Pistonless Dual Chamber Rocket Fuel Pump,” AIAA Paper 2003-4479, July 2003.
- <sup>8</sup>Eddleman, D., Blackmon, J., and Moser, M., “Reciprocating Feed System for In-Space Propulsion System,” AIAA Paper 2003-4500, July 2003.
- <sup>9</sup>Blackmon, J., and Eddleman, D. “Reciprocating Feed Development Status,” AIAA Paper 2005-3648, July 2005.
- <sup>10</sup>Eddleman, D., “Reciprocating Propellant Feed System Development Program,” University of Alabama in Huntsville Mechanical and Aerospace Engineering Department Master’s Thesis, 2006.

- <sup>11</sup>Eddleman, D., Blackmon, J., and Morton, C., “Analytical Assessment of the Reciprocating Feed System,” AIAA Paper 2006-4759, July 2006.
- <sup>12</sup>Eddleman, D., Blackmon, J., and Morton, C., “Experimental Assessment of the Reciprocating Feed System,” AIAA Paper 2006-5266, July 2006.
- <sup>13</sup>Mensurati, E., “Prototype System Requirements Document 15,000 lbf Engine,” Attachment A1, NASA NNC06ZPT003R, 2005.
- <sup>14</sup>Krzycki, Leroy J., “How to Design, Build and Test Small Liquid-Fuel Rocket Engines,” 1967.
- <sup>15</sup>Baumeister, Theodore, and Marks, Lionel S., *Standard Handbook for Mechanical Engineers*, 7<sup>th</sup> Ed., McGraw-Hill Professional, 1958.
- <sup>16</sup><http://www.astronautix.com>
- <sup>17</sup>Cequel: Chemical Equilibrium in Excel by Software and Engineering Associates, Inc. <http://www.seainc.com/cequel.d.html>
- <sup>18</sup>Hill, Philip, and Peterson, Carl, *Mechanics and Thermodynamics of Propulsion*, 2<sup>nd</sup> Ed., Addison-Wesley Publishing Company, 1992.
- <sup>19</sup>[http://en.wikipedia.org/wiki/Saturn\\_V](http://en.wikipedia.org/wiki/Saturn_V)
- <sup>20</sup>“Apollo Systems Description Vol. II Saturn Launch Vehicles,” Marshall Space Flight Center, Technical Memorandum X-881, 1964.
- <sup>21</sup>Bilstein, Roger E., *Stages to Saturn: A Technological History of the Apollo/Saturn Launch Vehicles*, University Press of Florida, 1980.
- <sup>22</sup>U.S. Department of Defense, “Reliability Modeling and Prediction,” MIL-STD-756B, November 18, 1981.
- <sup>23</sup>Coleman, Hugh W., and Steele, W. Glenn, *Experimentation and Uncertainty Analysis for Engineers*, 2<sup>nd</sup> Ed., J. Wiley & Sons, New York, 1999.
- <sup>24</sup>Polovko, A., *Fundamentals of Reliability*, Academic Press, New York, 1968.
- <sup>25</sup>Bettely, I.G., “The Addition Law For Expectations”, [www.jstor.org](http://www.jstor.org)

<sup>26</sup>Pieruschka, E., *Principles of Reliability*, Prentice Hall, Inc., Englewood Cliffs, 1963.

<sup>27</sup>[http://en.wikipedia.org/wiki/Error\\_function](http://en.wikipedia.org/wiki/Error_function)

<sup>28</sup>“Flow of Fluids Through Valves, Fittings, and Pipe,” Crane Co., Technical Paper No. 410, Long Beach, CA, December 2001.

<sup>29</sup>Arfken, George B., and Weber, Hans J., *Mathematical Methods for Physicists*, 4<sup>th</sup> Ed., Academic Press, San Diego, 1995.

<sup>30</sup>John, James E.A., and Haberman, William L., *Introduction to Fluid Mechanics*, 2<sup>nd</sup> Ed., Prentice-Hall, 1980.

<sup>31</sup>White, Frank M., *Fluid Mechanics*, 5<sup>th</sup> Ed., McGraw-Hill Science/Engineering/Math, 2003.

<sup>32</sup>[http://en.wikipedia.org/wiki/Flow\\_coefficient](http://en.wikipedia.org/wiki/Flow_coefficient)