2018-06

UPGRADE TRANSONIC COMPRESSOR TEST RIG DATA ACQUISITION SYSTEM

Belna, Thomas D.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

UPGRADE TRANSONIC COMPRESSOR TEST RIG DATA ACQUISITION SYSTEM

by

Thomas D. Belna

June 2018

Thesis Advisor: Anthony J. Gannon
Co-Advisor: Garth V. Hobson

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2018	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE UPGRADE TRANSONIC COMPRESSOR TEST RIG DATA ACQUISITION SYSTEM	**5. FUNDING NUMBERS**
6. AUTHOR(S) Thomas D. Belna	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

This thesis presents an upgraded data acquisition system (DAS) for the transonic compressor test rig (TCR). The legacy DAS acquired and processed pressure, temperature, and rotational speed data. The data were mass averaged, and the performance of the compressor was measured and compared to that acquired by the legacy system. Total-to-total pressure ratio and isentropic efficiency versus corrected mass flow rate were compared at speeds of 70%, 80%, 85%, and 90% of the design speed. The legacy DAS was robust but outdated compared to currently available technology.

This thesis documents new hardware from National Instruments that was procured and integrated into the DAS. New software programming using commercial-off-the-shelf- software (NI-MAX, DsaLink3, and MATLAB) for acquiring data, processing data, displaying results, and generating reports was also implemented and documented.

Data output files between the legacy DAS and new DAS were within reasonable tolerances and all functionality of the system was maintained.

14. SUBJECT TERMS transonic, compressor, data acquisition, DAS, TCR	**15. NUMBER OF PAGES** 135
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

UPGRADE TRANSONIC COMPRESSOR TEST RIG DATA ACQUISITION SYSTEM

Thomas D. Belna
Lieutenant, United States Navy
BS, Boston University, 2012

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2018

Approved by: Anthony J. Gannon
Advisor

Garth V. Hobson
Co-Advisor

Garth V. Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis presents an upgraded data acquisition system (DAS) for the transonic compressor test rig (TCR). The legacy DAS acquired and processed pressure, temperature, and rotational speed data. The data were mass averaged, and the performance of the compressor was measured and compared to that acquired by the legacy system. Total-to-total pressure ratio and isentropic efficiency versus corrected mass flow rate were compared at speeds of 70%, 80%, 85%, and 90% of the design speed. The legacy DAS was robust but outdated compared to currently available technology.

This thesis documents new hardware from National Instruments that was procured and integrated into the DAS. New software programming using commercial-off-the-shelf- software (NI-MAX, DsaLink3, and MATLAB) for acquiring data, processing data, displaying results, and generating reports was also implemented and documented.

Data output files between the legacy DAS and new DAS were within reasonable tolerances and all functionality of the system was maintained.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
A. MOTIVATION....1
B. TRANSONIC COMPRESSOR TEST RIG DESIGN AND INSTRUMENTATION....1

II. LEGACY DATA ACQUISITION SYSTEM....5
A. HP 75000 SERIES B VXI MAINFRAME....5
B. HPVEE SOFTWARE....6
C. 32-BIT WINDOWS XP COMPUTER....7
D. SYSTEM CONFIGURATION OVERVIEW....7
E. LEGACY DAS PERFORMANCE MAPS....8

III. NEW DATA ACQUISITION SYSTEM....11
A. REQUIREMENTS OF UPDATED DATA ACQUISITION SYSTEM....11
B. PROCUREMENT OF UPDATED DATA ACQUISITION SYSTEM....11
1. Digital Sensor Array....12
2. Chassis....13
3. Temperature Module....13
4. Rotational Speed Module....14
5. MATLAB Software....15
6. 64-bit WINDOWS 10 Computer....15

IV. HARDWARE IMPLEMENTATION....17
A. DSA 3217 PRESSURE BRICK....17
B. NI CDAQ 9188XT CHASSIS AND MODULES....18
C. SYSTEM CONFIGURATION OVERVIEW....19

V. DATA REDUCTION....21

VI. HARDWARE COMMUNICATION....23
A. PRESSURE MEASUREMENT PROGRAM FLOW....23
B. TEMPERATURE MEASUREMENT PROGRAM FLOW....25
C. ROTATIONAL SPEED MEASUREMENT PROGRAM FLOW....27
D. COMBINED PROGRAM FLOW....28

VII. RESULTS 35

VIII. CONCLUSIONS AND RECOMMENDATIONS 39
A. CONCLUSIONS 39
B. RECOMMENDATIONS 39

APPENDIX A. TEST INSTRUMENT PLACEMENT 41

APPENDIX B. DATA REDUCTION 45

APPENDIX C. INSTRUMENTATION SETUP 47

APPENDIX D. INDIVIDUAL COMPONENT DESCRIPTION AND MATLAB CODE 53
A. MATLAB FUNCTIONS 53
B. DSA 3217 PRESSURE BRICK 55
C. NI 9214 THERMOCOUPLE MODULE 57
D. NI 9402 COUNTER-TOTALIZER MODULE 59

APPENDIX E. DAS DESCRIPTION AND MATLAB CODE 61
A. DAS PROGRAM DESCRIPTION 61
B. DSA MATLAB CODE 63
1. Main.m 63
2. Pparameters_multibrick.m 77
3. Calc_Contstants.m 80
4. Constants.m 83
5. Run_Startmulti.m 84
6. Save_Run.m 99
7. Run_Stopmulti.m 99
8. Reset_num.m 100
9. Pulsewidth_main.m 101
10. Temp_readings_main.m 101
11. XSteam.m 102

APPENDIX F. DAS OUTPUT FILES 103

LIST OF REFERENCES 115

INITIAL DISTRIBUTION LIST 117

LIST OF FIGURES

Figure 1. Transonic Compressor Test Rig Design. Source: [4]. 2

Figure 2. Probe Positions of the Test Section. Source: [4]....2

Figure 3. Isometric Air Flow Diagram....3

Figure 4. Split View Air Flow Diagram....3

Figure 5. (a) T-S Diagram, (b) Pressure Ratio Map, and (c) Isentropic Efficiency Map....4

Figure 6. HP 75000 Series B VXI Mainframe....5

Figure 7. HPVEE Software GUI....6

Figure 8. Schematic of Legacy DAS. Adapted from [7]....7

Figure 9. Pressure Ratio Map....8

Figure 10. Isentropic Efficiency Map....9

Figure 11. DSA 3217/16Px “Pressure Brick” 12

Figure 12. NI cDAQ-9188XT Chassis....13

Figure 13. NI 9214 High-Density Thermocouple Module. Source: [9]....14

Figure 14. NI 9402 Digital Input/Output Module. Source: [10]. 15

Figure 15. Pressure Data Collection Protocol 18

Figure 16. Schematic of Upgraded DAS....19

Figure 17. Pressure Measurement Flow Diagram....23

Figure 18. Pressure Measurement Bar Graph 24

Figure 19. Pressure Measurement Raw Data 24

Figure 20. Temperature Measurement Flow Diagram 25

Figure 21. Temperature Measurement Bar Graph....26

Figure 22. Temperature Measurement Raw Data 26

Figure 23. Rotational Speed Measurement Flow Diagram 27

Figure 24. RPM Data.....28

Figure 25. DAS Initialization Flow Diagram.....30

Figure 26. HMI Figure 31

Figure 27. User Input Constants Figure 31

Figure 28. Raw Pressure and Temperature Data Figure 32

Figure 29. Data Collection Process Flow Diagram.....33

Figure 30. User Input Constants Figure Disabled.....36

Figure 31. HMI Figure with Adjusted P_{o3} and T_{o3} Data 36

Figure 32. Raw Pressure and Temperature Data Figure with Adjusted P_{o3} and T_{o3} Data.....37

Figure 33. Pressure and Temperature Instrumentation Ports for Outlet Casing 41

Figure 34. Diagram of Pressure and Temperature Instrumentation Ports for Outlet Casing. Source: [7].....42

Figure 35. NI MAX Home Screen 47

Figure 36. NI MAX Network Settings.....48

Figure 37. PC IPv4 Properties.....48

Figure 38. DSA Link 3 Connection Display.....49

Figure 39. DSA Link 3 Bar Graph Display.....49

Figure 40. Test and Measurement Tool App New Object Creation.....50

Figure 41. Test and Measurement Tool App Communication Tab.....51

Figure 42. Test and Measurement Tool App Session Log Tab.....52

LIST OF TABLES

Table 1. MATLAB Scripts.....29

Table 2. Pressure and Temperature Port Position. Source: [13]. 43

Table 3. Main MATLAB Functions 53

Table 4. MATLAB Script Overview 61

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AS1	Axial Station 1
AS3	Axial Station 3
ASCII	American Standard Code for Information Interchange
DAS	Data acquisition system
FPS	Frames per second
GUI	Graphic user interface
HMI	Human machine interface
HPVEE	Data acquisition software program created by Hewlett Packard
IP	Internet Protocol
MATLAB	Programming language and software developed by MathWorks.
NI cDAQ	National Instruments Compact Data Acquisition
NI MAX	National Instruments Measurement and Automation software
NPS	Naval Postgraduate School
OS	Operating system
PC	Personal computer
TCP	Transmission Control Protocol
TCR	Transonic Compressor Test Rig
UDP	User Datagram Protocol

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

It is with immense gratitude that I acknowledge the support and help of my principal advisors, Dr. Anthony Gannon and Dr. Garth Hobson. Their mentorship and wealth of knowledge was fundamental to ensuring my work was focused and met the key objectives. Their enthusiasm toward my thesis made this a truly unforgettable experience. I would like to recognize John Gibson for his support and for being instrumental to the installation of the system.

I would also like to thank my wife, Grace, for her enduring love and support. This has been a challenging process and without her constant encouragement, this would not have been possible.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

The Naval Postgraduate School (NPS) has frequently used the transonic compressor test rig (TCR) to test new axial transonic compressor rotors and casings for design and performance improvements [1]. Technology has improved the ability to create more detailed designs and has led to an increase in testing at NPS. While the rotors and casings produced are becoming more advanced, the data acquisition system (DAS) used to measure the performance of these products has become slow and outdated in comparison to currently available DAS products.

This work involves the procurement, design, and implementation of upgrades to the current DAS. This report documents replacement hardware for measuring temperature and rotational speed as well as the source code executed to acquire data, process data, display results, and generate reports.

B. TRANSONIC COMPRESSOR TEST RIG DESIGN AND INSTRUMENTATION

The TCR design and instrumentation description from previous work by Drayton [2], Grossman [3] and Hobson et al. [4] is summarized below. The TCR diagram in Figure 1 consists of a transonic axial rotor mounted on a single shaft with two opposed rotor, single stage air-operated drive turbines. The turbines are powered by supply air from an Allis Chalmers compressor. Atmospheric air is drawn into the inlet plenum and throttled via an electrically actuated butterfly valve; it then passes through the settling chamber, flow rate nozzle and finally to the transonic axial rotor where it is expelled back to the atmosphere.

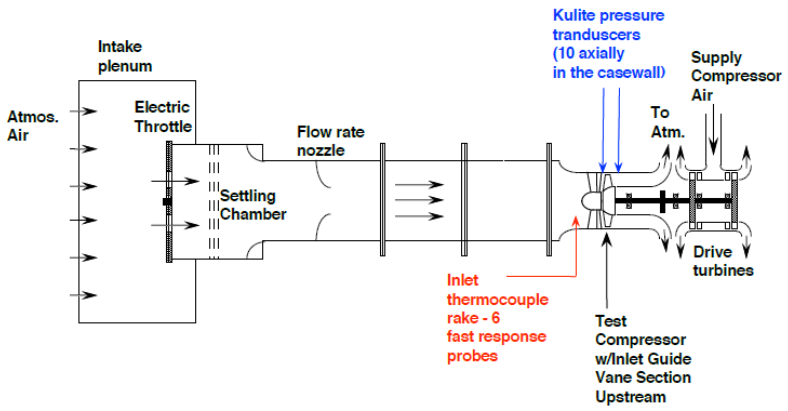

Figure 1. Transonic Compressor Test Rig Design. Source: [4].

Mass flow is measured by a flow rate nozzle placed just aft of the settling chamber depicted in Figure 1. Inlet stagnation pressure and stagnation temperature are measured with two combination-probes. A static pressure port measures inlet static pressure [4]. Outlet stagnation pressure is measured with eleven Kiel pressure probes and nine combination-probes. Outlet stagnation temperature is measured by the nine combination probes. Static pressure is measured with four static pressure ports in the hub, and two static pressure ports in the casing. Inlet measurements are taken at AS1 and outlet measurements are taken at AS3, as shown in Figure 2. Pressure and temperature probe placement is described in Appendix A.

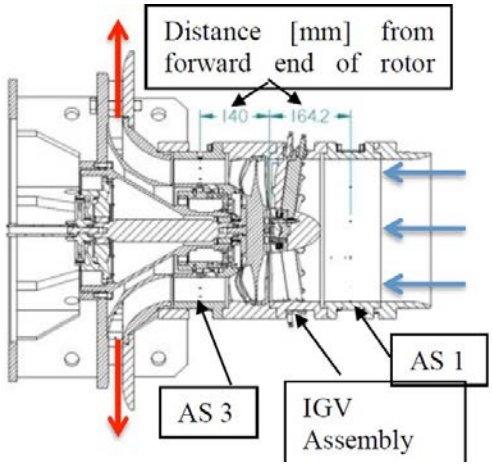

Figure 2. Probe Positions of the Test Section. Source: [4].

Throughflow analysis of the TCR utilizes the stream-tube method illustrated in Figures 3 and 4. Probes measure streamlines of the stream-tube (r_i and r_o) at stations AS1 and AS3. The probes are placed at various radial placements described in Appendix A to create inlet and outlet pressure profiles. The stage or rotor performance is calculated using the mass flow rate and circumferentially mass averaged pressures and temperatures [5].

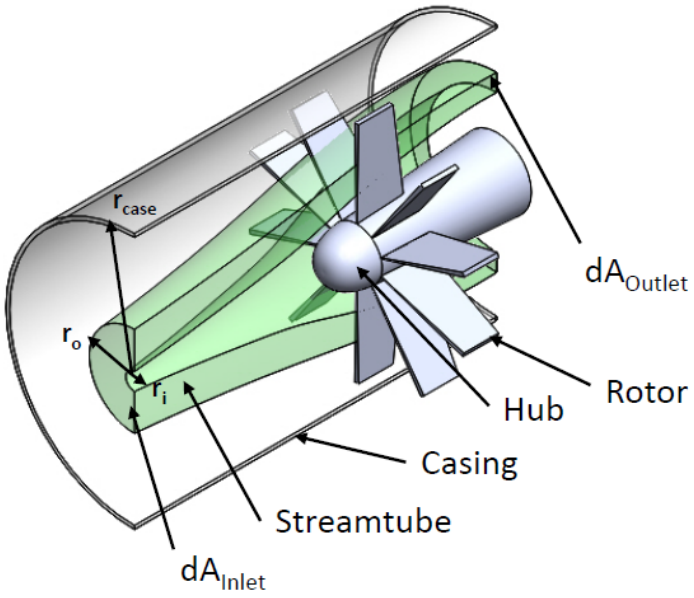

Figure 3. Isometric Air Flow Diagram

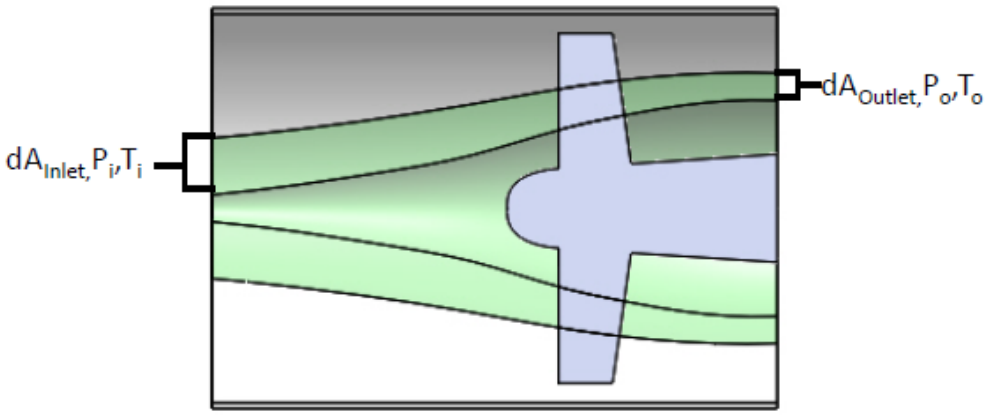

Figure 4. Split View Air Flow Diagram

The throughflow analysis provides the stagnation pressure ratio, isentropic efficiency, and power [6]. The data reduction formulas required to extract useful data are detailed in Chapter V and Appendix B. Figure 5 provides a basic outlook of what the result of data reduction provides for each experiment. The experimental results are compared to the isentropic, or idealized, results as a useful measure of the compressor's performance (Figure 5 (a)). Figure 5 (b) and (c) illustrate the mass averaged pressure ratio and isentropic efficiency versus mass flow at 70%, 80%, and 90% design speed.

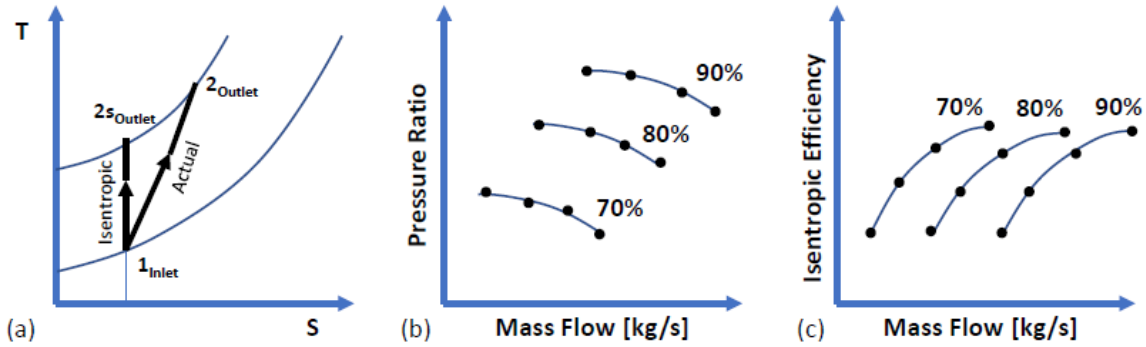

Figure 5. (a) T-S Diagram, (b) Pressure Ratio Map, and (c) Isentropic Efficiency Map

II. LEGACY DATA ACQUISITION SYSTEM

The following section discusses the hardware and software components of the legacy system that will be replaced.

A. HP 75000 SERIES B VXI MAINFRAME

The TCR utilized the HP 75000 Series B VXI mainframe depicted in Figure 6. The VXI (VMEbus eXtensions for Instrumentation) was designed for various applications from test and measurement to data acquisition and analysis. The slot-based chassis was designed as an open, multivendor platform that allowed for modules from different companies to interact within the same software programs. This allowed for more precise timing and synchronization between multiple instruments, and increased data transfer rates.

Figure 6. HP 75000 Series B VXI Mainframe

Thermocouple readings were sent to the HP 75000 via three HP E1347A 16 channel thermocouple multiplexer modules. The VXI mainframe was then connected to the PC with a HP 82341C controller card [7].

A magnetic speed pickup module was located aft of the drive turbines in the TCR and sends a voltage signal for each revolution. The signal was read by a HP E1332A 4 channel counter-totalizer module housed in the HP 75000, and was connected to the PC through the same HP 82341C controller card [7].

B. HPVEE SOFTWARE

Pressure, temperature, and speed data was transferred to the PC and utilized by HPVEE software. HPVEE was a graphical programming language that could acquire data, process data, display results, and generate reports. The graphical user interface (GUI) used a block diagram structure with each block resembling a function, variable, or sub-block diagram. HPVEE also allowed MATLAB code to be used for processing data as shown in Figure 7.

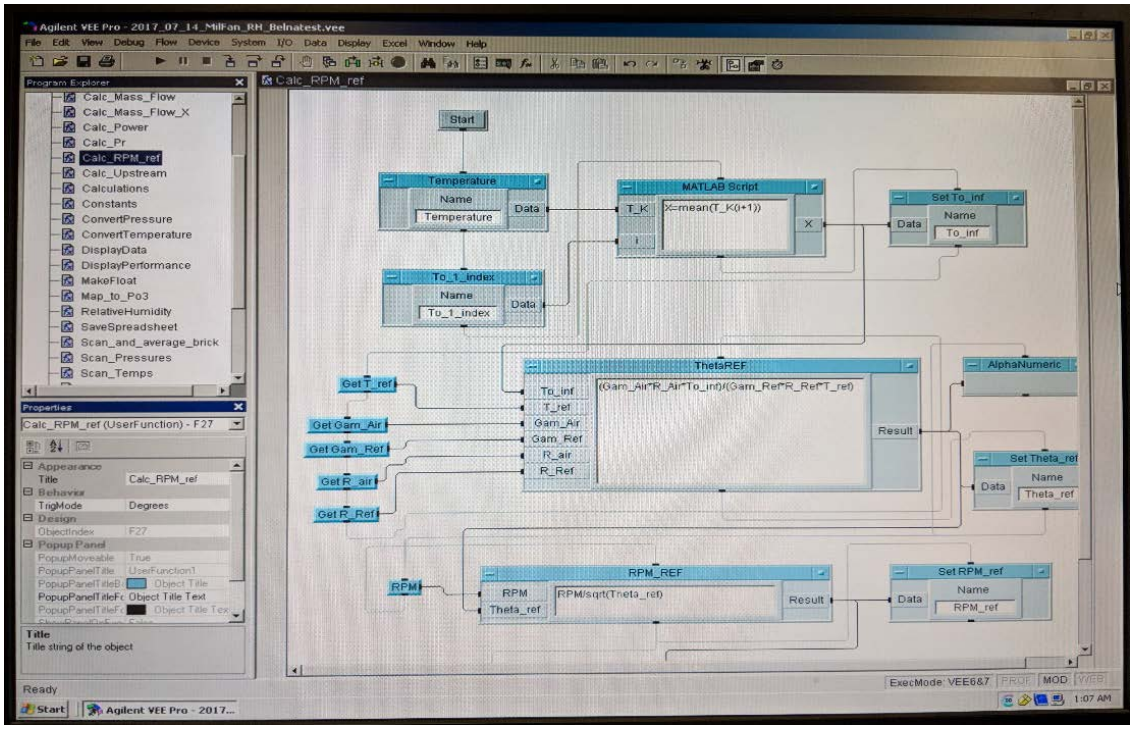

Figure 7. HPVEE Software GUI

HPVEE was limited in how it could process data. It was designed to run in a loop format and took an average of four to six seconds to refresh data. The software could only perform one function at a time, which resulted in a timestamp deviation between the pressure, temperature, and rotational speed data. The program was also very difficult to manipulate and did not have a clear organizational structure. The display formatting options limited the amount of data and graphs that could be displayed. While newer

versions of HPVEE are available, they do not interact with the HP 75000 Series B mainframe.

C. 32-BIT WINDOWS XP COMPUTER

The legacy DAS used a 32-bit Windows XP computer. This computer was significantly slower than the current 64-bit Windows 10 computers and had limited support when repairs were needed. The computer also lacked the ability to communicate with newer hardware components.

D. SYSTEM CONFIGURATION OVERVIEW

An updated schematic of the legacy DAS from McNab [7] is displayed in Figure 8. The thermocouple inputs and magnetic speed pickup were read into the HP 75000 and then relayed to the PC for processing and analysis with HPVEE. The pressure inputs from the pressure bricks were connected to an 8-port hub and then relayed to the same PC. The pressure instrumentation will be explained in further detail in the next chapter. The strain measurements recorded by the strain gauges were routed to an NI cDAQ 9181 and then to the same 8-port hub. The strain measurements were then sent to a separate PC and were not modified as part of this study.

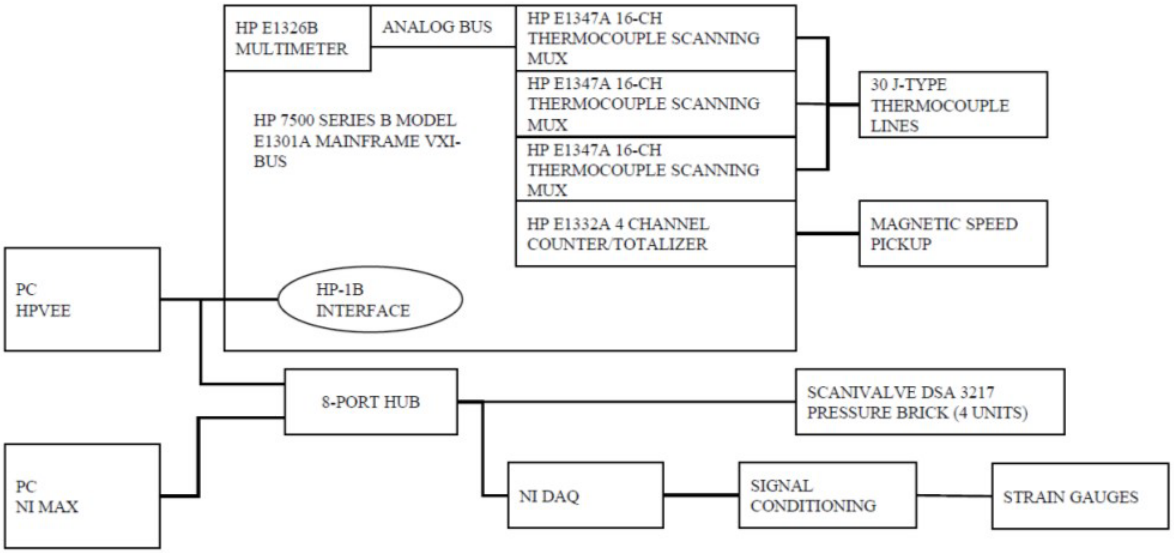

Figure 8. Schematic of Legacy DAS. Adapted from [7].

E. LEGACY DAS PERFORMANCE MAPS

Rotor only tests were run at speeds of 70%, 80%, 85%, and 90% of the design rotational speed. During each of these test runs, the air flow into the compressor was throttled by an electronically actuated butterfly valve. Measurements were taken at each throttle setting until the system reached stall conditions.

Figure 9 displays the results of these test runs as the total-to-total pressure ratio versus the corrected mass flow rate. Figure 10 displays the results as isentropic efficiency versus the corrected mass flow rate. Each speed line in the maps corresponded to a test run at the specific percentage of the design rotational speed. These maps are used to compare the performance and efficiency of different rotor and casing designs.

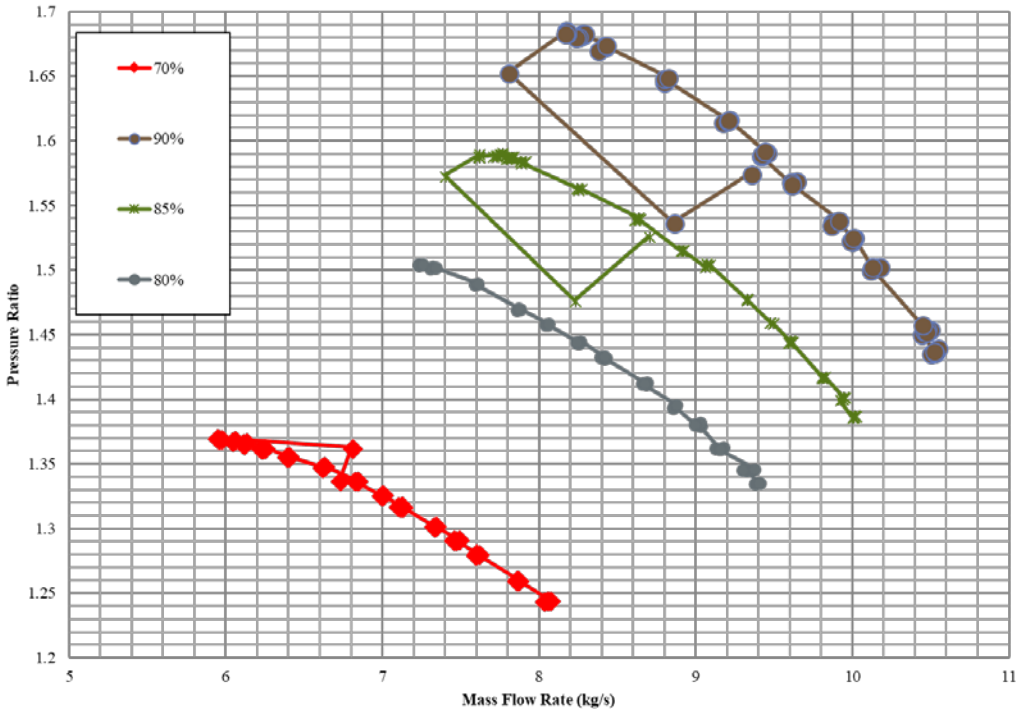

Figure 9. Pressure Ratio Map

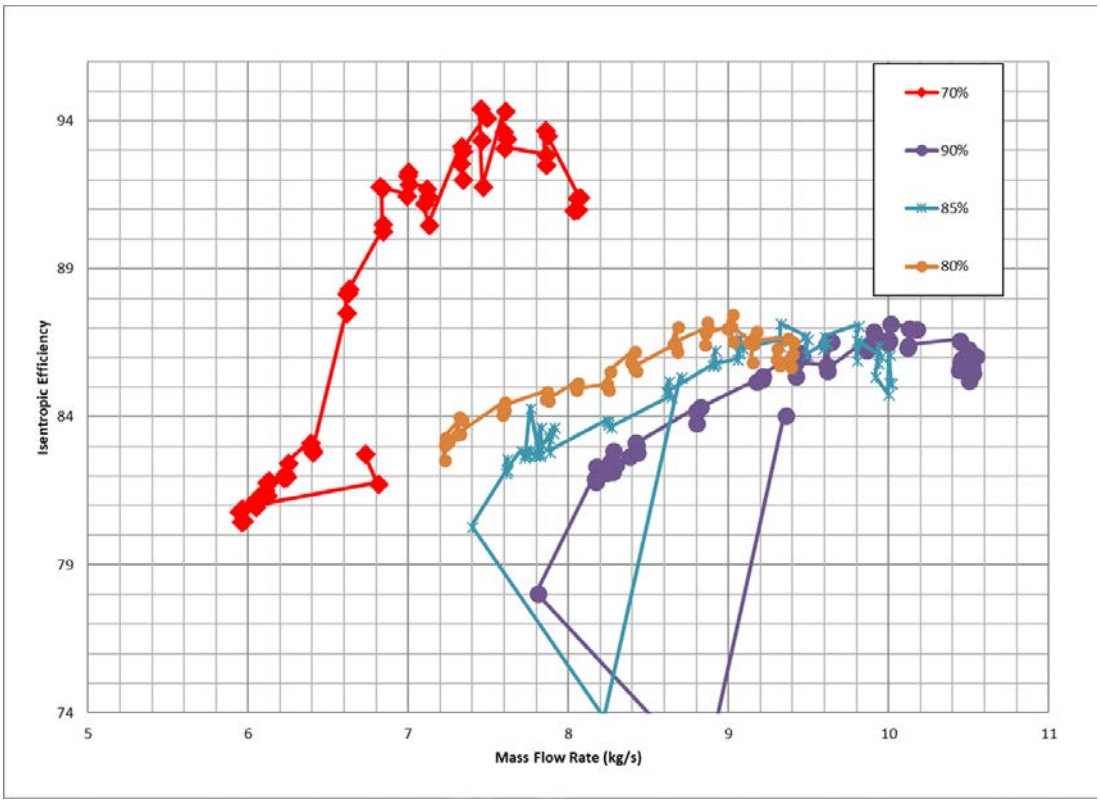

Figure 10. Isentropic Efficiency Map

THIS PAGE INTENTIONALLY LEFT BLANK

III. NEW DATA ACQUISITION SYSTEM

This section will describe the hardware components of the new data acquisition system.

A. REQUIREMENTS OF UPDATED DATA ACQUISITION SYSTEM

The new DAS hardware met the following requirements to be considered for the new system:

1. **Compact:** The hardware needs to be small enough to fit in the test cell.
2. **Ruggedized:** The hardware must be able to withstand high temperatures and vibrations.
3. **Ethernet Capable:** The hardware needs to connect to the PC via an ethernet connection.
4. **Measure J-Type Thermocouple:** The thermocouples in the TCR will remain the same and the new system must connect to the existing thermocouples.
5. **Counter-totalizer:** The hardware should have a counter-totalizer that can measure pulse width and frequency.
6. **Synchronized Timing:** Must provide synchronized time to accurately correlate and log the temperature and speed data.
7. **Modern Computing Equipment and Software:** A modern high-level programming language that operates on a 64-bit computer.

B. PROCUREMENT OF UPDATED DATA ACQUISITION SYSTEM

The existing digital sensor array or “pressure bricks” were kept in the new DAS because they are rugged, reliable, and could interact directly using the prevalent ethernet connection standard.

Several compact data acquisition systems were considered from Measurement Computing Corp., National Instruments, DATAQ Instruments, and Pacific Instruments to replace the temperature and speed instrumentation. The National Instruments CompactDAQ system was chosen for its ability to meet all seven hardware requirements. The CompactDAQ products are rugged, portable, and are module based. National Instruments also provides a wide array of support, tutorials, and sample code for utilizing their instruments with MATLAB.

1. Digital Sensor Array

Pressure measurements from the probes installed in the TCR were monitored in near real time by three Scanivalve Corporation DSA 3217/16Px (25psid) pressure bricks pictured in Figure 11, and one Scanivalve Corporation DSA 3217/8Px (25psid) pressure brick. The DSA 3217 pressure bricks can scan at a rate of 500 samples/chan/sec with UDP Binary acquisition or 200 samples/chan/sec with TCP/IP ASCII acquisition with an accuracy of ±0.05% FS [8].

Figure 11. DSA 3217/16Px “Pressure Brick”

2. Chassis

The NI cDAQ-9188XT chassis displayed in Figure 12 has the following design specifications:

- 8 modular input slots
- Data-synchronization with a built-in clock
- Supports several measurements such as temperature, strain, voltage, and resistance
- Operating temperature ranging from -40 ºC to 70 ºC
- High ruggedness, vibration tolerance ranging from 10 Hz to 500 Hz

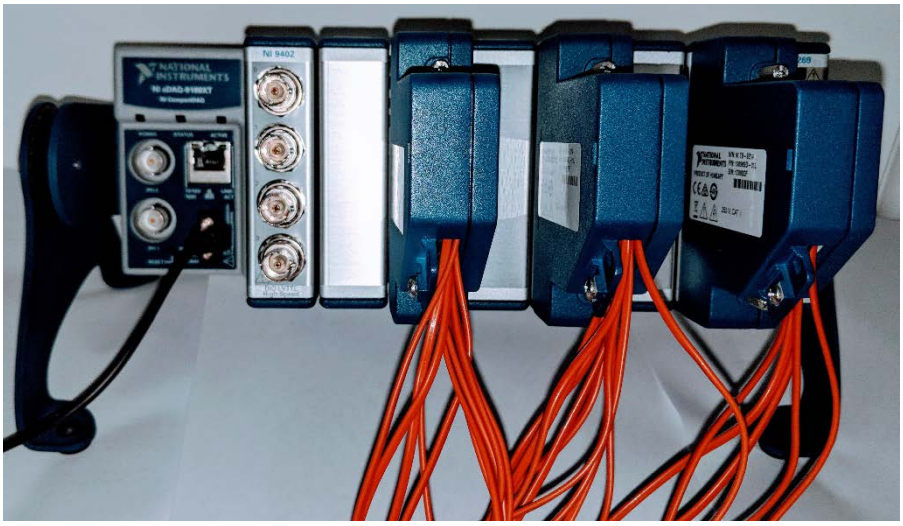

Figure 12. NI cDAQ-9188XT Chassis

3. Temperature Module

Three NI 9214 C series thermocouple modules were chosen for thermocouple readings, as pictured in Figure 13. The module has the following specifications:

- 16 thermocouple channels

- Maximum accuracy of 1.24 ºC
- Autozero channel for compensation of offset errors
- Sample rate of 68 Scans/s
- High-resolution and high-speed timing modes

Figure 13. NI 9214 High-Density Thermocouple Module. Source: [9].

4. Rotational Speed Module

The NI 9402 C series digital module in Figure 14 was chosen to measure rotational speed. The module has the following specifications:

- 4 bidirectional individually configurable channels
- BNC connectivity
- Update rate of 55 ns
- Frequency and pulse width measurement capable

Figure 14. NI 9402 Digital Input/Output Module. Source: [10].

5. MATLAB Software

MATLAB software has a wide array of capabilities available including a full suite of data acquisition support. It can talk directly with the new DAS hardware and can run on the most up to date OS platforms. MATLAB can also acquire continuous data in the background from the chassis, which allows for near real time synchronization with the pressure bricks.

6. 64-bit WINDOWS 10 Computer

64-bit WINDOWS 10 computers use the most up-to-date OS available and can communicate with all required hardware and software.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. HARDWARE IMPLEMENTATION

A. DSA 3217 PRESSURE BRICK

The DSA 3217 pressure bricks have a static IP address and require the PC to be configured with a static IP containing the same first three octets as the pressure bricks. More modern measurement instruments have dynamic IP's that simplify the connection process. The pressure brick interfaces with MATLAB using the Instrument Control App. From the app, commands can be sent and received with the pressure brick and are provided in the DSA3217 operation and service manual [8]. After connecting to the pressure brick and sending and receiving data through the Instrument Control App, the session was exported to generate the code used by the app. This code was then modified to run the pressure bricks from a main script. Full details on how to setup the pressure bricks and what commands to use are discussed in Appendix C.

The ASCII packet format is used to receive pressures from the pressure brick. The pressure brick can be customized to take a specified number of readings or frames per scan (FPS). The range is from 1 FPS to continuous FPS. The pressure brick formats these pressures into packets and places them in an output buffer. The packet is in the following format.

Frame # <number> (Subpacket 1)

<port #> <pressure [PA]> <temperature [K]> (Subpacket 2)

" " " "

" " " "

<port #> <pressure [PA]> <temperature [K]> (Subpacket n)

MATLAB can only process one subpacket per read command from the output buffer. To ensure near real time measurements, the FPS is set to 1 to ensure that the most recent pressure scan can be paired with the most recent temperature and rotational speed data. Figure 15 is a schematic of the pressure data collection protocol.

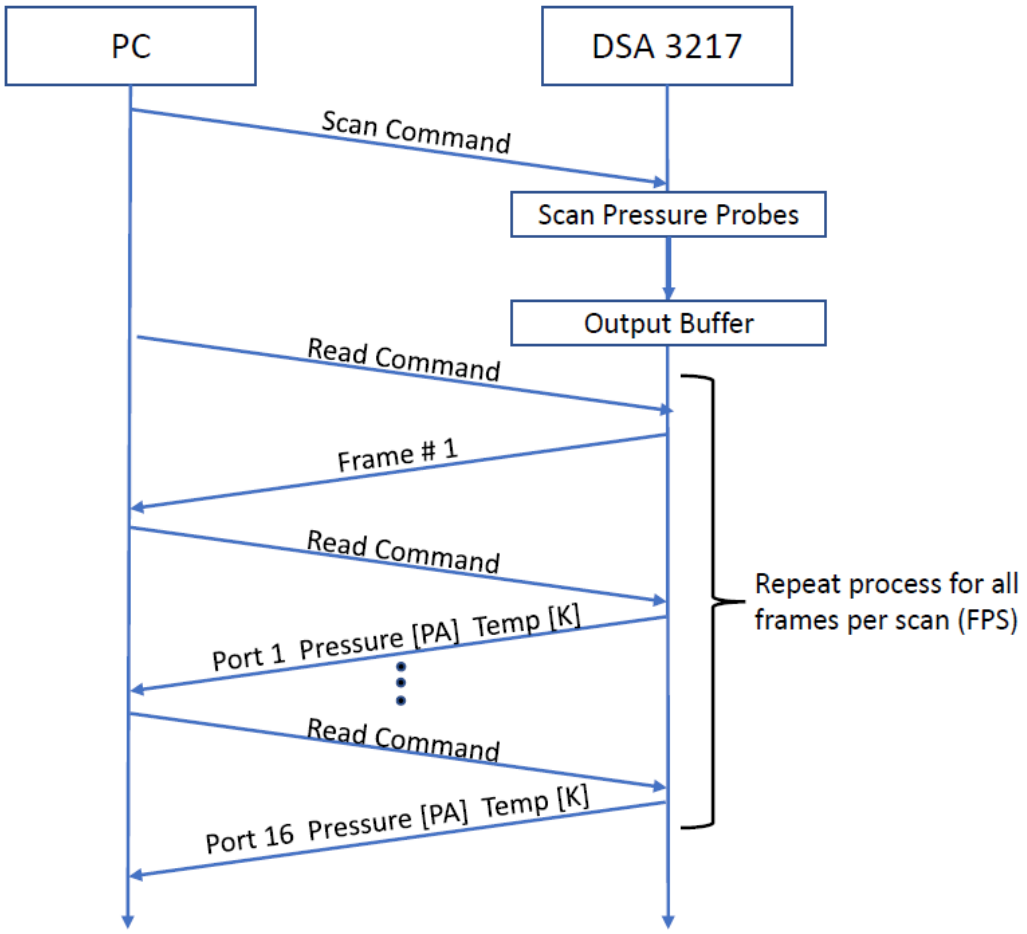

Figure 15. Pressure Data Collection Protocol

B. NI CDAQ 9188XT CHASSIS AND MODULES

The chassis require the installation of the National Instruments Measurement and Automation Explorer (NI MAX). Prior to taking measurements the chassis' IP configuration is set to static IP due to the pressure bricks requiring a static IP connection. Prior to running the chassis with MATLAB, the chassis and its associated modules must perform a self-test in NI MAX. Detailed procedures for connecting the chassis and associated modules is found in Appendix C.

The NI 9214 thermocouple and NI 9402 counter-totalizer modules are setup using example MATLAB scripts provided by MathWorks [11]. They are configured for high-

resolution mode with a scan rate of two scans per second. Detailed setup instructions are provided in Appendix C.

C. SYSTEM CONFIGURATION OVERVIEW

Figure 16 depicts the upgraded DAS diagram. The pressure data acquired by the DSA 3217 pressure bricks directly connect to the 8-port hub. The temperature and speed data from the NI cDAQ 9188XT and the strain gauge data from the NI cDAQ 9181 also connect directly to the 8-port hub. The pressure, temperature, and speed data are passed to the first PC, and the strain gauge data is passed to the second PC.

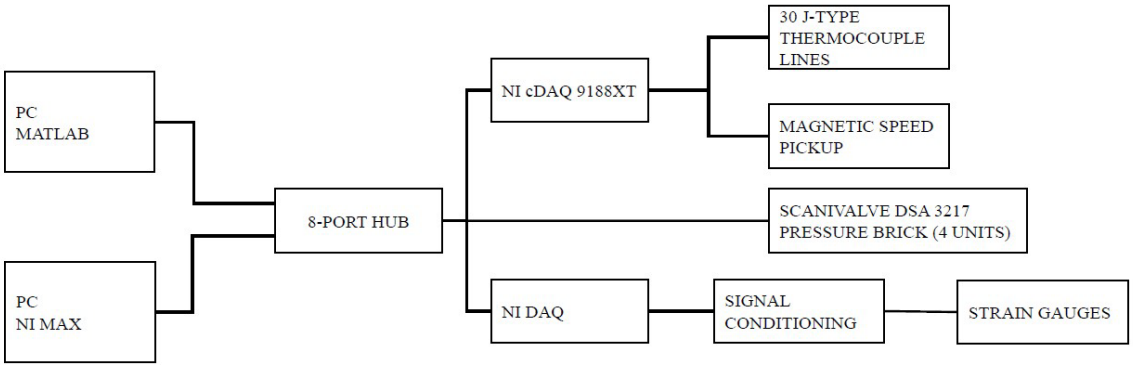

Figure 16. Schematic of Upgraded DAS

THIS PAGE INTENTIONALLY LEFT BLANK

V. DATA REDUCTION

As explained by Sayari and Bölcs [5], “on each stream-tube, the flow is assumed one-dimensional and different circumferential averaging techniques are employed in order to ensure the conservation laws.” There are three types of averages that can be used in data reduction of the TCR.

The first definition is the area average,

$$\left(\overline{P^a}\right) = \frac{\int P dA}{\int dA}$$

The second definition is the density average,

$$\left(\overline{P^d}\right) = \frac{\int P \rho dA}{\int \rho dA}$$

The third definition is the mass average,

$$\left(\overline{P^m}\right) = \frac{\int P \rho v_z dA}{\int \rho v_z dA}$$

Hirsch and Dring [12] debated at length the differences between using density averaged and mass averaged quantities for throughflow analysis. The density averaged equation reduces the area averages in incompressible flow. Additional “interaction” terms are also produced that require iterative computations of the stream surfaces. Density averaged quantities had been the standard method for processing throughflow models. But Hirsch and others in his field explored using mass averaged quantities to validate the TCR throughflow models.

Hirsch and Dring [12] postulated that the mass averaged quantities had, “physical arguments and the strong connection between mass averaged quantities, like stagnation pressure, total energy and machine efficiency, are essential to the correct estimation of the energy within the turbomachinery blade row.” With the mass averaged quantity, the stream-tube is assumed constant at the inlet and outlet with the streamlines nearly

axisymmetric. This approach gives a more precise solution to the energy equation and a simplified data reduction that is further explained in Appendix B.

Utilizing the mass average definition produces the mass averaged pressure ratio,

$$\overline{\left(\frac{P_{o3}}{P_{01}}\right)} = \frac{\int\left(\frac{P_{o3}}{P_{o1}}\right)\rho_3 v_{z3} dA_3}{\int \rho_3 v_{z3} dA_3}$$

And the mass averaged efficiency ratio,

$$\bar{\eta} = \frac{\int \eta \rho_3 v_{z3} dA_3}{\int \rho_3 v_{z3} dA_3}$$

VI. HARDWARE COMMUNICATION

The first step in creating the new DAS program in MATLAB was to write individual test programs for each component.

A. PRESSURE MEASUREMENT PROGRAM FLOW

The flow diagram of the pressure measurement program is illustrated in Figure 17. The output of this program is the pressure graph in Figure 18, and the raw pressure data in the command line shown in Figure 19.

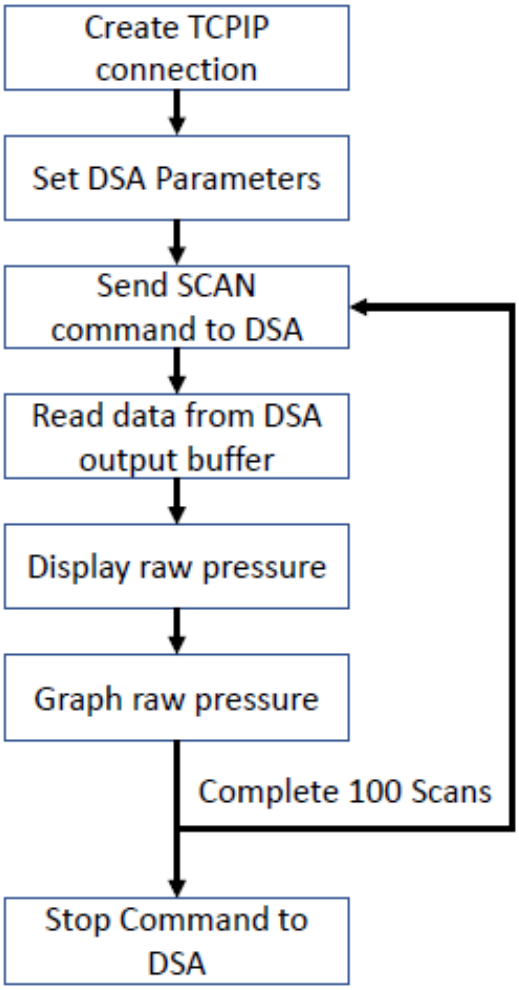

Figure 17. Pressure Measurement Flow Diagram

This program measures gauge pressure and does not add atmospheric pressure to the value. This results in small values shown in Figure 18. This figure constantly updates as the conditions change at the pressure brick.

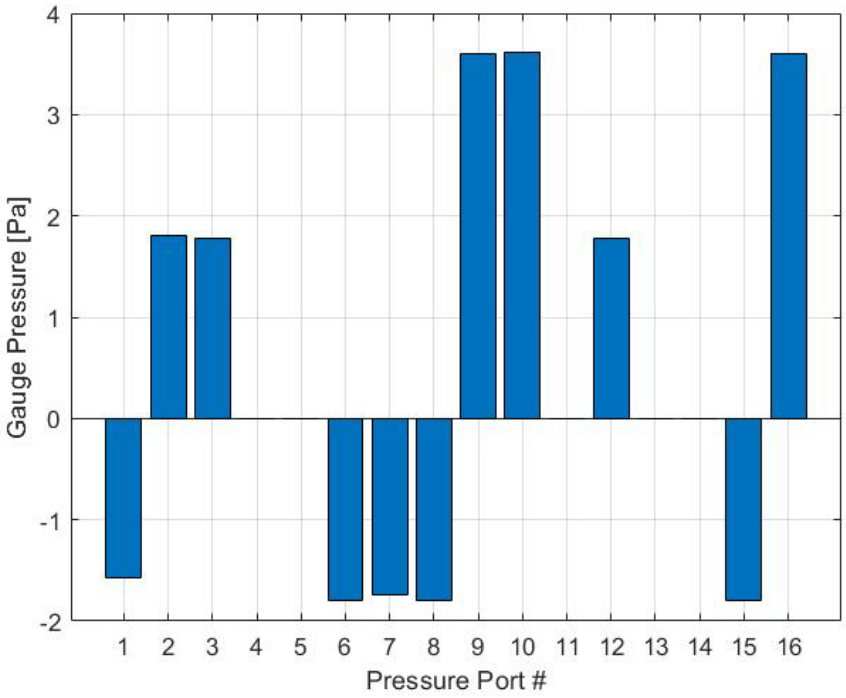

Figure 18. Pressure Measurement Bar Graph

Figure 19. Pressure Measurement Raw Data

B. TEMPERATURE MEASUREMENT PROGRAM FLOW

The flow diagram of the temperature measurement program is illustrated in Figure 20. The output of this program is the temperature graph in Figure 21, and the raw temperature data in the command line shown in Figure 22.

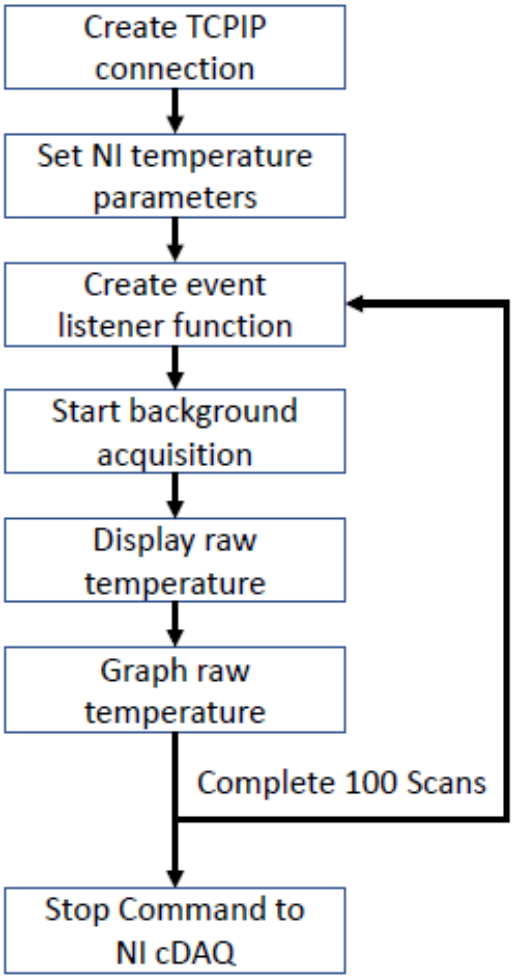

Figure 20. Temperature Measurement Flow Diagram

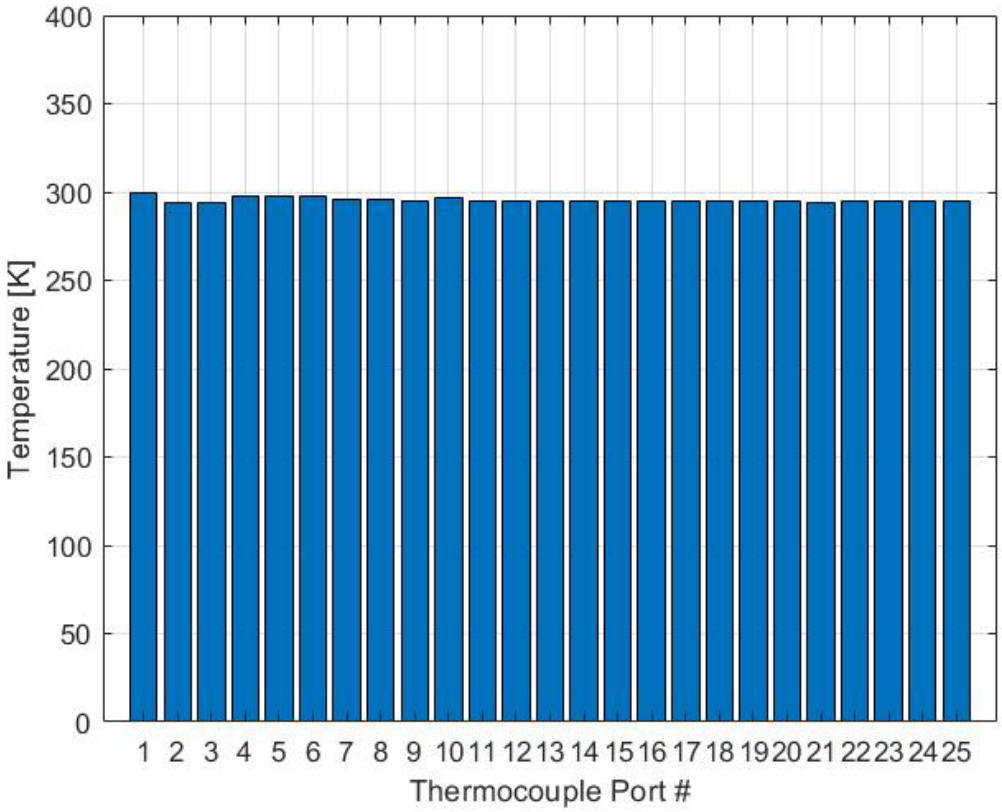

Figure 21. Temperature Measurement Bar Graph

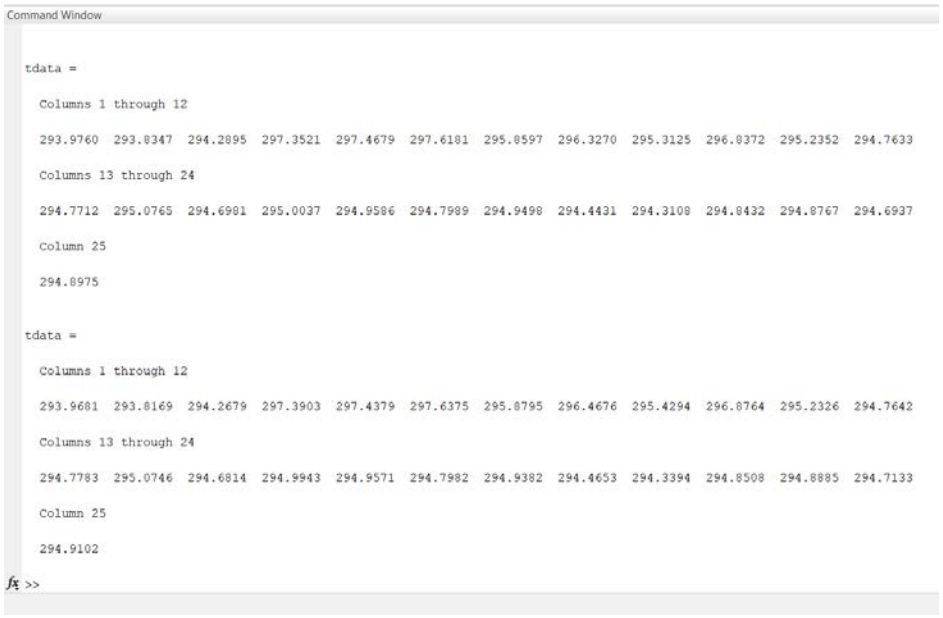

Figure 22. Temperature Measurement Raw Data

C. ROTATIONAL SPEED MEASUREMENT PROGRAM FLOW

The flow diagram of the rotational speed measurement program is illustrated in Figure 23. The compressor was not run during this program test and a signal generator was used to compensate. The output of this program is the RPM data in the command line shown in Figure 24.

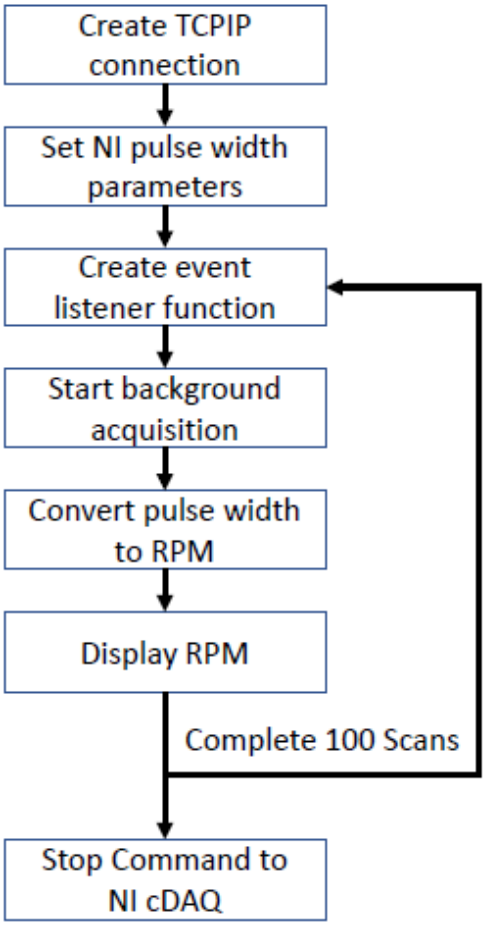

Figure 23. Rotational Speed Measurement Flow Diagram

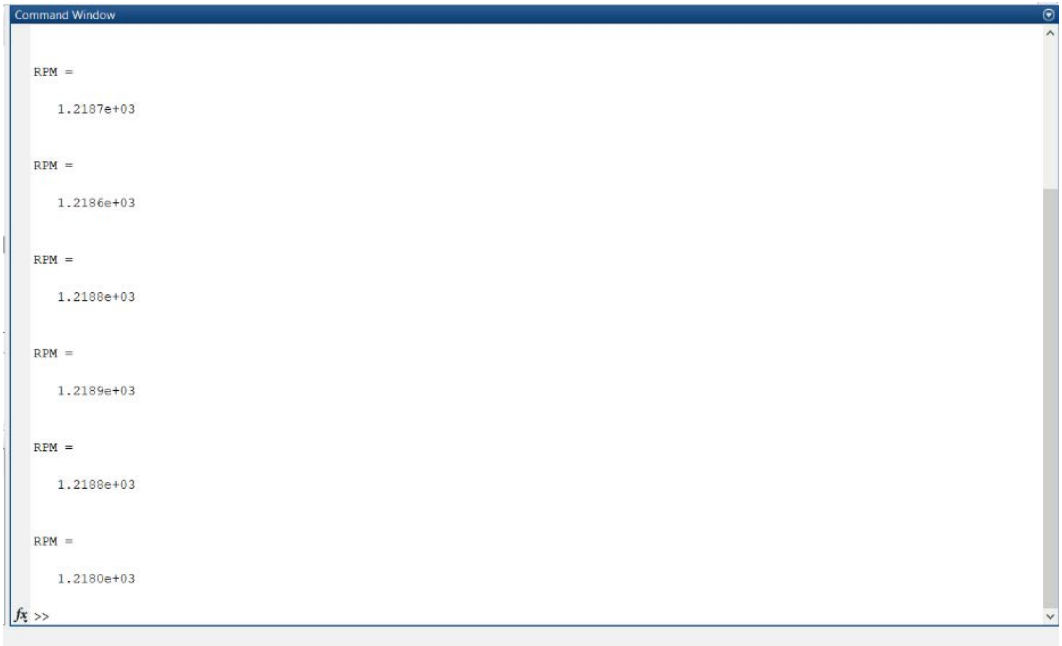

Figure 24. RPM Data

The MATLAB program scripts for the three test measurements is provided in Appendix D.

D. COMBINED PROGRAM FLOW

The DAS program consists of ten MATLAB scripts and functions. These include the following, the code of which can be found in Appendix E.

Table 1. MATLAB Scripts

Script Name	Description
Main.m	Initializes test instrumentation, calls for initial constants, and loads figures.
Pparameters_multibrick.m	Connects to pressure bricks and sets parameters such as units and FPS.
Calc_constants.m	Calculates initial constants.
Constants.m	Sets user inputs constants and calls for initial constants to be recalculated.
Pulsewidth_main.m	Reads pulse width from NI cDAQ and converts to RPM.
Temp_readings_main.m	Reads temperature from NI cDAQ.
Run_Startmulti.m	Starts data acquisition process, continuously performs data reduction, and outputs results to figures.
Run_Stopmulti.m	Stops data acquisition process.
Reset_Num.m	Resets run number to one.
Save_Run.m	Saves output variables in .txt file and adds run number.

The DAS program is initiated with the script Main.m. This connects the pressure bricks and the NI cDAQ 9188XT chassis to MATLAB, loads the constants from the previous test run, and loads the required figures. The flow diagram for DAS initialization is illustrated in Figure 25.

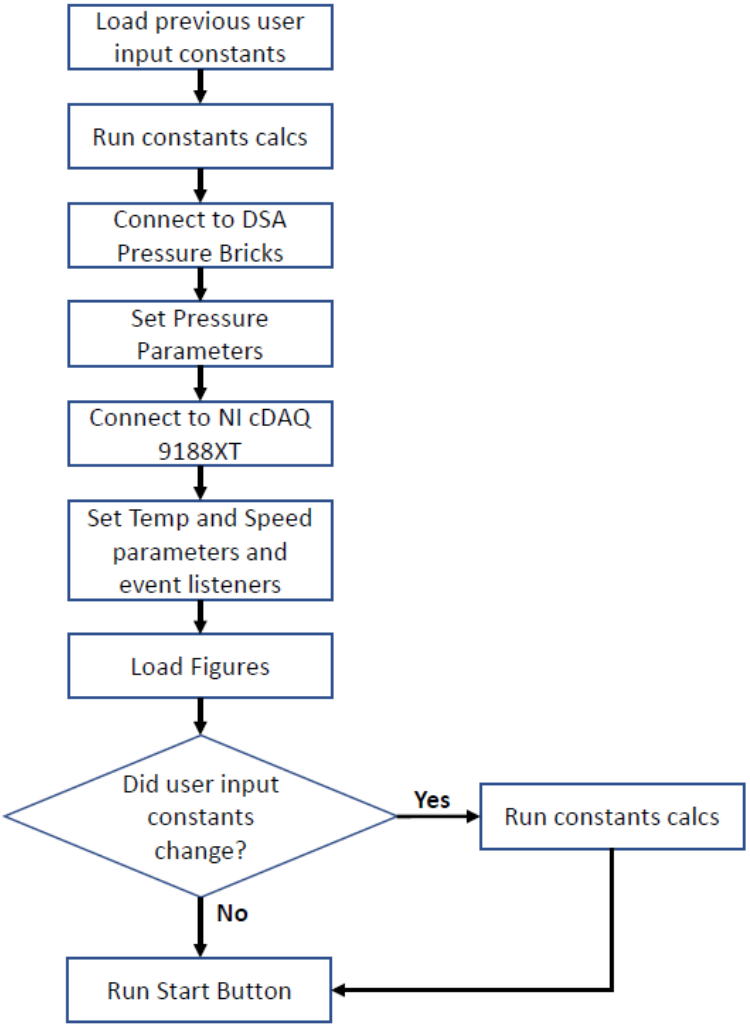

Figure 25. DAS Initialization Flow Diagram

Figure 26 depicts the main HMI figure which includes the command pushbuttons, data reduction calculations, and the downstream pressure and temperature graphs.

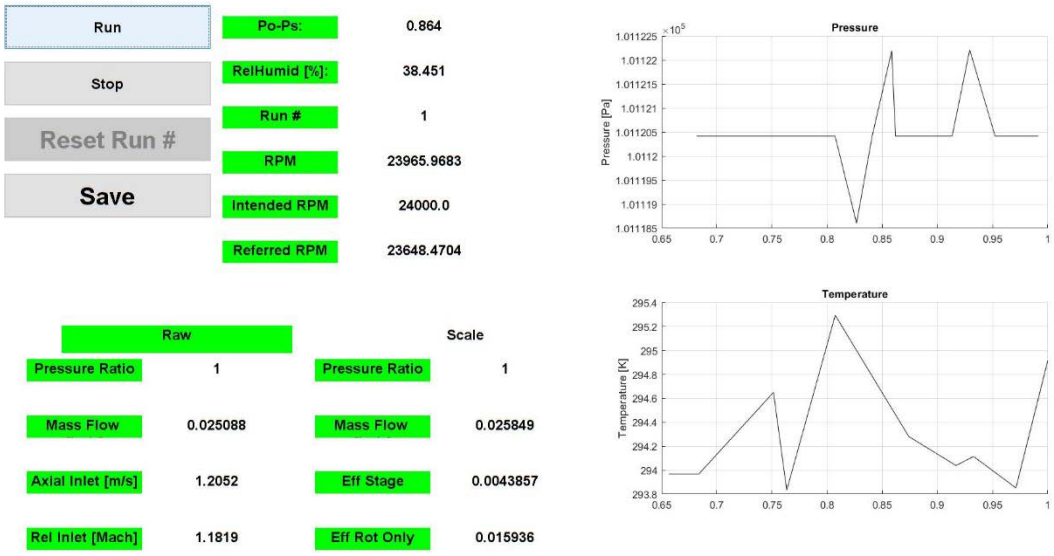

Figure 26. HMI Figure

Figure 27 is the user input constants figure. When generated, the figure contains constants from the previous test run. If any of the constants change, the user can enter new constants and the program will rerun the Calc_constants.m file with the new data.

Figure 27. User Input Constants Figure

When the data acquisition process is running, Figure 28 displays the raw pressure and temperature data. The pressure data displayed is the gauge pressure read from the pressure brick added to atmospheric pressure.

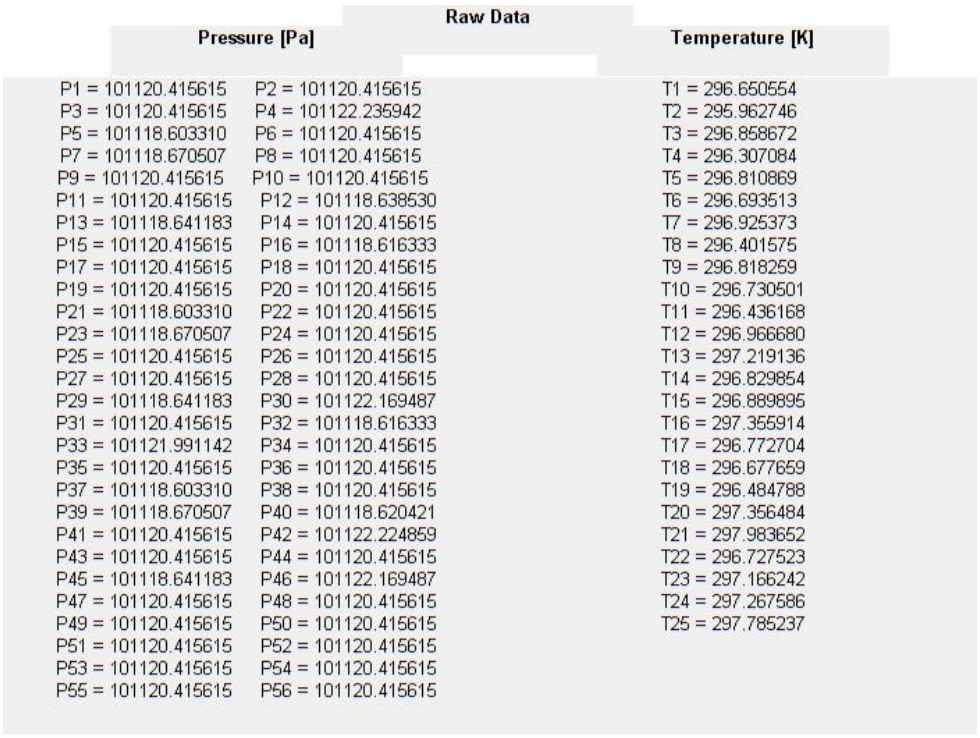

Figure 28. Raw Pressure and Temperature Data Figure

After the DAS program completes initialization, the user can either input new constants, or they can start the data collection process by pressing the run push button in the HMI figure. The data collection flow diagram when the user starts the run is depicted in Figure 29.

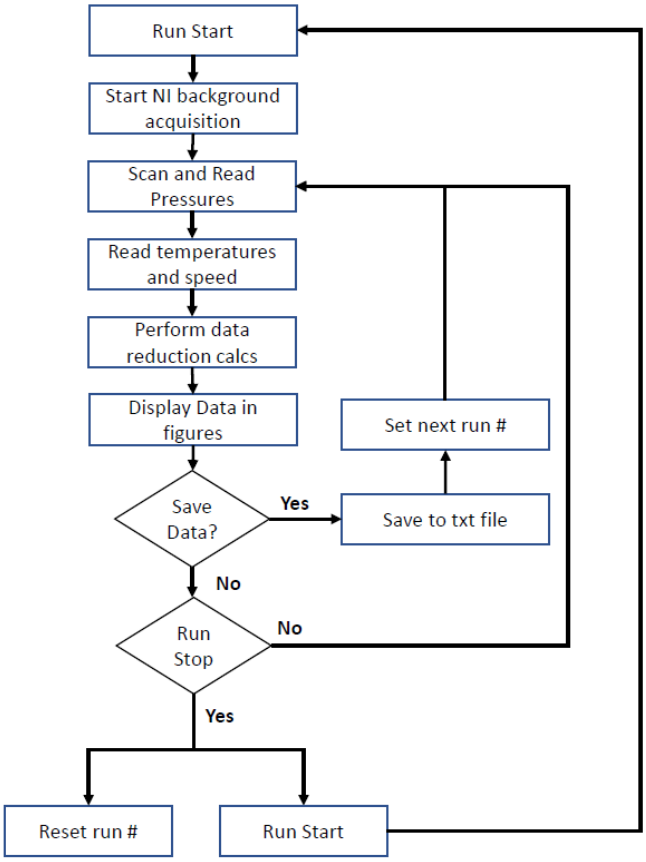

Figure 29. Data Collection Process Flow Diagram

The programmatic implementation of the MATLAB scripts and figures is designed to emulate the HPVEE program. This was chosen to ensure minimum disruption to the testing program. The main changes are the use of multiple figures and the ability to disable certain features depending on what process is being performed. For example, when the data collection process is not active, the save pushbutton is disabled so that the user will not collect false data stored in MATLAB's memory. When the data collection process is active, the reset run number and apply changes to constants pushbuttons are disabled.

While each pushbutton in the figures has its own function associated to it, the primary process including the calculations and data display are carried out directly in the Run_Startmulti.m function. A detailed table of the program steps with all functions described is provided in Appendix E.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. RESULTS

To validate the new DAS, three criteria were chosen.

- All pushbuttons must operate as intended.
- All figures must display and update the required data.
- The results of the data reduction calculations must match the old DAS.

An actual test run of the TCR was not chosen for initial system validation. To reduce risk involved in changing to the new DAS, simulated tests were conducted between the legacy DAS and new DAS. The criteria were tested using a spare pressure brick with 16 pressure ports and the cDAQ 9188XT chassis connected to 25 thermocouples and a function generator. The pressure brick ports were called multiple times in a loop to simulate 56 pressure ports. To compare the two systems, 50,000 Pascals was added to the P_{o3} pressure ports and 150 K was added to the T_{o3} thermocouples. This produced a pressure ratio of approximately 1.5.

After the Main.m script successfully loaded all figures and calculated the constants, the user input constants were changed to match each other as shown in Figure 30. Next the "Run" pushbutton was pressed. This successfully disabled the "Reset Run #" and "Apply" pushbuttons and enabled the "Save" pushbutton as shown in Figures 30 and 31. This also started the data acquisition process. Approximately every 2 seconds, the HMI and raw data figures updated with new values as shown in Figures 31 and 32. The "Save", "Stop", and "Reset Run #" pushbuttons operated as intended as well. This successfully completed the first two criteria.

Figure 30. User Input Constants Figure Disabled

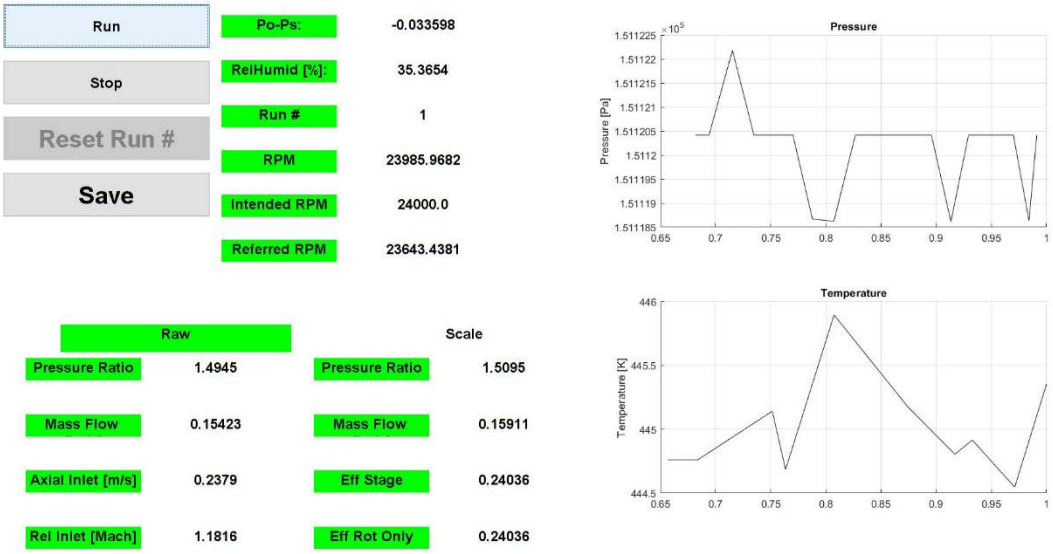

Figure 31. HMI Figure with Adjusted P_{o3} and T_{o3} Data

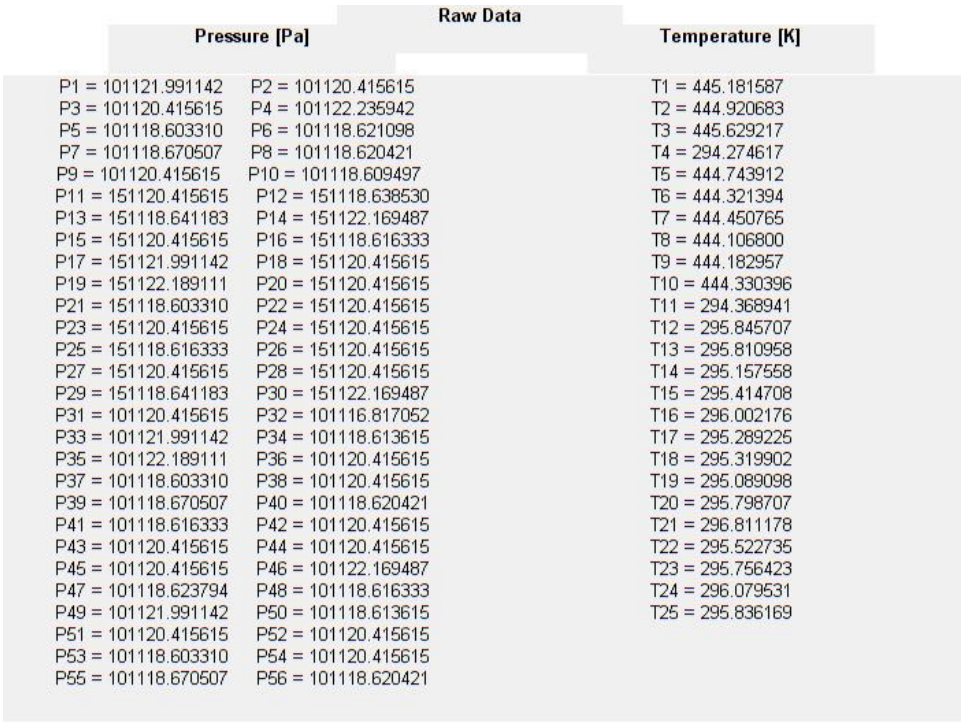

Figure 32. Raw Pressure and Temperature Data Figure with Adjusted P_{o3} and T_{o3} Data

To meet the third criterion, the HPVEE system was run with the same input constants and parameters as the new DAS. Both DAS programs are designed to output a text file with several variables including raw data and the results of data reduction calculations. These files were reviewed in Excel for any major deviations. The review found that the values of the new DAS were within an acceptable range of the old DAS values. The only values that deviated significantly were some of the raw pressure and temperature values. This was due the fact that not all the pressure and temperature ports of the TCR are actively used. The output data files for both systems are provided in Appendix F.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The objective of this work was to procure, design, and implement upgrades to the TCR DAS, all of which has been accomplished. The following summarizes the main accomplishments of this work.

1. New commercial-off-the-shelf hardware has been procured that integrates well with the existing system.

2. The new DAS utilizes MATLAB to perform all necessary data acquisition and data processing. The code can easily be modified to add new data reduction calculations pertinent to testing.

3. The upgraded DAS can perform all the same functions as the previous system and can output results in the same format.

B. RECOMMENDATIONS

This work requires further implementation and testing before completely transferring to the new system. Recommendations are as follows:

1. To complete the system installation, a switchboard for the thermocouple ports must be installed. This would allow for the thermocouples to be connected to the new DAS and the old DAS as needed to continue comparisons of the actual TCR.

2. The figures should be modified to meet the needs of the user. The figure design emulated the HPVEE design for comparison purposes. The new figures could be customized to be easier to view as the user desires.

3. Full TCR test runs should be completed with the new DAS to confirm the validity of the program.

4. The MATLAB program scripts and functions should be modified to make the program more organized. Multiple sections of the code could be broken down into subsections to reduce lag and make future changes easier.

APPENDIX A. TEST INSTRUMENT PLACEMENT

Pressure and temperature measurements are taken forward and aft of the rotor at varying radial locations. Figures 33 and 34 display the instrumentation ports at the AS3 location.

Figure 33. Pressure and Temperature Instrumentation Ports for Outlet Casing

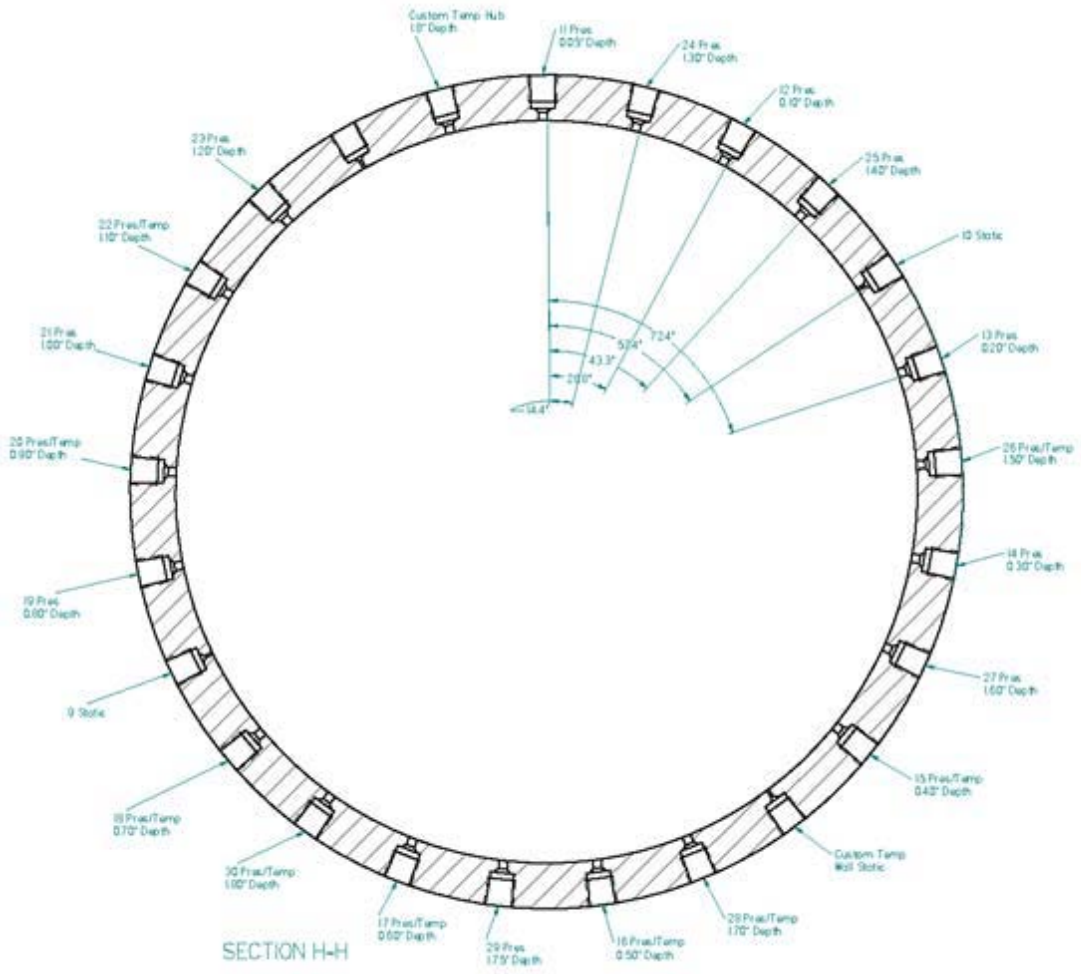

Figure 34. Diagram of Pressure and Temperature Instrumentation Ports for Outlet Casing. Source: [7].

Table 2 lists all the instrumentation ports and their respective locations. For example, the current system configuration has ports 5 and 6 located upstream at AS1 and ports 11 through 30 located downstream at AS3.

Table 2. Pressure and Temperature Port Position. Source: [13].

Port #	Scanivalve 1	Desired Depth (P)	Actual Depth	Desired Depth (T)	Actual Depth	Percentage	Angle position	Orientation, Rotor only	Temp Channels
1	Tare								
2	Scale								
3	Flow Nozzle P6								
4	Flow Nozzle Ps								
5	Inlet Pt1 Kiel Probe/TC Combo								15
6	Inlet Pt1 Kiel Probe/TC Combo								13
7	Inlet Ps1 (static)								
8	Inlet Ps1 (static)								
9	Outlet Ps3 (static)								
10	Outlet Ps3 (static)								
11	Kiel Probe	0.05	0.06				0°	39.5	
12	Kiel Probe	0.1	0.12				28.8°	39.6	
13	Kiel Probe	0.2	0.2				72°	39.7	
14	Kiel Probe	0.3	0.305				115.2°	39.8	
15	Kiel Probe	0.4	0.42				144°	39.8	
16	Kiel Probe	0.5	0.52				187.2°	39.8	
17	Kiel Probe/TC combo	0.6	0.59	0.225 [85.1%]	0.202	89.7%	216°	39.8	8
18	Kiel Probe/TC combo	0.7	0.775	0.323 [82.4%]	0.4	79.6%	244.8°	39.9	7
19	Kiel Probe	0.8	0.815				273.6°	39.9	
20	Kiel Probe/TC combo	0.9	0.91	0.525 [73.0%]	0.51	74.0%	302.4°	40	6
21	Kiel Probe	1	1				316.8°	40.4	
22	Kiel Probe/TC combo	1.1	1.13	0.722 [60.8%]	0.73	62.8%	331.2°	40.7	5
23	Kiel Probe	1.2	1.2				345.6°	41.1	
24	Kiel Probe	1.3	1.31		0.93	52.6%	14.4°	41.5	
25	Kiel Probe	1.4	1.41				43.2°	42.2	
26	Kiel Probe/TC combo	1.5	1.48	1.121 [37.8%]	1.05	46.4%	86.4°	42.8	3
27	Kiel Probe	1.6	1.64				129.6°	43.5	
28	Kiel Probe/TC combo	1.7	1.765	1.325 [28.3%]	1.37	30.1%	172.8°	44.2	9
29	Kiel Probe	1.75	1.78				201.6°	44.9	
30	Kiel Probe/TC combo	1.8	1.795	1.482[19.89%]	1.42	27.6%	230.4°	45.3	2
31	Hub static pressure P2 1/ Static Kulite [8]								
32	Hub static pressure P2 2								
33	Hub static pressure P2 3								
34	Hub static pressure P2 4								
35	Hub static pressure P3 1								
36	Hub static pressure P3 2								
37	Hub static pressure P3 3								
38	Hub static pressure P3 4								
39	Static Kulite [-2]								
40	Static Kulite [-1]								
41	Static Kulite [0] Leading edge								
42	Static Kulite [1]								
43	Static Kulite [2]								
44	Static Kulite [3]								
45	Static Kulite [4]								
46	Static Kulite [5]								
47	Static Kulite [6]								
48	Static Kulite [7]								
49	Custom temperature probe			1.805 [5%]					10
50	Custom temperature probe			0 [100%]					1

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. DATA REDUCTION

The data reduction calculations performed in this program were described in detail by Drayton [2], Hobson [4], Descovich [6], and McNab [7]. The only data reduction calculations not previously described in a thesis were the relative humidity calculations, which were provided by Gannon [13] and detailed by Borgnakke [14].

The first calculation determines the saturated air humidity ratio ω_{sat}. Gas constants R_{air} and R_{water}, and total pressure P_{tot} are provided in the initial constants calculations. Saturated vapor pressure P_{vapor_sat} is found using the wet bulb temperature and the XSTEAM function.

$$\omega_{sat} = \frac{m_{vapor\_sat}}{m_{air}} = \frac{R_{air}}{R_{water}}\left(\frac{p_{vapor\_sat}}{p_{tot} - p_{vapor\_sat}}\right) \tag{1}$$

From the adiabatic saturation process, the first law can be reduced to equation (2). Here C_{p_air} is the specific heat capacity of moist air. All variables are previously calculated except for the relative humidity ratio ω, saturated vapor enthalpy, $h_{vapor\_sat\_wet}$, saturated liquid enthalpy h_{liq_sat}, and saturated vapor enthalpy, $h_{vapor\_sat\_dry}$. The enthalpy values are calculated using the XSTEAM function with the wet bulb temperature for the first two and stagnation temperature at AS1, T_{01_avg}, for the latter.

$$C_{p\_air}\left(T_{wetbulb} - T_{01\_avg}\right) + \omega_{sat}\left(h_{vapor\_sat\_wet} - h_{liq\_sat}\right) + \omega h_{liq\_sat} = \omega h_{vapor\_sat} \tag{2}$$

The equation can then be rearranged to solve for the humidity ratio ω.

$$\omega = \frac{C_{p\_air}\left(T_{wetbulb} - T_{01\_avg}\right) + \omega_{sat}\left(h_{vapor\_sat\_wet} - h_{liq\_sat}\right)}{h_{vapor\_sat\_dry} - h_{liq\_sat}} \tag{3}$$

The relative humidity ϕ is then calculated from relationship between the humidity ratio, the gas constants, average inlet stagnation pressure, and saturated vapor pressure.

$$\phi = \frac{\omega}{\left(R_{air} / R_{water}\right) + \omega} \frac{p_{01\_avg}}{P_{vapor\_sat}} \tag{4}$$

The constant pressure specific heat of air, C_{p_air}, is recalculated with the updated humidity ratio.

$$C_{p\_air} = \frac{C_{p\_ref} + \omega * C_{p\_vapor}}{1 + \omega} \tag{5}$$

The gas constant of air, R_{air}, is recalculated as well using the updated humidity ratio.

$$R_{air} = \frac{R_{ref} + \omega * R_{water}}{1 + \omega} \tag{6}$$

The constant volume specific heat of air is then calculated from the difference of C_{p_air} and R_{air}.

$$C_{v\_air} = C_{p\_air} - R_{air} \tag{7}$$

The heat capacity ratio, γ_{air} is then found from the ratio of constant pressure specific heat of air over constant volume specific heat of air. The heat capacity ratio is a key variable used in several equations to include the air density, pressure ratio and absolute exit velocity.

$$\gamma_{air} = \frac{C_{p\_air}}{C_{v\_air}} \tag{8}$$

The first time the program ran, these variables were set as reference constants. Once the program did a full run through the above constants were updated with accurate values. It is important to allow the program to run for a few seconds prior to taking any measurements to ensure that the relative humidity equations have processed.

APPENDIX C. INSTRUMENTATION SETUP

The pressure bricks require static IP configuration to connect to the PC. Prior to setting the PC to a static configuration, the NI cDAQ 9188XT must be assigned a static IP address with the same first three octets as the pressure bricks. The first step is to open NI MAX and reserve the network device, as shown in Figure 35. Then switch to the network settings tab.

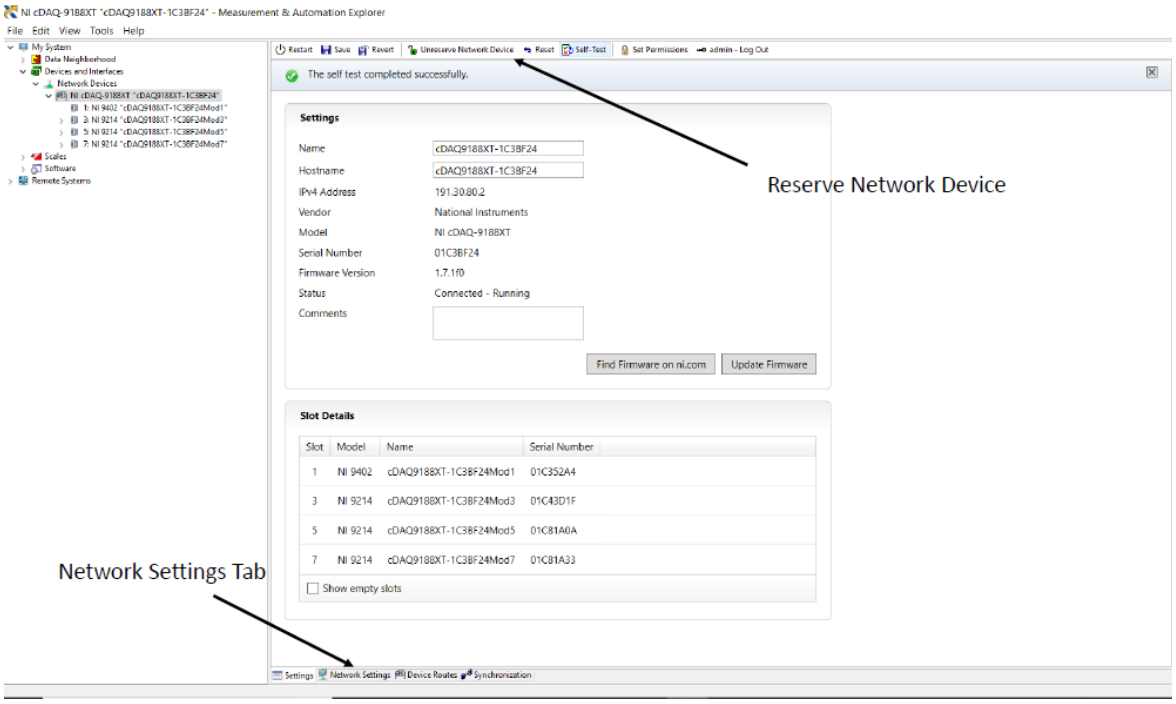

Figure 35. NI MAX Home Screen

On the network settings page the user can change the IP address to static and input a new IP address, as depicted in Figure 36. Once the static IP address is set, the PC must be configured with a static IP as well, as pictured in Figure 37.

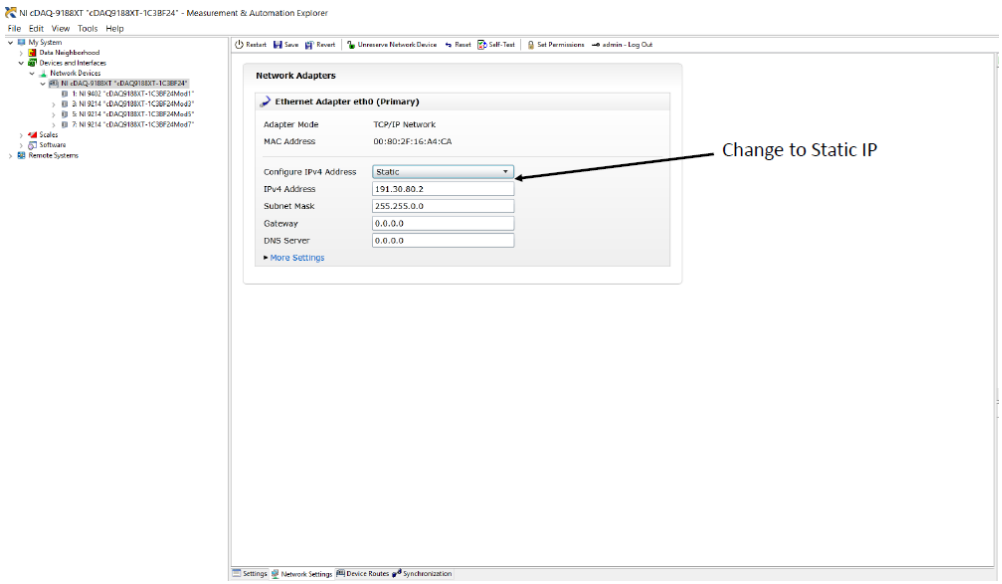

Figure 36. NI MAX Network Settings

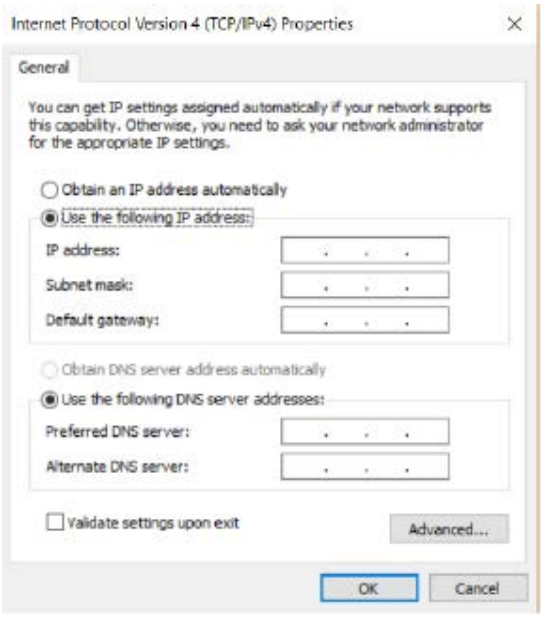

Figure 37. PC IPv4 Properties

Once the IP addresses for both the PC and the NI cDAQ 9188XT chassis were set, the chassis must perform a self-test. As shown in Figure 35, a self-test button is available and in the top menu bar. This ensured that the chassis and all modules were working properly. After the self-test, the chassis was ready to communicate with MATLAB.

Initial communication between the PC and the pressure bricks was achieved using the DSA Link 3 software program. This program allowed the user to verify the pressure brick was operational as well as perform a zero-calibration. Figures 38 and 39 display the software interface.

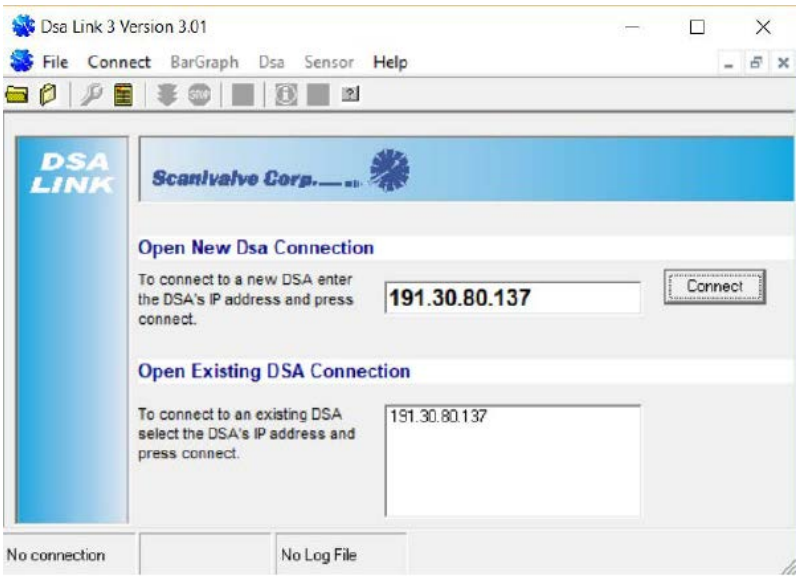

Figure 38. DSA Link 3 Connection Display

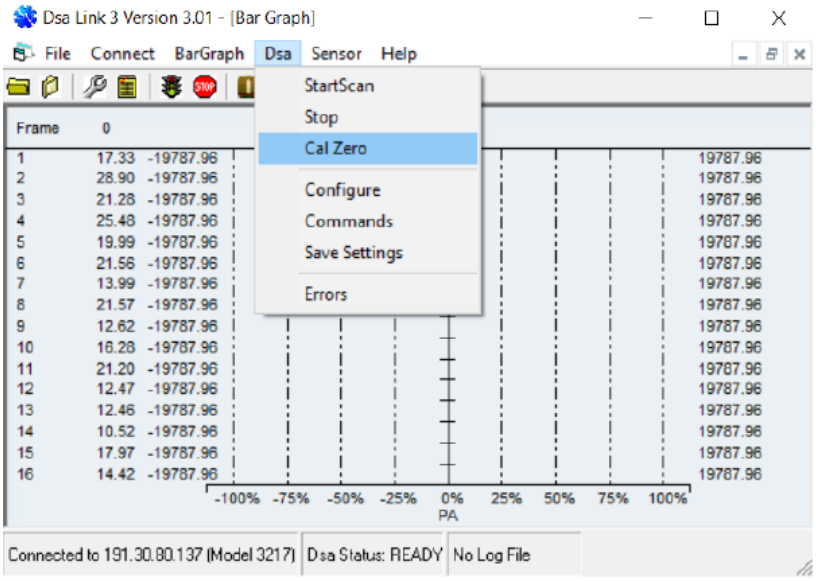

Figure 39. DSA Link 3 Bar Graph Display

The final MATLAB program uses only scripts and functions to communicate with the pressure bricks and NI cDAQ 9188XT chassis. National Instruments had provided ample example code to set up the chassis, but the pressure bricks had an internal command language with no MATLAB support. To determine how to connect the pressure bricks to MATLAB the Test and Measurement Tool App was utilized. An interface object was created with the IP address of the pressure brick. This opened a communication window that allowed the user to send commands specifically designed for the equipment. After connecting and sending sample commands, the user could view the actual MATLAB code utilized in the session log tab as depicted in Figures 40, 41, and 42.

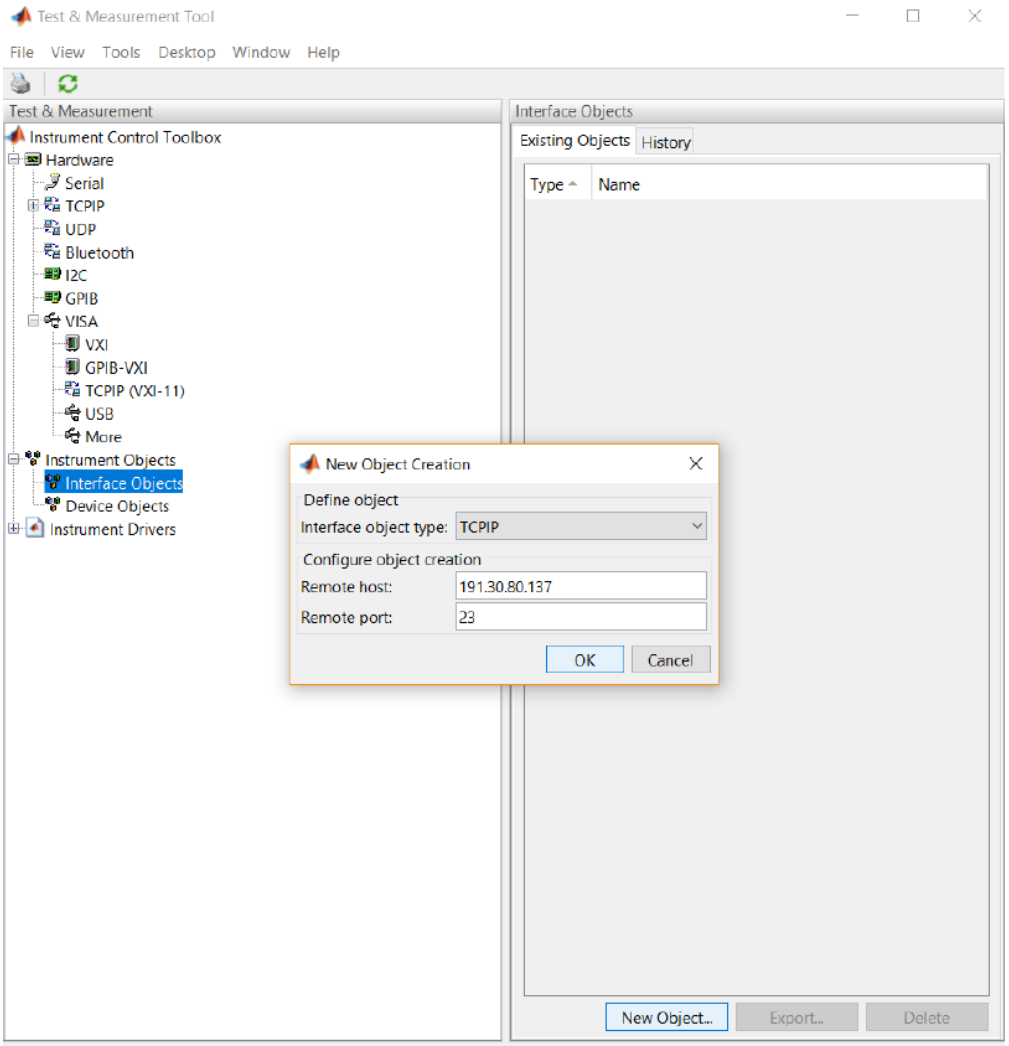

Figure 40. Test and Measurement Tool App New Object Creation

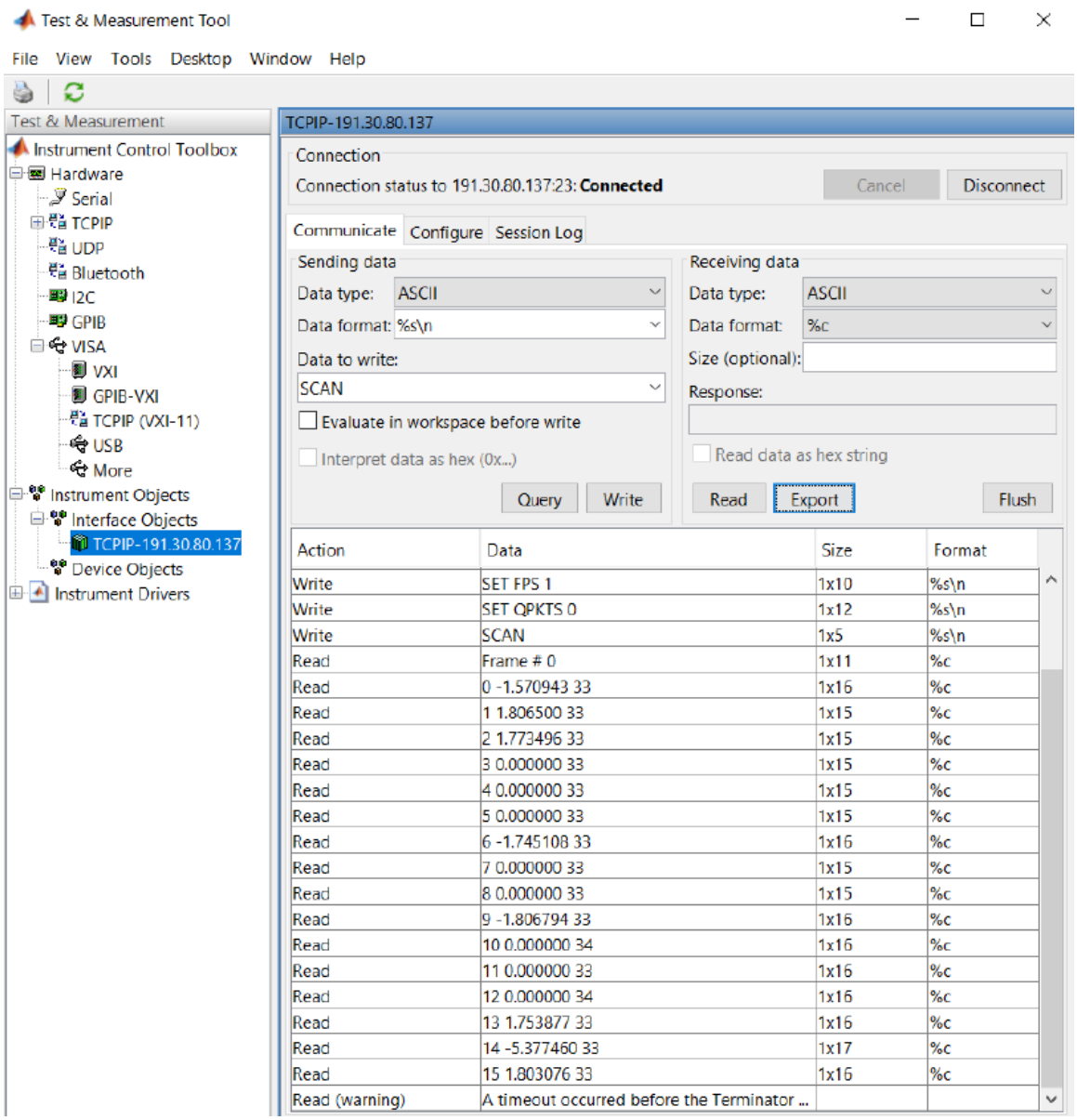

Figure 41. Test and Measurement Tool App Communication Tab

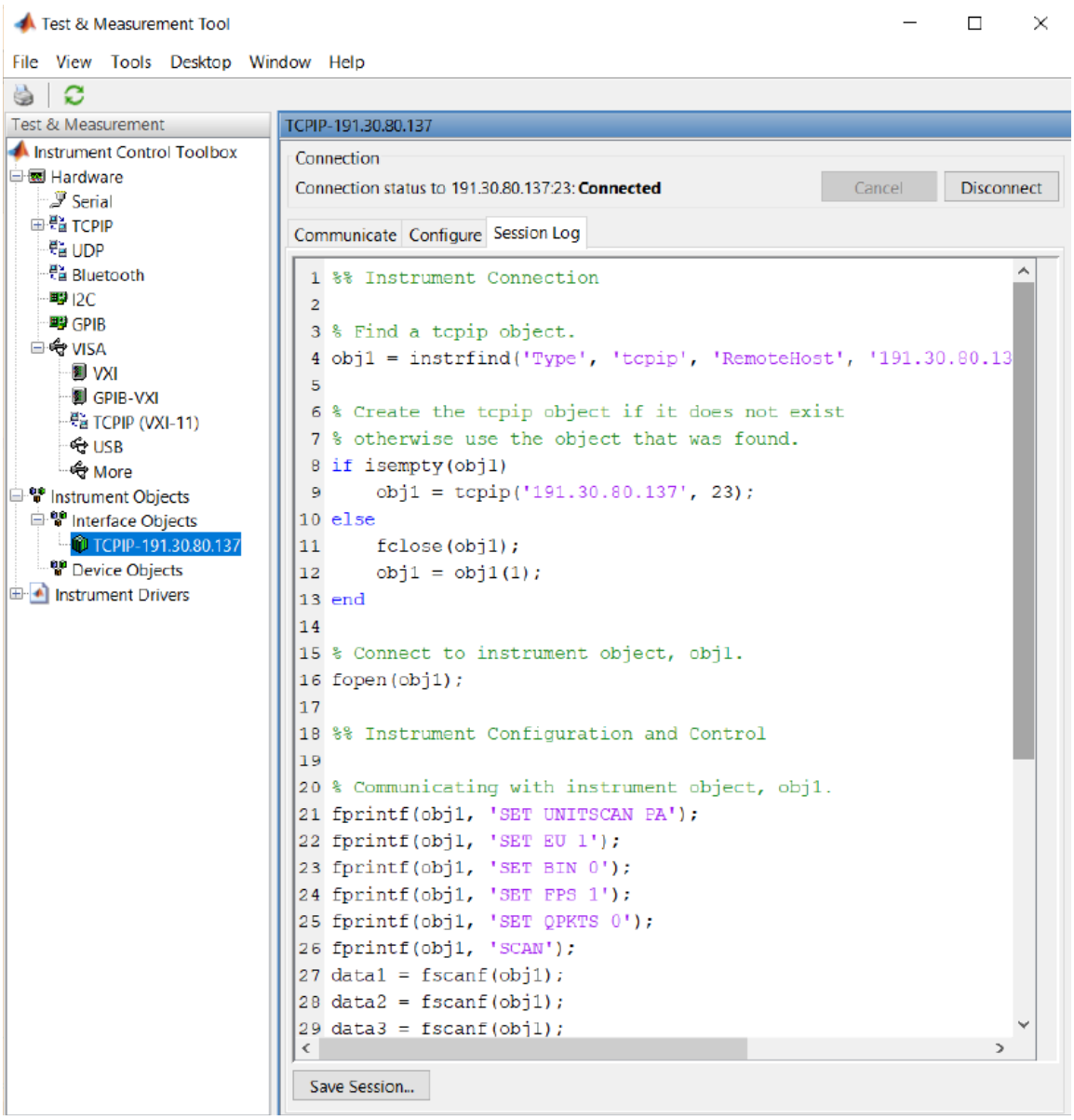

Figure 42. Test and Measurement Tool App Session Log Tab

APPENDIX D. INDIVIDUAL COMPONENT DESCRIPTION AND MATLAB CODE

A. MATLAB FUNCTIONS

Table 3 details the main functions used in MATLAB. The descriptions were adapted from Scanivalve's manual [8] and MathWorks documentation pages [15].

Table 3. Main MATLAB Functions

Name	Associated Equipment	Description
instrfind	DSA 3217	Reads serial port objects from memory to MATLAB workspace
tcpip	DSA 3217	Creates a TCPIP object
fopen	DSA 3217	Establishes a connection with the TCPIP object
fprintf	DSA 3217	Write function to communicate specific DSA commands
fscanf	DSA 3217	Read function that pulls packets from output buffer
SET UNITSCAN PA	DSA 3217	Sets scan units to Pascals
SET EU 1	DSA 3217	Sets scan units to Engineering Units
SET BIN 0	DSA 3217	Sets the format of the data packet to ASCII
SET FPS 1	DSA 3217	Sets the frames per scan to 1
SET QPKTS 0	DSA 3217	Frames will be discarded when the data buffer is full
SCAN	DSA 3217	Commands the DSA to scan the pressure sensors and send scan packets to the client
STOP	DSA 3217	Aborts current DSA operation
daq.getDevices	NI cDAQ 9188XT	Displays available data acquisition devices
daq.createSession	NI cDAQ 9188XT	Creates a data acquisition session
addAnalogInputChannel	NI cDAQ 9188XT	Adds an analog input channel
addCounterInputChannel	NI cDAQ 9188XT	Adds a counter input channel
s.Rate	NI cDAQ 9188XT	Specifies scans per second
s.DurationInSeconds	NI cDAQ 9188XT	Specifies the duration of acquisition
s.Channels	NI cDAQ 9188XT	Calls an array of channel objects associated with session object
s.ThermocoupleType	NI cDAQ 9188XT	Sets the thermocouple type

s.Units	NI cDAQ 9188XT	Sets the unit of thermocouple measurement
addlistener	NI cDAQ 9188XT	Calls a callback function every time the source object sends new data
s.Tag	MATLAB	Specifies a unique identifier of the object that can be called in separate functions
s.startBackground	NI cDAQ 9188XT	Starts background operations
s.Stop	NI cDAQ 9188XT	Stops background operations
global	MATLAB	Declares variables as global, and allows them to be called in any function
load()	MATLAB	Loads variables from file into the current workspace
save()	MATLAB	Saves the current workspace variables to file
uicontrol()	MATLAB	Create user interface control object
findobj()	MATLAB	Can locate an object based on the Tag identifier
s.String	MATLAB	Specifies the string array used by the object
dlmwrite	MATLAB	Writes to an ASCII-delimited file

B. DSA 3217 PRESSURE BRICK

Script file that connects to the pressure brick and outputs the raw pressure data and associated bar graph.

```
%% Pressure Brick Connection and Test
clear all
close all
clc
% Create figure to display pressure vs. port #
Pgraph = figure(1);
x=[1:1:16];
y=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
pbar = bar(x,y);
hold on;
grid on;
xlabel('Pressure Port #');
ylabel('Gauge Pressure [Pa]');
xticks([1:1:16]);
pause(0.1)
% Find a tcpip object.
obj1 = instrfind('Type', 'tcpip', 'RemoteHost', '191.30.80.137', 'RemotePort', 23,
'Tag', '');
% Create the tcpip object if it does not exist, otherwise use the object that was
found.
if isempty(obj1)
    obj1 = tcpip('191.30.80.137', 23);
else
    fclose(obj1);
```

```
    obj1 = obj1(1);
end
% Connect to instrument object, obj1.
fopen(obj1);
% Communicating with instrument object, obj1.
    % Set units to Pascals
    fprintf(obj1, 'SET UNITSCAN PA');
    % Set Engineering Units format
    fprintf(obj1, 'SET EU 1');
    % Set to ASCII data format
    fprintf(obj1, 'SET BIN 0');
    % Set Frames Per Scan
    fprintf(obj1, 'SET FPS 1');
    % Set Buffer to clear when full and continue sending data
    fprintf(obj1, 'SET QPKTS 0');
i=1;
for i=1:1:1000;
    % Sends Scan Command to DSA Pressure Brick
    fprintf(obj1, 'SCAN');
    % Discards first variable from packet which lists frame number,
    % if set to FPS = 1 all frames will be 0
    frame = fscanf(obj1);
for n=1:16
    % Exctracts Pressure and Temperature readings in order # 1 to 16 ports
    data1 = str2num(fscanf(obj1));
    % Takes Pressure readings and discards DSA temperature readings
    pdata(n)=data1(2)
end
% Updates pressure values in plot
```

```
pbar.YData=pdata;
pause(0.2)
i=i+1;
end
% If FPS is set to 0 (infinite) this will stop pressure brick in a secure manner
fprintf(obj1, 'STOP');
% Clear input buffer
flushinput(obj1);
% Clear output buffer
flushoutput(obj1);
```

C. NI 9214 THERMOCOUPLE MODULE

Script files that connects to the thermocouple module and outputs raw temperature data and associated bar graph.

```
%% NI 9214 Thermocouple Connection and Test
clear all
close all
clc
% Create figure to display temperature vs. port #
Tgraph = figure(1);
x=[1:1:25];
y=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
tbar = bar(x,y);
tbar.Tag= 'tbar';
hold on;
grid on;
```

```
xlabel('Thermocouple Port #');
ylabel('Temperature [K]');
xticks([1:1:25]);
yticks([0:50:400]);
ylim([0 400])
pause(0.1)
% Create DAQ Session
devices = daq.getDevices;
s = daq.createSession('ni');
% Add Thermocouple Modules to measure temperature
for i=0:9
addAnalogInputChannel(s,'cDAQ9188XT-1C3BF24Mod3',i, 'Thermocouple');
end
for i=0:9
addAnalogInputChannel(s,'cDAQ9188XT-1C3BF24Mod5',i, 'Thermocouple');
end
for i=0:4
addAnalogInputChannel(s,'cDAQ9188XT-1C3BF24Mod7',i, 'Thermocouple');
end
% Set rate [scans per second]
s.Rate = 1
% Set time of run, to set as infinite use s.IsContinuous = true
s.DurationInSeconds = 1000;
% Set up temperature units (Required to run script)
tc = s.Channels();
for i=1:25
tc(i).ThermocoupleType = 'J';
tc(i).Units = 'Kelvin';
end
```

```
% Add listener with function for reading thermocouples
lh=addlistener(s,'DataAvailable', @temp_readings_graph);
% Start background acquisition
s.startBackground();
% Use delete(lh) to end script early or if continuous run being used

%% Add listener function for thermocouple module
function temp_readings_graph(src,event)
% Reads temperature data from modules
tdata=event.Data
% Identifies tag for bar graph
h=findobj('Tag','tbar');
% Updates temperature values of bar graph
h.YData=tdata;
pause(1);
end
```

D. NI 9402 COUNTER-TOTALIZER MODULE

Script files that connects to the counter-totalizer module and outputs pulse width and RPM raw data.

```
%% NI 9402 Counter-Totalizer Connection and Test
clear all
close all
clc
% Create DAQ Session
s = daq.createSession('ni');
% Add BNC Module to measure Pulse Width
```

```
addCounterInputChannel(s,'cDAQ9188XT-1C3BF24Mod1', 0 , 'PulseWidth');
% Add Additional Module (Required for external clock sync in background function)
addAnalogInputChannel(s,'cDAQ9188XT-1C3BF24Mod3',0, 'Thermocouple');
% Set up temperature units (Required to run script)
tc = s.Channels(2);
tc.ThermocoupleType = 'J';
tc.Units = 'Kelvin';
% Set time of run, to set as infinite use s.IsContinuous = true
s.DurationInSeconds = 100;
% Set rate [scans per second]
s.Rate = 1;
% Add listener with function for reading Pulse Width and RPM
lh=addlistener(s,'DataAvailable', @pulsewidth);
% Start background acquisition
s.startBackground()
% Use delete(lh) to end script early or if continuous run being used


%% Add listener function for counter-totalizer module
function pulsewidth(src,event)
% Reads pulsewidth data from module and converts to RPM
pw = event.Data(1);
RPM = 60/pw
end
```

APPENDIX E. DAS DESCRIPTION AND MATLAB CODE

A. DAS PROGRAM DESCRIPTION

Table 4 describes the specific actions occurring within each MATLAB script and function.

Table 4. MATLAB Script Overview

1. Run main script
1.1 Constants script is called to run
1.1.1 User input parameters from previous test run are loaded
1.1.2 Required constants are calculated
1.1.3 Constants variables are saved as a workspace file
1.2 Pressure parameters function is called to run
1.2.1 Pressure bricks are connected to MATLAB
1.2.2 Pressure bricks are given setup commands
1.2.3 Pressure bricks perform zero-calibration
1.3 Data acquisition session is created with NI-DAQ9188XT
1.3.1 NI 9402 module configured for pulse width measurements
1.3.2 NI 9214 modules configured for thermocouple measurements
1.3.3 Measurement parameters and continuous listening functions set
1.4 User input constants figure created
1.4.1 Input fields for atmospheric pressure, percent of speed, design speed, calibration pressure, and wet bulb temperature are created
1.5 Raw pressure and temperature figure created
1.5.1 Display fields for pressure and temperature data created
1.6 HMI figure created
1.6.1 User command pushbuttons created
1.6.2 Static text displays for pertinent data created
1.6.3 Downstream pressure and temperature profile graphs created
1.6.4 Save pushbutton disabled
2. User input constants pushbutton function (Apply)
2.1 Saves user input data as workspace file
2.2 Calls on constants script
2.2.1 Loads user input data and preforms constants calculations
3. Run pushbutton function
3.1 Load constants workspace file
3.2 Disable user input constants figure and reset run number pushbutton
3.3 Enable save pushbutton
3.4 Start NI data acquisition in background

3.5 Send scan command to pressure bricks
3.6 Receive raw pressure data from pressure bricks
3.7 Convert raw pressure to absolute pressure
3.8 Receive temperature and rotational speed data from NI instruments
3.9 Perform torque calculations
3.10 Perform RPM calculations
3.11 Performs upstream calculations
3.12 Performs mass averaged pressure and temperature calculations
3.13 Performs mass flow calculations
3.14 Perform pressure ratio calculations
3.15 Perform power calculations
3.16 Perform relative humidity calculations
3.17 Update raw pressure and temperature figure display fields
3.18 Update HMI figure display fields and graphs
3.19 Repeat steps 3.5-3.18 until interrupted by outside command
4. Stop pushbutton function
4.1 Stop NI data acquisition in background
4.2 Change trigger variable to stop Run function while loop
4.3 Disable save pushbutton
4.4 Enable user input constants figure and reset run number
5. Save pushbutton function
5.1 Create .txt file with date and run speed as title (Only for first time pressed)
5.2 Add heading for each variable to be written to file (Only for first time pressed)
5.3 Write run number and required variables to file
5.4 Update run number
6. Reset run number pushbutton function
6.1 Resets run number to “1”

B. DSA MATLAB CODE

1. Main.m

The Main.m script is used to initialize the DAS program.

```
%% Main Run Script
clear all
close all
clc
% Set the required global variables
global runnum ni_set pline tline save_trigger firstsave
% Preloaded input constants from the previous run
load('preloaded_constants');
Atmos=num2str(Atmos_Hg);
Spdpercent=num2str(Spdpct);
DRpm=num2str(DesignRpm);
Calpress=num2str(Calpress_Hg);
Twbulb=num2str(Twetbulb);
%
pdata = [];
tdata = [];
save_trigger = 0;
firstsave = 0;
% Connect DSA pressure bricks and set parameters
Pparameters_multibrick;
% Create DAQ Session
devices = daq.getDevices;
```

```
ni_set = daq.createSession('ni');
% Add Counter-totalizer & Thermocouple Modules to measure rotational speed &
temperature
addCounterInputChannel(ni_set,'cDAQ9188XT-1C3BF24Mod1',0, 'PulseWidth');
for itemp = 0:9;
addAnalogInputChannel(ni_set,'cDAQ9188XT-1C3BF24Mod3',itemp, 'Thermocouple');
end
for itemp = 0:9;
addAnalogInputChannel(ni_set,'cDAQ9188XT-1C3BF24Mod5',itemp, 'Thermocouple');
end
for itemp = 0:4;
addAnalogInputChannel(ni_set,'cDAQ9188XT-1C3BF24Mod7',itemp, 'Thermocouple');
end
% Set rate [scans per second]
ni_set.Rate = 2;
% Set time of run, to set as infinite use s.IsContinuous = true
ni_set.IsContinuous = true;
% Set temperature units
tc = ni_set.Channels(2:26);
for i=1:25
tc(i).ThermocoupleType = 'J';
tc(i).Units = 'Kelvin';
end
% Add listener with function for reading Pulse Width and RPM
lh_pw=addlistener(ni_set,'DataAvailable', @Pulsewidth_main);
% Add listener with function for reading thermocouples
lh_t=addlistener(ni_set,'DataAvailable', @Temp_readings_main);
disp('NI DAQ Parameters set. Ready to scan...')
disp('Previous Constants loaded. Make changes now or start run...')
```

```
% Figure 1

cfig = figure(1);
  cfig.Visible = 'off';
 cfig.Units='Normalized';

 constant_title = uicontrol('Style','text','Parent',cfig,...
    'Units','normalized',...
    'Position',[0.35 0.88 0.3 0.07],...
    'FontWeight','bold','FontUnits','normalized','String','Constants');

 edit_atmp = uicontrol('Style','edit','Parent',cfig,...
    'Units','normalized',...
    'Position',[0.2 0.78 0.3 0.07],...
    'foregroundcolor','b','FontUnits','normalized',...
    'String',Atmos,...
    'Callback','','Tag','atmp');

 edit_percspd = uicontrol('Style','edit','Parent',cfig,...
    'Units','normalized',...
    'Position',[0.2 0.68 0.3 0.07],...
    'foregroundcolor','b',...
    'String',Spdpercent,'FontUnits','normalized',...
    'Callback','','Tag','spd');

 edit_dspd = uicontrol('Style','edit','Parent',cfig,...
    'Units','normalized',...
    'Position',[0.2 0.58 0.3 0.07],...
    'foregroundcolor','b','FontUnits','normalized',...
```

```
    'String',DRpm,...
    'Callback','','Tag','design');

edit_calp = uicontrol('Style','edit','Parent',cfig,...
    'Units','normalized',...
    'Position',[0.2 0.48 0.3 0.07],...
    'foregroundcolor','b','FontUnits','normalized',...
    'String',Calpress,...
    'Callback','','Tag','cal');

edit_twetbulb = uicontrol('Style','edit','Parent',cfig,...
    'Units','normalized',...
    'Position',[0.2 0.38 0.3 0.07],...
    'foregroundcolor','b','FontUnits','normalized',...
    'String',Twbulb,...
    'Callback','','Tag','wetbulb');

Bt_atmp = uicontrol('Style','text','Parent',cfig,...
    'Units','normalized',...
    'Position',[0.52 0.78 0.5 0.07],...
    'backgroundcolor','g',...
    'foregroundcolor','b',...
    'FontWeight','bold','FontUnits','normalized','String','Atmospheric Pressure
[Hg]');

Bt_percspd = uicontrol('Style','text','Parent',cfig,...
    'Units','normalized',...
    'Position',[0.52 0.68 0.5 0.07],...
    'backgroundcolor','g',...
```

```
    'foregroundcolor','b',...
    'FontWeight','bold','FontUnits','normalized','String','Percent of Speed [%]');

Bt_dspd = uicontrol('Style','text','Parent',cfig,...
    'Units','normalized',...
    'Position',[0.52 0.58 0.5 0.07],...
    'backgroundcolor','g',...
    'foregroundcolor','b',...
    'FontWeight','bold','FontUnits','normalized','String','Design Speed [rpm]');

 Bt_calp = uicontrol('Style','text','Parent',cfig,...
    'Units','normalized',...
    'Position',[0.52 0.48 0.5 0.07],...
    'backgroundcolor','g',...
    'foregroundcolor','b',...
    'FontWeight','bold','FontUnits','normalized','String','Calibration Pressure [in
Hg]');

Bt_twetbulb = uicontrol('Style','text','Parent',cfig,...
    'Units','normalized',...
    'Position',[0.52 0.38 0.5 0.07],...
    'backgroundcolor','g',...
    'foregroundcolor','b',...
    'FontWeight','bold','FontUnits','normalized','String','T WetBulb [C]');

Bp_constants = uicontrol('Style','pushbutton','Parent',cfig,...
    'Units','normalized',...
    'Position',[0.25 0.2 0.2 0.1],...
    'String','Apply',...
```

```
    'FontWeight','bold','FontSize',15,'Callback',@Constants,'Tag','applybtn');
% Figure 2
% Raw Data Figure Display
rawdatafig = figure(2);
rawdatafig.Visible = 'off';
rawdatafig.Units='Normalized';

rawdata_title = uicontrol('Style','text','Parent',rawdatafig,...
   'Units','normalized',...
   'Position',[0.35 0.93 0.3 0.07],...
   'FontWeight','bold','FontUnits','normalized','String','Raw Data');

pressure_title = uicontrol('Style','text','Parent',rawdatafig,...
   'Units','normalized',...
   'Position',[0.11 0.9 0.3 0.07],...
   'FontWeight','bold','FontUnits','normalized','String','Pressure [Pa]');

temp_title = uicontrol('Style','text','Parent',rawdatafig,...
   'Units','normalized',...
   'Position',[0.61 0.9 0.3 0.07],...
   'FontWeight','bold','FontUnits','normalized','String','Temperature [K]');

p_readings = uicontrol('Style','text','Parent',rawdatafig,...
  'Units','Normalized','Position',[0 0 .5 .9],'min',0,'max',2,...
  'enable','inactive','FontUnits','normalized','Tag','rawpressure');

t_readings = uicontrol('Style','text','Parent',rawdatafig,...
  'Units','Normalized','Position',[.5 0 .5 .9],'min',0,'max',2,...
  'enable','inactive','FontUnits','normalized','Tag','rawtemp');
```

```
 % Figure 3
  runnum = 1;
  run_count = num2str(runnum);
  hmifig = figure(3);
  hmifig.Visible = 'off';
  % HMI Figure Display
  hmifig.Units='Normalized';
hold on

Run_btn = uicontrol('Style','pushbutton','Parent',hmifig,...
  'String','Run','Units','Normalized','Position',[0 .9 .18 .08],...
  'FontWeight','bold','FontUnits','normalized','Callback',@Run_Startmulti);

Stop_btn = uicontrol('Style', 'pushbutton','Parent',hmifig,...
  'String','Stop','Units','Normalized','Position',[0 .8 .18 .08],...
  'FontWeight','bold','FontUnits','normalized','Callback',@Run_Stopmulti);

Resetnum_btn = uicontrol('Style', 'pushbutton','Parent',hmifig,...
  'String','Reset Run #','Units','Normalized','Tag','resetbtn',...
  'FontWeight','bold','FontUnits','normalized','Position',[0 .7 .18
.08],'Callback',@Reset_num);

Save_btn = uicontrol('Style', 'pushbutton','Parent',hmifig,...
  'String','Save','Units','Normalized','Enable','off','Tag','savebtn',...
  'FontWeight','bold','FontUnits','normalized','Position',[0 .6 .18
.08],'Callback',@Save_Run);

PoPs_txt=uicontrol('Style','text','Parent',hmifig,...
```

```
  'Units','normalized','Position',[.19 .92 .1 .04],...
  'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Po-
Ps:');

PoPs_num=uicontrol('Style','text','Parent',hmifig,...
  'Units','normalized','Position',[.29 .92 .15 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','po_ps');

relhum_txt=uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[.19 .84 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','RelHum
id [%]:');

relhum_num=uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[.29 .84 .15 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','humid');

run_txt=uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[.19 .76 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Run
#');

run_num=uicontrol('Style','text','Parent',hmifig,...
```

```
    'Units','normalized','Position',[.29 .76 .15 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String',run_cou
nt,'Tag','runnumber');

rpm_txt = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[.19 .68 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','RPM');

rpm_num = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[.29 .68 .15 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','rpmvalue');

rpmint_txt = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[.19 .6 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Intend
ed RPM');

rpmint_num = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[.29 .6 .15 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','rpmintended');

rpmref_txt = uicontrol('Style','text','Parent',hmifig,...
```

```
    'Units','normalized','Position',[0.19 .52 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Referr
ed RPM');

rpmref_num = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[.29 .52 .15 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','rpmref');

raw_title = uicontrol('Style','text','Parent',hmifig,...
   'Units','normalized',...
   'Position',[0.05 0.36 0.2 0.05],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Raw');

scale_title = uicontrol('Style','text','Parent',hmifig,...
   'Units','normalized',...
   'Position',[0.3 0.36 0.2 0.05],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','Scale'
);

pratioraw_txt = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.02 0.31 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Pressu
re Ratio');
```

```
pratioraw_num = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.12 0.31 .13 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','praw');

massflowraw_txt = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.02 0.21 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Mass
Flow [kg/s]');

massflowraw_num = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.12 0.21 .13 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','massfraw');

pratioscale_txt = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.27 0.31 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Pressu
re Ratio');

pratioscale_num = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.37 0.31 .13 .04],...
```

```
'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','pscale');

massflowscale_txt = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.27 0.21 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Mass
Flow [kg/s]');

massflowscale_num = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.37 0.21 .13 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','massfscale');

axial_txt = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.02 0.11 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Axial
Inlet [m/s]');

axial_num = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.12 0.11 .13 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','axial');

Relinlet_txt = uicontrol('Style','text','Parent',hmifig,...
```

```
    'Units','normalized','Position',[0.02 0.01 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Rel
Inlet [Mach]');

Relinlet_num = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.12 0.01 .13 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','mach');

Eff_Stage_txt = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.27 0.11 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Eff
Stage');

Eff_Stage_num = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.37 0.11 .13 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','stage');

Eff_Rot_txt = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.27 0.01 .1 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','g','String','Eff
Rot Only');
```

```
Eff_Rot_num = uicontrol('Style','text','Parent',hmifig,...
    'Units','normalized','Position',[0.37 0.01 .13 .04],...

'FontWeight','bold','FontUnits','normalized','backgroundcolor','w','String','','Tag
','rot');

[Atmos_Hg,Spdpct,DesignRpm,Calpress_Hg,Twetbulb,Pdist,Do3,P_HubTip,comp_out_hub_in,
HubTip,Rho_Hg,TWetBulb,Dialin_Hg,Inch_m,g,Po_3_index,Po_1_index,Ps_1_case_index,Ps_
3_hub_index,Ps_3_case_index,To_1_index,d_p1,AnH,To_3_index, T_dist,T_HubTip,
lbs_N,Intended_RPM,CToK,Torque_coefficients,
P_ref,T_WetBulb,Referred_T,T_ref,Gam_Air, R_air,R_Ref, R_wat,
Cp_air,Cp_Ref,Dnoz,Anoz, Cnoz,Fan_dia, Dinf, Ainf, DNose,ANose, Gam_Ref,
RelHumid]=Calc_constants();
py=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
tx=[T_HubTip,HubTip];
ty=[0 0 0 0 0 0 0 0 0 0];
  DSTRMP = subplot(2,2,2);
  hold on;
    pline = animatedline(P_HubTip,py);
    pline.Tag = 'pline';
    title('Pressure');
    ylabel('Pressure [Pa]')
    grid on;
    DSTRMP.YLimMode = 'auto';
  DSTRMT = subplot(2,2,4);
  hold on;
    tline= animatedline(tx,ty);
    tline.Tag = 'tline';
    title('Temperature');
```

```
    ylabel('Temperature [K]');
    grid on;
    DSTRMT.YLimMode = 'auto';

% Turn on figures
hmifig.Visible = 'on';
rawdatafig.Visible = 'on';
cfig.Visible = 'on';
```

2. Pparameters_multibrick.m

The pressure bricks were connected and formatted using the Pparameters.m script.

```
%% DSA pressure brick connection and formatting
function Pparameters_multibrick()
% Find a tcpip object.
global pbrick1 pbrick2 pbrick3 pbrick4
% pbrick1 = instrfind('Type', 'tcpip', 'RemoteHost', '191.30.80.137',
% 'RemotePort', 23, 'Tag', ''); Used for test pressure brick
pbrick1 = instrfind('Type', 'tcpip', 'RemoteHost', '172.20.120.230', 'RemotePort',
23, 'Tag', '');
pbrick2 = instrfind('Type', 'tcpip', 'RemoteHost', '172.20.120.231', 'RemotePort',
23, 'Tag', '');
pbrick3 = instrfind('Type', 'tcpip', 'RemoteHost', '172,20,120,232', 'RemotePort',
23, 'Tag', '');
pbrick4 = instrfind('Type', 'tcpip', 'RemoteHost', '172.20.120.235', 'RemotePort',
23, 'Tag', '');
```

```
% Create the tcpip object if it does not exist
% otherwise use the object that was found.
if isempty(pbrick1)
    pbrick1 = tcpip('172.20.120.230', 23);
else
    fclose(pbrick1);
    pbrick1 = pbrick1(1);
end
if isempty(pbrick2)
    pbrick2 = tcpip('172.20.120.231', 23);
else
    fclose(pbrick2);
    pbrick2 = pbrick2(1);
end
if isempty(pbrick3)
    pbrick3 = tcpip('172,20,120,232', 23);
else
    fclose(pbrick3);
    pbrick3 = pbrick3(1);
end
if isempty(pbrick4)
    pbrick4 = tcpip('172.20.120.235', 23);
else
    fclose(pbrick4);
    pbrick4 = pbrick4(1);
end
%Connect to instrument object, obj1.
fopen(pbrick1);
fopen(pbrick2);
```

```
fopen(pbrick3);
fopen(pbrick4);
% Communicating with instrument object, obj1.
    fprintf(pbrick1, 'SET UNITSCAN PA');
    fprintf(pbrick1, 'SET EU 1');
    fprintf(pbrick1, 'SET BIN 0');
    fprintf(pbrick1, 'SET FPS 1');
    fprintf(pbrick1, 'SET QPKTS 0');
    fprintf(pbrick2, 'SET UNITSCAN PA');
    fprintf(pbrick2, 'SET EU 1');
    fprintf(pbrick2, 'SET BIN 0');
    fprintf(pbrick2, 'SET FPS 1');
    fprintf(pbrick2, 'SET QPKTS 0');
    fprintf(pbrick3, 'SET UNITSCAN PA');
    fprintf(pbrick3, 'SET EU 1');
    fprintf(pbrick3, 'SET BIN 0');
    fprintf(pbrick3, 'SET FPS 1');
    fprintf(pbrick3, 'SET QPKTS 0');
    fprintf(pbrick4, 'SET UNITSCAN PA');
    fprintf(pbrick4, 'SET EU 1');
    fprintf(pbrick4, 'SET BIN 0');
    fprintf(pbrick4, 'SET FPS 1');
    fprintf(pbrick4, 'SET QPKTS 0');
    disp('Pressure Parameters set. Ready to scan...')
end
```

3. Calc_Contstants.m

The system constants were calculated with Cal_Constants.m script.

```
%% Constants preload
function
[Atmos_Hg,Spdpct,DesignRpm,Calpress_Hg,Twetbulb,Pdist,Do3,P_HubTip,comp_out_hub_in,
HubTip,Rho_Hg,TWetBulb,Dialin_Hg,Inch_m,g,Po_3_index,Po_1_index,Ps_1_case_index,Ps_
3_hub_index,Ps_3_case_index,To_1_index, d_p1,AnH,To_3_index, T_dist,T_HubTip,
lbs_N,Intended_RPM,CToK, Torque_coefficients,
P_ref,T_WetBulb,Referred_T,T_ref,Gam_Air, R_air, R_Ref, R_wat,
Cp_air,Cp_Ref,Dnoz,Anoz, Cnoz,Fan_dia, Dinf, Ainf, DNose, ANose, Gam_Ref, RelHumid]
= Calc_constants()
load('preloaded_constants');
Pdist=[0.05 0.09 0.17 0.27 0.40 0.49 0.59 0.78 0.80 0.90 0.98 1.09 1.20 1.30 1.40
1.50 1.61 1.73 1.78 1.80];
Do3 = 11.308; % Comp_out_dia [in]
P_HubTip = [];
for i=1:20;
    P_HubTip(i)=1-2*(Pdist(i))/(Do3);
end
comp_out_hub_in = 7.425;
HubTip = comp_out_hub_in/Do3;
Rho_Hg = 13550; % Mercury density [kg/m^3]
TWetBulb = 285; % Wetbulb [in K]
Dialin_Hg = 10; % Cal Press [in Hg]
Inch_m = 0.0254; % Inch to meters [m]
g = 9.81; % Gravitational constant [m/s^2]
Po_3_index = [11:1:30]; % port #
```

```
Po_1_index = [5 6]; % port #
Ps_1_case_index = [7 8];
Ps_3_hub_index = [35 36 37];
Ps_3_case_index = [9 10];
To_1_index = [15 13];%remove
d_p1 = 0;
AnH = (Do3-comp_out_hub_in)/2; %Annulus Height
To_3_index = [1 8 7 6 5 3 9 2 10]; %port #
T_dist = [100 91.6 80.4 75.8 63.3 43.9 31.1 27.6 7.9];
T_HubTip = [];
for i = 1:9;
    T_HubTip(i) = (Do3-2*AnH*(1-T_dist(i)/100))/Do3;
end
lbs_N = 4.44822; %pound force in Newton [N]
Intended_RPM = DesignRpm*Spdpct/100;
rpmint = findobj('Tag','rpmintended');
rpmint.String=sprintf('%0.1f',Intended_RPM);
CToK = 273.15;% Celcius to Kelvin
KToC = -273.15; % Kelvin to Celcius
BarToPa=100000;
PaToBar=1/100000;
Torque_coefficients = [-1.064 5113];
P_ref = 101300; % Referred Press [Pa]
T_WetBulb = CToK+Twetbulb; % T Wetbulb [K]
Referred_T = 518.7; % Referred Temp [R];
T_ref = Referred_T/1.8;
Gam_Air = 1.4;
R_air = 287; % [J/kgK]
R_Ref = 287; % [J/kgK]
```

```
R_wat = 461.5; % R wat vap[J/kgK]
Cp_air = Gam_Air*R_air/(Gam_Air-1);
Cp_Ref = Gam_Air*R_air/(Gam_Air-1);
Dnoz = 12; % Nozzle Diam [in]
Anoz = (pi*(Dnoz*Inch_m)^2)/4;
Cnoz =  1.03; % Noz discharge coeff
Fan_dia = 11.308; % Fan diameter [in]
Dinf = Fan_dia*Inch_m; % Fan diameter [m]
Ainf = (pi*Dinf^2)/4; % Rotor Area [m^2]
DNose = 3.6*Inch_m;
ANose = (pi*DNose^2)/4;
Gam_Ref = 1.4;
RelHumid = 0;
save('Current_constants');
end
```

4. Constants.m

User input constants are loaded with Constants.m script.

```
%% User input Constants Function
function Constants(Bp_constants,~)
h=findobj('Tag','atmp');
Atmos_Hg = h.String;
Atmos_Hg = str2num(Atmos_Hg);
h=findobj('Tag','spd');
Spdpct = h.String;
Spdpct = str2num(Spdpct);
h=findobj('Tag','design');
DesignRpm = h.String;
DesignRpm = str2num(DesignRpm);
h=findobj('Tag','cal');
Calpress_Hg =h.String;
Calpress_Hg = str2num(Calpress_Hg);
h=findobj('Tag','wetbulb');
Twetbulb = h.String;
Twetbulb = str2num(Twetbulb);
save('preloaded_constants','Atmos_Hg','Spdpct','DesignRpm','Calpress_Hg','Twetbulb'
)
Calc_constants
end
```

5. Run_Startmulti.m

Data acquisition occurs within Run_Startmulti.m script.

```
%% Run DAS Function
function [pdata] = Run_Startmulti(Run_btn,~)
global ni_set pressurestat  pbrick1 pbrick2 pbrick3 pbrick4 tdatar RPMr
save_trigger runnum firstsave
load('Current_constants');
h=findobj('Tag','atmp');
h.Enable = 'off';
h=findobj('Tag','spd');
h.Enable = 'off';
h=findobj('Tag','design');
h.Enable = 'off';
h=findobj('Tag','cal');
h.Enable = 'off';
h=findobj('Tag','wetbulb');
h.Enable = 'off';
h=findobj('Tag','applybtn');
h.Enable = 'off';
h=findobj('Tag','savebtn');
h.Enable = 'on';
h=findobj('Tag','resetbtn');
h.Enable = 'off';
refrpm = findobj('Tag','rpmref');
pops = findobj('Tag','po_ps');
axial=findobj('Tag','axial');
```

```
machin =findobj('Tag','mach');
effstage=findobj('Tag','stage');
effrot=findobj('Tag','rot');
mfraw=findobj('Tag','massfraw');
mfscale=findobj('Tag','massfscale');
prraw=findobj('Tag','praw');
prscale=findobj('Tag','pscale');
rhumid=findobj('Tag','humid');
pline=findobj('Tag','pline');
tline=findobj('Tag','tline');
pressurestat=1;
% i=1;
ni_set.startBackground();
pause(0.5);
while pressurestat == 1;
pause(1);
fprintf(pbrick1, 'SCAN');
frame = fscanf(pbrick1);
for n=1:16
data1 = str2num(fscanf(pbrick1));
pdata(n)=data1(2);
end
fprintf(pbrick2, 'SCAN');
frame = fscanf(pbrick2);
for n=17:32
data1 = str2num(fscanf(pbrick2));
pdata(n)=data1(2);
end
fprintf(pbrick3, 'SCAN');
```

```
frame = fscanf(pbrick3);
for n=33:48
data1 = str2num(fscanf(pbrick3));
pdata(n)=data1(2);
end
fprintf(pbrick4, 'SCAN');
frame = fscanf(pbrick4);
for n=49:64
data1 = str2num(fscanf(pbrick4));
pdata(n)=data1(2);
end
P_Atmos = Rho_Hg*g*Atmos_Hg*Inch_m;
pdata=pdata+P_Atmos;
for i=11:30
    pdata(i)=pdata(i);
   % pdata(i)=pdata(i)+50000; Used for testing without TCR running
end
tdata = tdatar;
% tdata(1)=tdata(1)+150; Used for testing without TCR running
% tdata(8)=tdata(8)+150;
% tdata(7)=tdata(7)+150;
% tdata(6)=tdata(6)+150;
% tdata(5)=tdata(5)+150;
% tdata(3)=tdata(3)+150;
% tdata(9)=tdata(9)+150;
% tdata(2)=tdata(2)+150;
% tdata(10)=tdata(10)+150;
RPM = RPMr;
%% Calculations_main
```

```
% ScanOneTorque
Torque_Nm = -(polyval(Torque_coefficients,0.1)); %off by one decimal place
Torque_inlbs = Torque_Nm/(Inch_m*lbs_N);
% Calc_RPM_ref
To1 = [tdata(To_1_index)];
To1_avg = mean(To1);%To_inf
Theta_ref = (Gam_Air*R_air*To1_avg)/(Gam_Ref*R_Ref*T_ref);
RPM_ref = RPM/sqrt(Theta_ref);
%%%%refrpm
Scale_RPM = Intended_RPM/RPM_ref;
% Calc Upstream
Ps1_case=[pdata(Ps_1_case_index)];
Ps1_avg=mean(Ps1_case); %ps_inf
Po1=[pdata(Po_1_index)];
Po1_avg=mean(Po1); %po_inf
d_P1 = Po1_avg-Ps1_avg;
%%%%pops
Delta_ref=(Po1_avg*Gam_Air)/(P_ref*Gam_Air);
U_inf = sqrt(abs(1-(Ps1_avg/Po1_avg)^((Gam_Air-
1)/Gam_Air)))*sqrt(2*Cp_air*To1_avg); %Inlet velocity
%%%%axial
UTip = (2*pi*RPM/60)*Dinf/2;
AR = (Ainf-ANose)/Ainf;
vo = sqrt(2*Cp_air*To1_avg);
X1 = U_inf/vo;
U1 = vo*(X1/AR);
v = (U1^2+UTip^2); % Relative Inlet Velocity
vo2 = (2*Cp_air*To1_avg); % Stagnation Velocity
X2 = v/vo2;
```

```
MachinRel = sqrt(2*X2/((Gam_Air-1)*(1-X2)));
%%%%%machin
% Calc_Mass_Avg
d_HubTip = (P_HubTip(1:(end-2))-P_HubTip(3:end))/2; %Core Values  (p(1:18)-
p(3:20))/2 --->(18 numbers)
d_HubTip = [(1-P_HubTip(2))/2, d_HubTip]; % Tip value  [(1-
P(2))/2(1number);(18numbers)--->(19 numers)
d_HubTip = [d_HubTip, (P_HubTip(end-1)-HubTip)/2]; % Hub Value
Po3_profile = [pdata(Po_3_index)];
Ps3_case = mean(pdata(Ps_3_case_index));
Ps3_hub = mean(pdata(Ps_3_hub_index));
Y_HubTip = [Ps3_case; Ps3_hub];
X_HubTip = [1; HubTip];
Y1_HubTip = interp1(X_HubTip,Y_HubTip,P_HubTip);
for i=1:20
PrA(i) = Y1_HubTip(i)/Po3_profile(i);
end
for i=1:20
PrB(i) = ((abs(PrA(i)))^(1/Gam_Air))*sqrt(abs(1-PrA(i)^((Gam_Air-1)/Gam_Air)));
end
To3 = [tdata(To_3_index)];
To3_array = [To3,To3(end)];
X_HubTip =[T_HubTip,HubTip];
Y_HubTip = To3_array;
To3_Map = interp1(X_HubTip,Y_HubTip,P_HubTip);
for i=1:20
int(i) =(Po3_profile(i)/sqrt(To3_Map(i)))*PrB(i)*P_HubTip(i)*d_HubTip(i);
end
Po3 = (sum(Po3_profile.*int))/sum(int); % po3 avg
```

```
To3 = (sum(To3_Map.*int))/sum(int); %to3 avg
Ps3 = (sum(Y1_HubTip.*int))/sum(int); %ps3 avg
%Effic_Avg;
 % Exit whirl data
Omega  = 2*pi*RPM/60;                                    % [rad/s]
ro3      = (Do3*.0254)/2;                                  % Radii
V_th3  = (Cp_air./(Omega*ro3)).*(To3_Map-To1_avg);  % Exit Whirl
Vabs   = sqrt(abs(1-(Y1_HubTip./Po3_profile).^((Gam_Air-
1)/Gam_Air))).*sqrt(2*Cp_air.*To3_Map); % Absolute EXit Velocity
Alp2     = real(asin(V_th3./Vabs));                          % Exit angle
Alp2(Alp2>(pi/2-0.01)) = pi/2-0.01;
Alp2(Alp2<(-pi/2+0.01)) = -pi/2+0.01;
fred     = Vabs.*cos(Alp2);                                    % Junk data for
output
fred     = Alp2;
%Alp2    = zeros(size(Alp2))  % For debugging
Vz3    = Vabs.*cos(Alp2)                                  % Axial velocity

% Mass averaged Properties
rho   = (Po3_profile./(R_air*To3_Map)).*(Y1_HubTip./Po3_profile).^(1/Gam_Air);    %
Density
dA      = 2*pi.*(ro3^2).*P_HubTip.*d_HubTip;                        % Areas
dmA     = rho.*Vz3.*dA;                                                    %
Mass Flow based on axial
dm      = rho.*Vabs.*dA;
% Mass Flow based on absolute
EffMA_Map = ((Po3_profile/Po1_avg).^((Gam_Air-1)/Gam_Air)-1)./((To3_Map/To1_avg)-
1);
```

```
% Values based on absolute velocity
To3_new = sum(To3_Map.*dm)/sum(dm);
Po3_new = sum(Po3_profile.*dm)/sum(dm);
Ps3_new = sum(Y1_HubTip.*dm)/sum(dm);
Eff_Stage = sum(EffMA_Map.*dm)/sum(dm); % Stage suitable
%%%%effstage
Power_MA = Cp_air*sum((To3_Map-To1_avg).*dm);
mdot3 = sum(dm);
mdot3_Rot=sum(dmA);
Area = sum(dA);

% Rotor only values based on axial velocity
To3_Rot    = sum(To3_Map.*dmA)/sum(dmA);
Po3_Rot    = sum(Po3_profile.*dmA)/sum(dmA);
Ps3_Rot    = sum(Y1_HubTip.*dmA)/sum(dmA);
Eff_Rot_Only = sum(EffMA_Map.*dmA)/sum(dmA);      % Efficiency for Rotor only
%%%%effrot
Power_MA_Rot = Cp_air*sum((To3_Map-To1_avg).*dmA); % Power for Rotor only
% Calc_Mass Flow Old
Flow_Noz_Stag = pdata(51);
Flow_Noz_Stat = pdata(52);
Ds = Flow_Noz_Stag-Flow_Noz_Stat;
Term1 = Anoz*Cnoz;
Term2 = sqrt(abs(2*Flow_Noz_Stag*Ds/(R_air*To1_avg)));
Term3 = 1-(3/(4*Gam_Air))*(Ds/Flow_Noz_Stag);
m_dot_old = Term1*Term2*Term3;
% Calc_Mass Flow X
Flow_Noz_Stag = pdata(51);
Flow_Noz_Stat = pdata(52);
```

```
X2_noz = abs(1-(Flow_Noz_Stat/Flow_Noz_Stag)^((Gam_Air-1)/Gam_Air));
Vnoz = sqrt(2*Cp_air*To1_avg*X2_noz);
rhonoz =
(Flow_Noz_Stag/(R_air*To1_avg))*((Flow_Noz_Stat/Flow_Noz_Stag)^(1/Gam_Air));
m_dot = rhonoz*Vnoz*Anoz*Cnoz;
m_dot_ref = m_dot*(sqrt(Theta_ref))/Delta_ref;
m_dot_scale = m_dot_ref*(Scale_RPM);
% calc_Pr
Pr = Po3/Po1_avg;
PrRot = Po3_Rot/Po1_avg;
Ps3_Po1 = Ps3/Po1_avg;
Pr_Scale = 1+(Pr-1)*(Scale_RPM^2);
PrRot_Scale = 1+(PrRot-1)*(Scale_RPM^2);
Ps3_Po1_Scale = 1+(Ps3_Po1-1)*(Scale_RPM^2);
% Calc_eff1
Eff1 = ((To1_avg/(To3-To1_avg))*(Pr^((Gam_Air-1)/Gam_Air)-1))*100;
% Calc_eff2
Vt = sqrt(2*Cp_air*To1_avg);
Binf = (Gam_Air/(Gam_Air-1))*(Ps1_avg/(Vt*(m_dot/Ainf)));
Xinf = sqrt(Binf^2+1)-Binf;
Eff2 = ((To1_avg/(To3-To1_avg))*(2*Binf*Xinf*(Po3/Ps1_avg)^((Gam_Air-1)/Gam_Air)-
1))*100;
% Calc_Power
Power = m_dot*Cp_air*(To3-To1_avg);
Power_REF = Power/(Delta_ref*sqrt(Theta_ref));
Power_MA_REF = Power_MA/(Delta_ref*sqrt(Theta_ref));
Power_MA_Rot_REF = Power_MA_Rot/(Delta_ref*sqrt(Theta_ref));
%% Relative Humidity
To1_avg_C=To1_avg + KToC;
```

```
PVapSat1_Bar = XSteam('psat_T',To1_avg_C);
PVapSat1 = PVapSat1_Bar*BarToPa;
TWetBulb_C=TWetBulb+KToC;
PVapSat2_Bar = XSteam('psat_T',TWetBulb_C);
PVapSat2 = PVapSat2_Bar*BarToPa;
OmegaSat = (R_air/R_wat)*PVapSat2/(P_Atmos-PVapSat2);
hVapWet_kjkg = XSteam('hV_T',TWetBulb_C);
hVapWet=hVapWet_kjkg*1000;
hLiqWet_kjkg = XSteam('hL_T',TWetBulb_C);
hLiqWet=hLiqWet_kjkg*1000;
hVapDry_kjkg = XSteam('hV_T',To1_avg_C);
hVapDry=hVapDry_kjkg*1000;
OmegaRel = (Cp_air*(T_WetBulb-To1_avg)+OmegaSat*(hVapWet-hLiqWet))/(hVapDry-
hLiqWet);
PVap1 = OmegaRel*Po1_avg/((R_air/R_wat)+ OmegaRel);
PVap1_Bar = PVap1*PaToBar;
RelHumid = 100*PVap1/PVapSat1;
CpVap_si = XSteam('Cp_pT',PVap1_Bar,To1_avg_C);
CpVap= CpVap_si*1000;
Cp_air = (Cp_Ref+OmegaRel*CpVap)/(1+OmegaRel);
R_air = (R_Ref+OmegaRel*R_wat)/(1+OmegaRel);
Cv_Air = Cp_air-R_air;
Gam_Air = Cp_air/Cv_Air;
% Update Figures
%Raw Pressure
pformat = "P1 = %f     P2 = %f\nP3 = %f     P4 = %f\nP5 = %f     P6 = %f\nP7 = %f
P8 = %f\nP9 = %f     P10 = %f\nP11 = %f     P12 = %f\nP13 = %f     P14 = %f\nP15 =
%f     P16 = %f\nP17 = %f     P18 = %f\nP19 = %f     P20 = %f\nP21 = %f     P22 =
%f\nP23 = %f     P24 = %f\nP25 = %f     P26 = %f\nP27 = %f     P28 = %f\nP29 = %f
```

```
P30 = %f\nP31 = %f     P32 = %f\nP33 = %f     P34 = %f\nP35 = %f     P36 = %f\nP37
= %f     P38 = %f\nP39 = %f     P40 = %f\nP41 = %f     P42 = %f\nP43 = %f     P44 =
%f\nP45 = %f     P46 = %f\nP47 = %f     P48 = %f\nP49 = %f     P50 = %f\nP51 = %f
P52 = %f\nP53 = %f     P54 = %f\nP55 = %f     P56 = %f\n";
P1=pdata(1);P2=pdata(2);P3=pdata(3);P4=pdata(4);P5=pdata(5);P6=pdata(6);
P7=pdata(7);P8=pdata(8);P9=pdata(9);P10=pdata(10);P11=pdata(11);P12=pdata(12);
P13=pdata(13);P14=pdata(14);P15=pdata(15);P16=pdata(16);P17=pdata(17);
P18=pdata(18);P19=pdata(19);P20=pdata(20);P21=pdata(21);P22=pdata(22);
P23=pdata(23);P23=pdata(23);P24=pdata(24);P25=pdata(25);P26=pdata(26);
P27=pdata(27);P28=pdata(28);P29=pdata(29);P30=pdata(30);P31=pdata(31);
P32=pdata(32);P33=pdata(33);P34=pdata(34);P35=pdata(35);P36=pdata(36);
P37=pdata(37);P38=pdata(38);P39=pdata(39);P40=pdata(40);P41=pdata(41);
P42=pdata(42);P43=pdata(43);P44=pdata(44);P45=pdata(45);P46=pdata(46);
P47=pdata(47);P48=pdata(48);P49=pdata(49);P50=pdata(50);P51=pdata(51);
P52=pdata(52);P53=pdata(53);P54=pdata(54);P55=pdata(55);P56=pdata(56);
pstr=sprintf(pformat,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,P15,P16,P17,P18
,P19,P20,P21,P22,P23,P24,P25,P26,P27,P28,P29,P30,P31,P32,P33,P34,P35,P36,P37,P38,P3
9,P40,P41,P42,P43,P44,P45,P46,P47,P48,P49,P50,P51,P52,P53,P54,P55,P56);
pnumbers = findobj('Tag','rawpressure');
%Raw Temperature
tformat = "T1 = %f\nT2 = %f\nT3 = %f\nT4 = %f\nT5 = %f\nT6 = %f\nT7 = %f\nT8 =
%f\nT9 = %f\nT10 = %f\nT11 = %f\nT12 = %f\nT13 = %f\nT14 = %f\nT15 = %f\nT16 =
%f\nT17 = %f\nT18 = %f\nT19 = %f\nT20 = %f\nT21 = %f\nT22 = %f\nT23 = %f\nT24 =
%f\nT25 = %f\n";
T1=tdata(1);T2=tdata(2);T3=tdata(3);T4=tdata(4);T5=tdata(5);T6=tdata(6);
T7=tdata(7);T8=tdata(8);T9=tdata(9);T10=tdata(10);T11=tdata(11);T12=tdata(12);
T13=tdata(13);T14=tdata(14);T15=tdata(15);T16=tdata(16);T17=tdata(17);
T18=tdata(18);T19=tdata(19);T20=tdata(20);T21=tdata(21);T22=tdata(22);
T23=tdata(23);T24=tdata(24);T25=tdata(25);
```

```
tstr=sprintf(tformat,T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15,T16,T17,T18
,T19,T20,T21,T22,T23,T24,T25);
tread = findobj('Tag','rawtemp');
%RPM
 rpmtxt=findobj('Tag','rpmvalue');
%
 %All Update
 pnumbers.String=pstr;
 tread.String=tstr;
 rpmtxt.String=num2str(RPM);
 refrpm.String=num2str(RPM_ref);
 pops.String=num2str(d_P1);
 axial.String=num2str(U_inf);
 machin.String=num2str(MachinRel);
 effstage.String=num2str(Eff_Stage);
 effrot.String=num2str(Eff_Rot_Only);
 mfraw.String=num2str(m_dot);
 mfscale.String=num2str(m_dot_scale);
 prraw.String=num2str(Pr);
 prscale.String=num2str(Pr_Scale);
 rhumid.String=num2str(RelHumid);
clearpoints(pline);
addpoints(pline,P_HubTip,Po3_profile);
drawnow
clearpoints(tline);
addpoints(tline,X_HubTip,To3_array);
drawnow
% Save Function
save('testvariables')
```

```
if save_trigger == 1
date = datestr(now, 'dd-mmm-yyyy');
    Filename = sprintf('%d_pc_Spreadsheet_%s.txt',Spdpct,date);
if firstsave == 0
    hdr={'Run no','Eff Classic Stage','Eff Rot Mass Avg','Eff Stage Mass Avg','RPM
Scale','m_dot Scale','PR Stage Scale','PR Rot
Scale','Ps3/Po1_Stage_Scale','m_dot_REF[kg/s]','RPM_REF','Blank','Pow_Stage_ref[W]'
,'Pow_Rot_ref[W]','Pow_Stage[W]','Pow_Rot[W]','Theta_ref','Delta_ref','Gam','Cp','R
','OmegaRel','OmegaSat','Humidity [%]','PrStage','PrRot','Ps3/Po1
Stage','Blank','RPM','m_dot_X','Blank','Blank','To1','Blank','To3_Stage','(Po1-
Ps1)','Ps_1','Po1','Ps3_Stage','Po3_Stage','To3_Rot','Ps3_Rot','Po3_Rot','P1
[Pa]','P2 [Pa]','P3 [Pa]','P4 [Pa]','P5 [Pa]','P6 [Pa]','P7 [Pa]','P8 [Pa]','P9
[Pa]','P10 [Pa]','P11 [Pa]','P12 [Pa]','P13 [Pa]','P14 [Pa]','P15 [Pa]','P16
[Pa]','P17 [Pa]','P18 [Pa]','P19 [Pa]','P20 [Pa]','P21 [Pa]','P22 [Pa]','P23
[Pa]','P24 [Pa]','P25 [Pa]','P26 [Pa]','P27 [Pa]','P28 [Pa]','P29 [Pa]','P30
[Pa]','P31 [Pa]','P32 [Pa]','P33 [Pa]','P34 [Pa]','P35 [Pa]','P36 [Pa]','P37
[Pa]','P38 [Pa]','P39 [Pa]','P40 [Pa]','P41 [Pa]','P42 [Pa]','P43 [Pa]','P44
[Pa]','P45 [Pa]','P46 [Pa]','P47 [Pa]','P48 [Pa]','P50 [Pa]','P51
[Pa]','To_inf5','To_inf6','Tcase1down','Tcase2down','Tcase3down','Tcase1up','Tcase2
up','Tcase3up','To_11[K]','To_12[K]','To_13[K]','To_14[K]','To_15[K]','To_16[K]','T
o_17[K]','To_18[K]','To_19[K]','To_20[K]','To_21[K]','To_22[K]','To_23[K]','To_24[K
]','To_25[K]','To_26[K]','To_27[K]','To_28[K]','To_29[K]','To_30[K]','T_Raw_0[K]','
T_Raw_1[K]','T_Raw_2[K]','T_Raw_3[K]','T_Raw_4[K]','T_Raw_5[K]','T_Raw_6[K]','T_Raw
_7[K]','T_Raw_8[K]','T_Raw_9[K]','T_Raw_10[K]','T_Raw_11[K]','T_Raw_12[K]','T_Raw_1
3[K]','T_Raw_14[K]','T_Raw_15[K]','Eff_11[K]','Eff_12[K]','Eff_13[K]','Eff_14[K]','
Eff_15[K]','Eff_16[K]','Eff_17[K]','Eff_18[K]','Eff_19[K]','Eff_20[K]','Eff_21[K]',
'Eff_22[K]','Eff_23[K]','Eff_24[K]','Eff_25[K]','Eff_26[K]','Eff_27[K]','Eff_28[K]'
,'Eff_29[K]','Eff_30[K]','Vth3_11[m/s]','Vth3_12[m/s]','Vth3_13[m/s]','Vth3_14[m/s]
','Vth3_15[m/s]','Vth3_16[m/s]','Vth3_17[m/s]','Vth3_18[m/s]','Vth3_19[m/s]','Vth3_
```

```
20[m/s]','Vth3_21[m/s]','Vth3_22[m/s]','Vth3_23[m/s]','Vth3_24[m/s]','Vth3_25[m/s]'
,'Vth3_26[m/s]','Vth3_27[m/s]','Vth3_28[m/s]','Vth3_29[m/s]','Vth3_30[m/s]','Vz3_11
[m/s]','Vz3_12[m/s]','Vz3_13[m/s]','Vz3_14[m/s]','Vz3_15[m/s]','Vz3_16[m/s]','Vz3_1
7[m/s]','Vz3_18[m/s]','Vz3_19[m/s]','Vz3_20[m/s]','Vz3_21[m/s]','Vz3_22[m/s]','Vz3_
23[m/s]','Vz3_24[m/s]','Vz3_25[m/s]','Vz3_26[m/s]','Vz3_27[m/s]','Vz3_28[m/s]','Vz3
_29[m/s]','Vz3_30[m/s]','m_dot_old','Power_Ref_old','Power_old','mdot_dm_Stage','md
ot_dm_Rotor'};
    txt=sprintf('%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t
%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t
%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t%s\t %s\t %s\t %s\t %s\t %s\t
%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t%s\t %s\t %s\t
%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t
%s\t%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t
%s\t %s\t %s\t %s\t%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t
%s\t %s\t %s\t %s\t %s\t %s\t %s\t%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t
%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t%s\t %s\t %s\t %s\t %s\t %s\t %s\t
%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t%s\t %s\t %s\t %s\t
%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t%s\t
%s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t %s\t
%s\t %s\t %s\t %s\t %s\r\t',hdr{:});
    txt(end)='';
    vars=[runnum Eff1 Eff_Rot_Only Eff_Stage Scale_RPM m_dot_scale Pr_Scale
PrRot_Scale Ps3_Po1_Scale m_dot_ref RPM_ref 0 Power_MA_REF Power_MA_Rot_REF
Power_MA Power_MA_Rot Theta_ref Delta_ref Gam_Air Cp_air R_air OmegaRel OmegaSat
RelHumid Pr PrRot Ps3_Po1 0 RPM m_dot tdata(1) tdata(2) To1_avg 0 To3 d_P1 Po1_avg
Po1_avg Ps3 Po3 To3_Rot Ps3_Rot Po3_Rot pdata(1) pdata(2) pdata(3) pdata(4)
pdata(5) pdata(6) pdata(7) pdata(8) pdata(9) pdata(10) pdata(11) pdata(12)
pdata(13) pdata(14) pdata(15) pdata(16) pdata(17) pdata(18) pdata(19) pdata(20)
pdata(21) pdata(22) pdata(23) pdata(24) pdata(25) pdata(26) pdata(27) pdata(28)
```

```
pdata(29) pdata(30) pdata(31) pdata(32) pdata(33) pdata(34) pdata(35) pdata(36)
pdata(37) pdata(38) pdata(39) pdata(40) pdata(41) pdata(42) pdata(43) pdata(44)
pdata(45) pdata(46) pdata(47) pdata(48) pdata(50) pdata(51) tdata(1) tdata(2)
tdata(16) tdata(17) tdata(18) tdata(19) tdata(20) tdata(21) To3_Map(1) To3_Map(2)
To3_Map(3) To3_Map(4) To3_Map(5) To3_Map(6) To3_Map(7) To3_Map(8) To3_Map(9)
To3_Map(10) To3_Map(11) To3_Map(12) To3_Map(13) To3_Map(14) To3_Map(15) To3_Map(16)
To3_Map(17) To3_Map(18) To3_Map(19) To3_Map(20) tdata(1) tdata(2) tdata(3) tdata(4)
tdata(5) tdata(6) tdata(7) tdata(8) tdata(9) tdata(10) tdata(11) tdata(12)
tdata(13) tdata(14) tdata(15) tdata(16) EffMA_Map(1)  EffMA_Map(2) EffMA_Map(3)
EffMA_Map(4) EffMA_Map(5) EffMA_Map(6) EffMA_Map(7) EffMA_Map(8) EffMA_Map(9)
EffMA_Map(10) EffMA_Map(11) EffMA_Map(12) EffMA_Map(13) EffMA_Map(14) EffMA_Map(15)
EffMA_Map(16) EffMA_Map(17) EffMA_Map(18) EffMA_Map(19) EffMA_Map(20) V_th3(1)
V_th3(2) V_th3(3) V_th3(4) V_th3(5) V_th3(6) V_th3(7) V_th3(8) V_th3(9) V_th3(10)
V_th3(11) V_th3(12) V_th3(13) V_th3(14) V_th3(15) V_th3(16) V_th3(17) V_th3(18)
V_th3(19) V_th3(20) Vz3(1) Vz3(2) Vz3(3) Vz3(4) Vz3(5) Vz3(6) Vz3(7) Vz3(8) Vz3(9)
Vz3(10) Vz3(11) Vz3(12) Vz3(13) Vz3(14) Vz3(15) Vz3(16) Vz3(17) Vz3(18) Vz3(19)
Vz3(20) m_dot_old Power Power_REF mdot3 mdot3_Rot]; % Power Power_REF runnum
    dlmwrite(Filename,txt,'');
    dlmwrite(Filename,vars,'delimiter','\t','-append');
runnum = runnum +1;
run_count = num2str(runnum);
run=findobj('Tag','runnumber');
run.String = run_count;
firstsave = 1;
else
    vars=[runnum Eff1 Eff_Rot_Only Eff_Stage Scale_RPM m_dot_scale Pr_Scale
PrRot_Scale Ps3_Po1_Scale m_dot_ref RPM_ref 0 Power_MA_REF Power_MA_Rot_REF
Power_MA Power_MA_Rot Theta_ref Delta_ref Gam_Air Cp_air R_air OmegaRel OmegaSat
RelHumid Pr PrRot Ps3_Po1 0 RPM m_dot tdata(1) tdata(2) To1_avg 0 To3 d_P1 Po1_avg
```

```
Po1_avg Ps3 Po3 To3_Rot Ps3_Rot Po3_Rot pdata(1) pdata(2) pdata(3) pdata(4)
pdata(5) pdata(6) pdata(7) pdata(8) pdata(9) pdata(10) pdata(11) pdata(12)
pdata(13) pdata(14) pdata(15) pdata(16) pdata(17) pdata(18) pdata(19) pdata(20)
pdata(21) pdata(22) pdata(23) pdata(24) pdata(25) pdata(26) pdata(27) pdata(28)
pdata(29) pdata(30) pdata(31) pdata(32) pdata(33) pdata(34) pdata(35) pdata(36)
pdata(37) pdata(38) pdata(39) pdata(40) pdata(41) pdata(42) pdata(43) pdata(44)
pdata(45) pdata(46) pdata(47) pdata(48) pdata(50) pdata(51) tdata(1) tdata(2)
tdata(16) tdata(17) tdata(18) tdata(19) tdata(20) tdata(21) To3_Map(1) To3_Map(2)
To3_Map(3) To3_Map(4) To3_Map(5) To3_Map(6) To3_Map(7) To3_Map(8) To3_Map(9)
To3_Map(10) To3_Map(11) To3_Map(12) To3_Map(13) To3_Map(14) To3_Map(15) To3_Map(16)
To3_Map(17) To3_Map(18) To3_Map(19) To3_Map(20) tdata(1) tdata(2) tdata(3) tdata(4)
tdata(5) tdata(6) tdata(7) tdata(8) tdata(9) tdata(10) tdata(11) tdata(12)
tdata(13) tdata(14) tdata(15) tdata(16) EffMA_Map(1)  EffMA_Map(2) EffMA_Map(3)
EffMA_Map(4) EffMA_Map(5) EffMA_Map(6) EffMA_Map(7) EffMA_Map(8) EffMA_Map(9)
EffMA_Map(10) EffMA_Map(11) EffMA_Map(12) EffMA_Map(13) EffMA_Map(14) EffMA_Map(15)
EffMA_Map(16) EffMA_Map(17) EffMA_Map(18) EffMA_Map(19) EffMA_Map(20) V_th3(1)
V_th3(2) V_th3(3) V_th3(4) V_th3(5) V_th3(6) V_th3(7) V_th3(8) V_th3(9) V_th3(10)
V_th3(11) V_th3(12) V_th3(13) V_th3(14) V_th3(15) V_th3(16) V_th3(17) V_th3(18)
V_th3(19) V_th3(20) Vz3(1) Vz3(2) Vz3(3) Vz3(4) Vz3(5) Vz3(6) Vz3(7) Vz3(8) Vz3(9)
Vz3(10) Vz3(11) Vz3(12) Vz3(13) Vz3(14) Vz3(15) Vz3(16) Vz3(17) Vz3(18) Vz3(19)
Vz3(20) m_dot_old Power Power_REF mdot3 mdot3_Rot];
    dlmwrite(Filename,vars,'delimiter','\t','-append');
    runnum = runnum +1;
run_count = num2str(runnum);
run=findobj('Tag','runnumber');
run.String = run_count;
end
save_trigger = 0;
end
```

```
end
end
```

6. Save_Run.m

This script triggers the program to save the workspace variables in a text file.

```
%% Save Run Function
function Save_Run(Save_btn,tdata,pdata)
global save_trigger
save_trigger=1;
end
```

7. Run_Stopmulti.m

This script stops the data acquisition process.

```
function Run_Stopmulti(p_stop_btn,~)
    % Stop Pressure Brick Run
    global pressurestat ni_set
    pressurestat = 2;
    stop(ni_set)
h=findobj('Tag','atmp');
h.Enable = 'on';
h=findobj('Tag','spd');
h.Enable = 'on';
h=findobj('Tag','design');
```

```
h.Enable = 'on';
h=findobj('Tag','cal');
h.Enable = 'on';
h=findobj('Tag','wetbulb');
h.Enable = 'on';
h=findobj('Tag','applybtn');
h.Enable = 'on';
h=findobj('Tag','savebtn');
h.Enable = 'off';
h=findobj('Tag','resetbtn');
h.Enable = 'on';
end
```

8. Reset_num.m

This script resets the run number.

```
%% Reset run number function
function Reset_num(Resetnum_btn,~)
global runnum run_count run_num;
runnum = 1;
run_count = sprintf('%d',runnum);
run = findobj('Tag','runnumber');
run.String = run_count;
end
```

9. Pulsewidth_main.m

This script reads the pulse width from the NI 9402 counter-totalizer module and converts it to RPM.

```
%% Addlistener function for counter-totalizer module
function [RPM] = Pulsewidth_main(src,event)
% Reads pulsewidth data from module and converts to RPM
global RPMr
pw = event.Data(1);
RPM = [60/pw];
% rpmtxt=findobj('Tag','rpmvalue');
% rpmtxt.String=sprintf('%d',RPM);
RPMr=RPM;
end
```

10. Temp_readings_main.m

This script reads the temperature from the NI 9214 thermocouple modules.

```
%% Addlistener function for thermocouple modules
function [tdata] = Temp_readings_main(src,event)
% Reads temperature data from modules
global tdatar

tdata = [1:1:25];
```

```
for i=1:25;
    s=i+1;
    tdata(i) = event.Data(s);
end
tdatar=tdata;
end
```

11. XSteam.m

This script provides the water and steam properties tables. It was created by Magnus Holmgren and www.x-eng.com and can be found on MATLAB's file exchange [16].

APPENDIX F. DAS OUTPUT FILES

The following is the output text files from both the HPVEE program and the MATLAB program compared to each other.

Run no	Eff Classic Stage	Eff Rot Mass Avg	Eff Stage Mass Avg	RPM Scale	m_dot Scale
HPVEE					
1	24.04673394	0.240467735	0.240467735	1.015405422	0.39577499
2	24.03692335	0.24036961	0.24036961	1.015298446	0.277898303
3	24.05579545	0.240558334	0.240558334	1.015496509	0.277949276
MATLAB					
1	23.697	0.23697	0.23697	1.0209	0.052365
2	23.689	0.23689	0.23689	1.0206	0.052349
3	23.68	0.2368	0.2368	1.0201	0.052325

PR Stage Scale	PR Rot Scale	Ps3/Po1_Stage_Scale	m_dot_REF[kg/s]	RPM_REF	Blank
HPVEE					
1.509911123	1.509911123	1.000124396	0.389770413	23635.87932	0
1.509817765	1.509817765	1.000107392	0.273710951	23638.36968	0
1.509996529	1.509996529	1.000095934	0.273707761	23633.75924	0
MATLAB					
1.5153	1.5153	0.99999	0.051295	23509	0
1.5149	1.5149	0.99999	0.051295	23517	0
1.5145	1.5145	0.99999	0.051294	23528	0

Pow_Stage_ref[W]	Pow_Rot_ref[W]	Pow_Stage[W]	Pow_Rot[W]	Theta_ref	Delta_ref
HPVEE					
1448592.919	14485.68776	1467671.607	14676.47146	1.031987298	0.997344871
1449143.869	14491.19717	1468126.442	14681.01973	1.031842066	0.997344833
1448140.007	14481.15872	1467330.694	14673.06238	1.032131104	0.99735557
MATLAB					
1.46E+06	14648	1.48E+06	14776	1.021	0.9983
1.47E+06	14652	1.48E+06	14779	1.0208	0.9983
1.47E+06	14657	1.48E+06	14782	1.0206	0.9983

Gam	Cp	R	OmegaRel	OmegaSat	Humidity [%]
HPVEE					
1.399195713	1009.395121	287.9841621	0.005671887	0.008690992	31.65001169
1.399193227	1009.409874	287.9870895	0.005688853	0.008690984	31.82347845
1.399198046	1009.38127	287.9814134	0.005655955	0.008691072	31.48770606
MATLAB					
1.399	1010.6	288.22	0.0070487	0.008698	48.125
1.399	1010.6	288.23	0.0070712	0.0086981	48.438
1.399	1010.7	288.23	0.0071067	0.0086983	48.939

PrStage	PrRot	Ps3/Po1 Stage	Blank	RPM	m_dot_X
HPVEE					
1.494556063	1.494556063	1.00012065	0	24010.92769	0.382663511
1.494569719	1.494569719	1.00010418	0	24011.76779	0.268739127
1.494550165	1.494550165	1.000093029	0	24010.4467	0.268701257
MATLAB					
1.4944	1.4944	1	0	23755	0.050678
1.4944	1.4944	0.99999	0	23760	0.050682
1.4944	1.4944	0.99999	0	23768	0.050689

Blank	Blank	To1	Blank	To3_Stage	(Po1-Ps1)
HPVEE					
0	0	296.5381836	0	446.3360213	-7.618632436
0	0	296.4966797	0	446.3383535	-6.359943867
0	0	296.5772461	0	446.3361847	-2.535009146
MATLAB					
444.07	443.98	293.19	0	443.39	1.9067
444.03	443.9	293.14	0	443.36	1.9067
444.06	443.82	293.05	0	443.28	1.0323

Ps_1	Po1	Ps3_Stage	Po3_Stage	To3_Rot	Ps3_Rot
HPVEE					
101121.6933	101114.0747	101126.2741	151120.6533	446.3360213	101126.2741
101120.4142	101114.0543	101124.5883	151122.0037	446.3383535	101124.5883
101117.8575	101115.3225	101124.7291	151121.9219	446.3361847	101124.7291
MATLAB					
1.01E+05	1.01E+05	1.01E+05	1.51E+05	443.39	1.01E+05
1.01E+05	1.01E+05	1.01E+05	1.51E+05	443.36	1.01E+05
1.01E+05	1.01E+05	1.01E+05	1.51E+05	443.28	1.01E+05

Po3_Rot	P1 [Pa]	P2 [Pa]	P3 [Pa]	P4 [Pa]	P5 [Pa]
HPVEE					
151120.6533	101122.9474	101120.4156	101112.6624	101112.6848	101120.4156
151122.0037	101125.5099	101120.4156	101115.2468	101110.0996	101115.3021
151121.9219	101130.6043	101117.8304	101115.2468	101104.9539	101115.3021
MATLAB					
1.51E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05
1.51E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05
1.51E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05

P6 [Pa]	P7 [Pa]	P8 [Pa]	P9 [Pa]	P10 [Pa]	P11 [Pa]
HPVEE					
101107.7337	101122.971	101120.4156	101125.4936	101125.5562	151135.6448
101112.8065	101122.971	101117.8575	101125.4936	101130.6967	151140.7212
101115.3429	101120.4156	101115.2994	101128.0326	101122.9859	151135.6448
MATLAB					
1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.51E+05
1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.51E+05
1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.51E+05

P12 [Pa]	P13 [Pa]	P14 [Pa]	P15 [Pa]	P16 [Pa]	P17 [Pa]
HPVEE					
151132.5251	151137.369	151134.9249	151133.1596	151132.8518	151120.4156
151132.5251	151137.369	151134.9249	151133.1596	151127.8773	151120.4156
151130.1032	151139.7909	151132.5067	151130.6108	151140.3135	151130.9443
MATLAB					
1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05
1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05
1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05

P18 [Pa]	P19 [Pa]	P20 [Pa]	P21 [Pa]	P22 [Pa]	P23 [Pa]
HPVEE					
151110.8278	151120.4156	151120.4156	151111.3454	151111.5831	151111.3324
151110.8278	151120.4156	151120.4156	151111.3454	151120.4156	151111.3324
151110.8278	151120.4156	151110.9325	151120.4156	151111.5831	151111.3324
MATLAB					
1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05
1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05
1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05

P24 [Pa]	P25 [Pa]	P26 [Pa]	P27 [Pa]	P28 [Pa]	P29 [Pa]
HPVEE					
151120.4156	151111.6143	151111.2364	151111.2231	151111.5885	151111.4166
151120.4156	151111.6143	151120.4156	151111.2231	151120.4156	151102.4176
151120.4156	151111.6143	151120.4156	151102.0306	151111.5885	151111.4166
MATLAB					
1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05
1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05
1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05	1.51E+05

P30 [Pa]	P31 [Pa]	P32 [Pa]	P33 [Pa]	P34 [Pa]	P35 [Pa]
HPVEE					
151111.3165	101111.1965	101111.7605	101120.4156	101120.4156	101130.5712
151120.4156	101111.1965	101111.7605	101120.4156	101120.4156	101120.4156
151120.4156	101111.1965	101120.4156	101120.4156	101130.594	101130.5712
MATLAB					
1.51E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05
1.51E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05
1.51E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05

P36 [Pa]	P37 [Pa]	P38 [Pa]	P39 [Pa]	P40 [Pa]	P41 [Pa]
HPVEE					
101130.5098	101120.4156	101130.6005	101120.4156	101120.4156	101120.4156
101120.4156	101120.4156	101120.4156	101110.2017	101120.4156	101120.4156
101120.4156	101120.4156	101130.6005	101120.4156	101110.1191	101120.4156
MATLAB					
1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05
1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05
1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05

P42 [Pa]	P43 [Pa]	P44 [Pa]	P45 [Pa]	P46 [Pa]	P47 [Pa]
HPVEE					
101120.4156	101120.4156	101130.4613	101120.4156	101120.4156	101120.4156
101120.4156	101120.4156	101130.4613	101120.4156	101120.4156	101120.4156
101120.4156	101120.4156	101130.4613	101130.6427	101110.2545	101120.4156
MATLAB					
1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05
1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05
1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05	1.01E+05

P48 [Pa]	P50 [Pa]	P51 [Pa]	To_inf5	To_inf6	Tcase1down
HPVEE					
101120.4156	101120.4156	101131.3628	9.91E+37	296.1890625	9.91E+37
101120.4156	101125.8767	101131.3628	9.91E+37	296.2134766	9.91E+37
101110.877	101125.8767	101131.3628	9.91E+37	296.2037109	9.91E+37
MATLAB					
1.01E+05	1.01E+05	1.01E+05	444.07	443.98	293.05
1.01E+05	1.01E+05	1.01E+05	444.03	443.9	293
1.01E+05	1.01E+05	1.01E+05	444.06	443.82	292.99

Tcase2down	Tcase3down	Tcase1up	Tcase2up	Tcase3up	To_11[K]
HPVEE					
9.91E+37	9.91E+37	9.91E+37	9.91E+37	9.91E+37	446.6513765
9.91E+37	9.91E+37	9.91E+37	9.91E+37	9.91E+37	446.6493591
9.91E+37	9.91E+37	9.91E+37	9.91E+37	9.91E+37	446.6553892
MATLAB					
293.06	293.39	293.09	293.21	292.65	443.9
293.01	293.33	293.04	293.21	292.65	443.86
292.94	293.27	292.95	293.19	292.63	443.85

To_12[K]	To_13[K]	To_14[K]	To_15[K]	To_16[K]	To_17[K]
HPVEE					
446.662634	446.6809559	446.6639798	446.5955725	446.3970116	446.2416871
446.6582214	446.6721668	446.6551908	446.587422	446.3915577	446.2382452
446.6597006	446.6654379	446.6492702	446.5836035	446.3904222	446.2314761
MATLAB					
443.77	443.51	443.38	443.2	443.09	443.16
443.72	443.46	443.34	443.19	443.11	443.14
443.68	443.36	443.25	443.09	442.99	443.04

To_18[K]	To_19[K]	To_20[K]	To_21[K]	To_22[K]	To_23[K]
HPVEE					
446.1051405	446.1266604	446.2127397	446.2859072	446.3752	446.3233287
446.1067555	446.1293124	446.2195402	446.2962337	446.3897648	446.3266548
446.0953511	446.1188155	446.2126731	446.2924521	446.3898162	446.3336224
MATLAB					
443.29	443.3	443.37	443.43	443.5	443.18
443.24	443.26	443.34	443.4	443.49	443.18
443.17	443.19	443.28	443.35	443.45	443.11

To_24[K]	To_25[K]	To_26[K]	To_27[K]	To_28[K]	To_29[K]
HPVEE					
446.2761731	446.1948234	446.1676214	446.1426249	446.115356	446.103994
446.269282	446.2163165	446.1867354	446.15556	446.1215505	446.1073799
446.2825371	446.2086605	446.1810653	446.1546645	446.1258636	446.1138633
MATLAB					
442.89	443.88	443.81	443.61	443.4	443.31
442.89	443.8	443.73	443.54	443.34	443.25
442.79	443.73	443.67	443.49	443.29	443.21

To_30[K]	T_Raw_0[K]	T_Raw_1[K]	T_Raw_2[K]	T_Raw_3[K]	T_Raw_4[K]
HPVEE					
446.1021484	9.91E+37	296.6373047	296.1890625	296.3755859	296.1988281
446.1050781	9.91E+37	296.6382813	296.2134766	296.3902344	296.1880859
446.1119141	9.91E+37	296.65	296.2037109	296.3902344	296.1861328
MATLAB					
443.29	444.07	443.98	443.5	293.16	443.24
443.24	444.03	443.9	443.49	293.11	443.19
443.19	444.06	443.82	443.45	293.01	443.11

T_Raw_5[K]	T_Raw_6[K]	T_Raw_7[K]	T_Raw_8[K]	T_Raw_9[K]	T_Raw_10[K]
HPVEE					
296.0513672	296.4283203	296.6421875	296.6832031	296.2583984	296.1021484
296.0503906	296.4224609	296.6333984	296.6744141	296.2476563	296.1050781
296.0367188	296.4224609	296.6285156	296.6675781	296.2632813	296.1119141
MATLAB					
443.08	443.23	443.52	442.78	443.29	292.91
443.1	443.22	443.47	442.79	443.24	292.87
442.97	443.13	443.37	442.67	443.19	292.84

T_Raw_11[K]	T_Raw_12[K]	T_Raw_13[K]	T_Raw_14[K]	T_Raw_15[K]	Eff_11[K]
HPVEE					
9.91E+37	9.91E+37	296.6578125	9.91E+37	296.4185547	0.240024869
9.91E+37	9.91E+37	296.665625	9.91E+37	296.3277344	0.239949616
9.91E+37	9.91E+37	296.6451172	9.91E+37	296.509375	0.240103721
MATLAB					
293.18	293.23	293.06	293.15	293.05	0.23617
293.11	293.19	292.99	293.08	293	0.23611
293.02	293.13	292.9	292.97	292.99	0.23592

Eff_12[K]	Eff_13[K]	Eff_14[K]	Eff_15[K]	Eff_16[K]	Eff_17[K]
HPVEE					
0.239993824	0.239984792	0.240001708	0.240103733	0.240420578	0.240617871
0.239901193	0.239899162	0.239916067	0.24001701	0.240308514	0.240523272
0.240073639	0.24010499	0.240100381	0.240197552	0.240547947	0.240764121
MATLAB					
0.23637	0.23677	0.23698	0.23726	0.23744	0.23733
0.23632	0.23672	0.23691	0.23714	0.23728	0.23722
0.23617	0.23668	0.23684	0.23709	0.23726	0.23718

Eff_18[K]	Eff_19[K]	Eff_20[K]	Eff_21[K]	Eff_22[K]	Eff_23[K]
HPVEE					
0.240797293	0.240802895	0.240664407	0.240508787	0.240366456	0.240448646
0.240694433	0.240698374	0.240553321	0.240392153	0.240280139	0.24034329
0.240898837	0.240901298	0.240710399	0.240621905	0.240428502	0.240517668
MATLAB					
0.23715	0.23711	0.237	0.23692	0.2368	0.2373
0.23708	0.23704	0.23691	0.23682	0.23669	0.23717
0.23699	0.23695	0.23681	0.23669	0.23653	0.23707

Eff_24[K]	Eff_25[K]	Eff_26[K]	Eff_27[K]	Eff_28[K]	Eff_29[K]
HPVEE					
0.240562454	0.240656293	0.240698458	0.240738619	0.240784041	0.240801611
0.24047343	0.240521597	0.240606039	0.2406176	0.24071086	0.240658137
0.240637842	0.240719715	0.240801059	0.240766378	0.24085288	0.240871487
MATLAB					
0.23775	0.23619	0.2363	0.23663	0.23695	0.23709
0.23761	0.23619	0.2363	0.2366	0.23691	0.23704
0.23756	0.2361	0.23619	0.23649	0.23679	0.23692

Eff_30[K]	Vth3_11[m/s]	Vth3_12[m/s]	Vth3_13[m/s]	Vth3_14[m/s]	Vth3_15[m/s]
HPVEE					
0.240804163	419.6174765	419.648945	419.7001608	419.652707	419.4614853
0.240737363	419.7126065	419.7373786	419.7763595	419.7289074	419.5394775
0.240912416	419.5334733	419.5455254	419.5615636	419.516368	419.3328009
MATLAB					
0.23711	426.32	425.94	425.22	424.85	424.34
0.23707	426.27	425.88	425.14	424.81	424.38
0.23694	426.36	425.87	424.97	424.66	424.22

Vth3_16[m/s]	Vth3_17[m/s]	Vth3_18[m/s]	Vth3_19[m/s]	Vth3_20[m/s]	Vth3_21[m/s]
HPVEE					
418.9064398	418.472255	418.0905609	418.1507162	418.3913373	418.5958652
418.9919901	418.563445	418.1958997	418.2589517	418.5111598	418.7255368
418.7927755	418.3484524	417.967924	418.0335172	418.2958899	418.5189067
MATLAB					
424.03	424.23	424.59	424.63	424.83	424.98
424.14	424.25	424.53	424.58	424.8	424.98
423.92	424.07	424.44	424.49	424.75	424.96

Vth3_22[m/s]	Vth3_23[m/s]	Vth3_24[m/s]	Vth3_25[m/s]	Vth3_26[m/s]	Vth3_27[m/s]
HPVEE					
418.845469	418.7004712	418.5686551	418.3412552	418.2652163	418.1953426
418.9869783	418.8105708	418.6502003	418.5021489	418.4194628	418.3323201
418.7910816	418.6339955	418.4911899	418.2846728	418.2075323	418.1337306
MATLAB					
425.18	424.29	423.47	426.27	426.07	425.51
425.22	424.34	423.54	426.11	425.91	425.38
425.23	424.26	423.37	426.02	425.85	425.33

Vth3_28[m/s]	Vth3_29[m/s]	Vth3_30[m/s]	Vz3_11[m/s]	Vz3_12[m/s]	Vz3_13[m/s]
HPVEE					
418.1191167	418.0873559	418.082197	3.125083658	3.125046055	3.12522517
418.2372554	418.197645	418.1912111	3.125113806	3.124951521	3.125130137
418.0532197	418.0196735	418.0142246	3.125116174	3.124997959	3.125255998
MATLAB					
424.89	424.64	424.6	3.1165	3.116	3.115
424.8	424.55	424.51	3.1164	3.1159	3.1149
424.77	424.54	424.5	3.1164	3.1157	3.1146

Vz3_14[m/s]	Vz3_15[m/s]	Vz3_16[m/s]	Vz3_17[m/s]	Vz3_18[m/s]	Vz3_19[m/s]
HPVEE					
3.125103673	3.124817351	3.124112379	3.123263847	3.122547532	3.122854845
3.125024319	3.124763751	3.123963118	3.123259815	3.122593417	3.12290871
3.125025428	3.124753972	3.124316471	3.123535967	3.122577107	3.122892786
MATLAB					
3.1146	3.1139	3.1137	3.1138	3.1144	3.1144
3.1145	3.114	3.1137	3.1138	3.1143	3.1142
3.1142	3.1136	3.1133	3.1135	3.114	3.1141

Vz3_20[m/s]	Vz3_21[m/s]	Vz3_22[m/s]	Vz3_23[m/s]	Vz3_24[m/s]	Vz3_25[m/s]
HPVEE					
3.123153039	3.123186242	3.123501223	3.123310274	3.12336276	3.122861322
3.123238842	3.123299197	3.123861786	3.123436188	3.123470322	3.123085631
3.122994229	3.123506374	3.123635936	3.123436746	3.123481734	3.123012703
MATLAB					
3.1146	3.1148	3.1151	3.114	3.1129	3.1164
3.1145	3.1148	3.1151	3.1139	3.1129	3.1161
3.1144	3.1146	3.1149	3.1137	3.1126	3.1159

Vz3_26[m/s]	Vz3_27[m/s]	Vz3_28[m/s]	Vz3_29[m/s]	Vz3_30[m/s]	m_dot_old
HPVEE					
3.122753887	3.122662716	3.122572476	3.122527005	3.122517504	0.017299738
3.123210133	3.122893645	3.123014987	3.122535631	3.122967374	0.262398196
3.123133004	3.122597745	3.122732852	3.122688275	3.122900552	0.370995448
MATLAB					
3.1161	3.1155	3.1146	3.1143	3.1142	0.050678
3.1159	3.1153	3.1145	3.1142	3.1141	0.050682
3.1157	3.1151	3.1144	3.1141	3.114	0.050689

Power_Ref_old	Power_old	mdot_dm_Stage	mdot_dm_Rotor
HPVEE			
57860.79276	57108.64355	9.706475542	0.097063138
40646.64629	40121.09353	9.706656119	0.097064943
40619.0722	40087.83007	9.706612679	0.097064509
MATLAB			
7692.3	7625.7	9.7349	0.097347
7694.2	7628.2	9.7352	0.09735
7696.1	7631.2	9.7359	0.097357

LIST OF REFERENCES

[1] Lehrfeld MC. “Performance Improvements to the Naval Postgraduate School Turbopropulsion Labs Transonic Axially Splittered Rotor.” Master’s Thesis. Naval Postgraduate School, Monterey, CA. 2013. http://hdl.handle.net/10945/38970.

[2] Drayton S. “Design, Test, and Evaluation of a Transonic Axial Compressor Rotor with Splitter Blades.” PhD Dissertation. Naval Postgraduate School, Monterey, CA. 2013. http://hdl.handle.net/10945/37616.

[3] Grossman BL. “Testing and Analysis of a Transonic Axial Compressor.” Master’s Thesis. Naval Postgraduate School, Monterey, CA. 1997. http://hdl.handle.net/10945/9064.

[4] Hobson GV, Gannon AJ, Holmes W, McCormick M, Capece V. “Experimental and Numerical Performance Characterization of a Transonic Compressor Rotor Operating Behind an Inlet-guide Vane with Variable Flap Angles.” Proceedings of the ASME TTCE. GT2014-27308: pp. V02AT37A056. Dusseldorf, Germany, June 16-20, 2014. DOI 10.1115/GT2014-27308. http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1907454.

[5] Sayari NN, Bölcs AA. A New Throughflow Approach for Transonic Axial Compressor Stage Analysis. Proceedings of the ASME IGTACE. 95-GT-195: pp. V001T01A054. Houston, TX, June 5-8, 1995. DOI 10.1115/95-GT-195. http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=2130078&resultClick=3

[6] Descovich GL. “Experimental and Computational Investigation of Steam-Induced Stall in a Transonic Compressor Rotor for a Military Fan.” Master’s Thesis. Naval Postgraduate School, Monterey, CA. 2011. https://dklsp.ern.nps.edu/npslimdist/Lists/Theses/DispForm.aspx?ID=542.

[7] McNab DJ. “Experimental Testing and CFD Modeling of an Advanced Transonic Compressor for Military Applications.” Master’s Thesis. Naval Postgraduate School, Monterey, CA. 2011. https://dklsp.ern.nps.edu/npslimdist/Lists/Theses/DispForm.aspx?ID=438.

[8] Scanivalve Corp. “DSA3217/3218 Series Pressure Scanner Digital Sensor Array Operation and Service Manual.” (2018). [Online]. Available: http://scanivalve.com/wp-content/uploads/2018/04/DSA3200v123.pdf.

[9] National Instruments. “DATASHEET NI 9214 and TB-9214.” National Instruments, (2018). [Online]. Available: http://www.ni.com/pdf/manuals/375138a_02.pdf

[10] National Instruments. “DATASHEET NI 9402.” National Instruments, (2018). [Online]. Available: http://www.ni.com/pdf/manuals/374614a_02.pdf

[11] MathWorks. “NI-DAQmx Support from Data Acquisition Toolbox.” (2018). [Online]. Available: https://www.mathworks.com/hardware-support/nidaqmx.html.

[12] Hirsch C, Dring RP. “Through-Flow Models for Mass and Momentum Averaged Variables.” Contractor Report. Naval Postgraduate School, Monterey, CA. 1985. www.dtic.mil/get-tr-doc/pdf?AD=ADA173887

[13] Gannon AJ. (private communication), 2018.

[14] Borgnakke C, Sonntag RE. *Fundamentals of Thermodynamics.* John Wiley and Sons, New Jersey (2009).

[15] MathWorks. “Documentation.” (2018). [Online]. Available: https://www.mathworks.com/help/.

[16] Holmgren M. “X Steam, Thermodynamic properties of water and steam.” MathWorks, (2018). [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/9817-x-steam--thermodynamic-properties-of-water-and-steam.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California