NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

NEAR-STALL MODAL DISTURBANCES WITHIN A TRANSONIC COMPRESSOR ROTOR

by

Andrea Londoño

December 2011

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

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2011	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Near-Stall Modal Disturbances Within a Transonic Compressor Rotor		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Andrea Londoño			
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. IRB Protocol number _____N/A_____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE** A	

13. ABSTRACT (maximum 200 words)

This research effort seeks to better understand non-periodic flow characteristics for a forward swept axial transonic compressor rotor when operating near stall. Improved performance of a military gas turbine engine may be achieved by better understanding the mechanisms responsible for near-stall non-periodic disturbances within a transonic compressor rotor. Using pressure transducers, embedded within the rotor wall casing, data were acquired and calibrated at various speeds up to 90% of maximum rotation velocity. Within the 90% design speed, various data sets were acquired for different throttle configurations. A new method to post-process the data to allow better investigating of the non-periodic flow characteristics was developed. Using Fast Fourier Transforms, two distinct and dominant frequencies were identified and analyzed. Contour pressure distribution maps for varying throttle configurations; and the amplitude differences for each frequency of interest was generated to illustrate correlations in frequency strength and its relationship with tip-leakage vortices, normal/oblique shocks, and passage-to-passage interactions. This study uses effective instrumentation and robust data reduction techniques to successfully identify passage-to-passage distribution of non-periodic and periodic low dominant frequencies within the rotor blade passage prior to stall.

14. SUBJECT TERMS Transonic, Compressor, Pressure Instability, Low Dominant Frequencies, Turbomachinery, Near Stall Disturbances.			**15. NUMBER OF PAGES** 153
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** 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

NEAR-STALL MODAL DISTURBANCES WITHIN A TRANSONIC COMPRESSOR ROTOR

Andrea Londoño
Lieutenant, United States Navy
B.S., United States Naval Academy, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2011

Author: Andrea Londoño

Approved by: Dr. Anthony J. Gannon
Thesis Advisor

Dr. Garth V. Hobson
Thesis Co-Advisor

Knox T. Milsaps
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This research effort seeks to better understand non-periodic flow characteristics for a forward swept axial transonic compressor rotor when operating near stall. Improved performance of a military gas turbine engine may be achieved by better understanding the mechanisms responsible for near-stall non-periodic disturbances within a transonic compressor rotor. Using pressure transducers, embedded within the rotor wall casing, data were acquired and calibrated at various speeds up to 90% of maximum rotation velocity. Within the 90% design speed, various data sets were acquired for different throttle configurations. A new method to post-process the data to allow better investigating of the non-periodic flow characteristics was developed. Using Fast Fourier Transforms, two distinct and dominant frequencies were identified and analyzed. Contour pressure distribution maps for varying throttle configurations; and the amplitude differences for each frequency of interest was generated to illustrate correlations in frequency strength and its relationship with tip-leakage vortices, normal/oblique shocks, and passage-to-passage interactions. This study uses effective instrumentation and robust data reduction techniques to successfully identify passage-to-passage distribution of non-periodic and periodic low dominant frequencies within the rotor blade passage prior to stall.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

TCR	Transonic Compressor Rig
TPL	Turbo Propulsion Laboratory
RANS	Reynolds-Averaged Navier-Stokes simulations
PIV	Particle Image Velocimetry
CFD	Computational Fluid Dynamics
DPIV	Digital Particle Image Velocimetry
LES	Large Eddy Simulation
BPF	Blade Passing Frequency
OPR	Once-Per-Revolution

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

A special thanks to Dr. Gannon for his support throughout this study. To the faculty and staff of the Turbo Propulsion Laboratory a sincere thanks, without your help and guidance this effort would not have been possible. Finally, thanks to my best friend, my companion, my protector—my dog, Lady.

THIS PAGE INTENTIONALLY LEFT BLANK

1. INTRODUCTION

The purpose of this study is to better understand near-stall non-periodic disturbances for flow in axial transonic compressors. In previous research non-periodic flow phenomena has been observed but there has been little effort to understand why it occurs. This thesis developed better methods for observing non-periodic flow characteristics within an axial transonic compressor rotor when operating near stall. Maximized performance of a military gas turbine engine may be achieved by identifying the mechanisms responsible for triggering near-stall modal disturbances in two ways. First, by using better instrumentation and positioning them in effective locations within the compressor; and secondly, by developing data reduction techniques to identify disturbances within blade passages of an axial transonic compressor rotor fan close to stall. This thesis was part of ongoing research on transonic compressor rotors to improve their operating range and overall system performance.

The methods used in this thesis were derived from Gannon and Hobson's [1] investigation where non-periodic flow was noticed between the blade passages when running a Sanger compressor rotor [2] close to stall. Using high-speed pressure transducers along the inner casing wall of the compressor fan, non-periodic flow was detected although the compressor appeared to operate in a stable steady-state condition. The greatest low frequency disturbance was found within the blade row. This was a complex flow region where the tip vortex/normal shock and shock/boundary-layer interacted. This thesis extended Gannon and Hobson's [1] method and applied it to a modern forward swept fan. In order to understand the pressure non-axisymmetric flow characteristics occurring prior to stall, a focus was placed on the process used to locate dominant low frequencies within blade passages during pre-stall operations.

Due to technological advancement in turbomachinery, disturbances that have potential to lead to instability within a transonic compressor have become an important area of research within the past two decades. Studies have successfully indicated that unstable structures exist just prior to stall and have characterized those areas at low, mid, and high compressor design speeds. Attempts to develop stall warning and avoidance

procedures based on flow irregularities have been made, but little effort has been made to characterize the irregularity or understand their underlying causes Young and Day [3].

Two fields of research in axial compressors exist, one area focused on implementing mechanisms to avoid stall within the current stall margins. Other research maximizes compressor performance, by optimizing blade design, and locating flow mechanisms that lead to stall or surge. This paper falls under the later of the two cases by detecting the mechanisms involved prior to the stall inception process. The differences between non-periodic disturbances and pre-stall instabilities will be clarified. Further understanding of non-periodic instabilities has significant impact on the design of military compressors with regards to their stable operation during or prior to stall.

A. TYPES OF "INSTABILITIES"

Spike, modal waves, and non-periodic, dominant low frequencies between blade passages

There are three different types of pressure "instabilities" that lead to compressor stall or surge: spike, modal waves, and low frequencies. The term "instabilities" assumed these structures were pre-cursors to stall. However, the differentiation between instabilities and disturbances is that compressors are able to operate in steady-state condition with non-axisymmetric flow disturbances present. Spike and modal waves were normally associated with imminent stall. Low non-periodic frequencies, which will be termed disturbances, were not pre-cursors to stall since they appear in steady-state conditions throughout peak efficiency and pre-stall operations. A loose description of the instabilities for corresponding compressor designs for different speeds is provided for each term.

In low design speed machines (subsonic flow) a compressor can experience "spike" or short-wave disturbances in pressure for certain frequencies prior to stall as noted in Hah et al. [4], Hoss and Fottner [5], and Tan et al. [6]. Hah [4] showed unsteady random behavior of the tip vortex, and described this interaction within the passage shock to be critical ingredients for the development of a spike-type rotating stall. A "spike" occurred when the mass flow decreased and shifted the low momentum area upstream by

the trailing edge plane which initiated reverse-tip clearance flow. This activity created a forward spillage of the tip-clearance flow which ultimately affected the blade passage or passages to eventually stall.

"Modal waves" were also detected prior to stall at mid-speed operating compressors by Hoss and Fottner [5] and Zaki et al. [7]. In mid-speed (Mach 0.9-1.1) compressor operations, Hoss and Fottner [5] observed long-length scale disturbances around the circumference of the machine and interactions between the stall initiating oscillations.

Dominant "low frequencies" that occurred at steady-state operations between blade passages were particular to high-speed compressors as observed by Gannon and Hobson [8] and Hah et al. [9]. Gannon and Hobson [8] illustrated small instabilities present at the leading edge of the rotor row that eventually decayed toward the trailing edge of the blade. As the rotor operation shifted closer to stall the first significant low-frequency instability occurred in the aft portion of the rotor where the normal shock wave formed on the blade surface.

This research effort focused on developing techniques to find the distribution of low dominant frequencies within blade passages at varying flow conditions. This technique fills a gap in current turbomachinery related literature because it provides a better understanding of modal disturbances in compressors near stall.

B. CONTROL SYSTEMS FOR STALL AVOIDANCE

Active and passive control devices are the current methods used to avoid stall or surge. Active control systems consist of different actuators such as inlet guide vanes; bleed valves; and air injection methods to stabilize the flow field within a compressor. A passive control system involves casing and blade treatments, and does not change rotor geometry. The disadvantages of using control systems for stall avoidance were due to limitations in instrumentation and data reduction techniques as well as their inherent complexity. Using dependable instrumentation and sound data reduction techniques, this

thesis effort may be used in combination with an effective control system for stall avoidance; or used to understand the flow behavior to design better passive stall avoidance systems.

Active (feedback) control systems work by using sensors to measure static pressure, which feed back into air injectors when stall was detected, as observed by in Weigl et al. [10], Bailie et al. [11], and Chi [12]. Weigl et al. [10] used active feedback control to stabilize rotating stall or surge in a transonic single stage axial compressor. The system consisted of using sensors that measured the upstream wall static pressure patterns. These signals were used by the control system to control annular modulated air injectors. Bailie et al. [11] used inlet guide vane flow control by using wake generators that injected air from their trailing edges wake upstream of the fan.

Passive control systems, such as casing and blade treatments, usually focus in the tip region where tip-leakage flow occurs. Gourdian and Leboeuf [13] showed an increase in the interaction between the tip-leakage flow and the main flow when the mass flow was reduced, a phenomenon responsible for the development of a large flow blockage region at the rotor leading edge. A separation of the rotor suction side boundary layer was also observed at near stall conditions. Wilke and Kau [14] used axial slots to remove fluid out of downstream parts of the blade passage into the main flow further upstream. Jha and Rao [15] also used a casing treatment due to intense flutter at high pressure and temperature conditions.

C. UNDERSTANDING THE CAUSES OF STALL AND/OR SURGE

Experimental measuring techniques: particle image velocimetry, pressure transducers, and infinite tube sensors

Current measuring techniques include: Particle Image Velocimetry (PIV), pressure transducers, and infinite tube sensors. The nature of the PIV design provided adequate instrumentation for experimental purposes, but it is difficult to use for operating compressor instrumentation. Pressure transducers and infinite tube sensors can be easily installed in rotor casings for both experimental and operating compressor analysis. Although this research used pressure transducers, it would not be difficult to modify this

method to use infinite tube sensors. Infinite tube sensors have robust technology that allows them to be positioned further back from the wall casing while still providing the frequency response needed to investigate the low frequencies of interest.

Hah et al. [4] used PIV, an optical method to measure the velocity of the flow field, to find unsteady flow areas due to the oscillating tip clearance. They then compared those results along with case mounted unsteady pressure transducer results to numerical results using Large Eddy Simulations (LES). The PIV data showed that the tip clearance vortex oscillates substantially near stall, which agreed with LES model simulations of the same data set. Although this method was useful in experiments, it is not practical to implement in an operating engine.

Voges et al. [16] conducted an experimental investigation of the blade interaction with casing treatment using PIV. Their PIV laser system consisted of two periscope light sheet probes at three different radial positions (87.5%, 95%, and 99% of the blade height) that precisely aligned the laser light sheet through the blade tip clearance. Four operating points were investigated, one near stall and the other at peak efficiency, for both 100% rpm and 65% rpm speeds. The PIV laser system technique provided high-quality velocity information of the tip-clearance region of the rotor blades, and results obtained agreed with Computational Fluid Dynamics (CFD) calculations.

Wernet et al. [17] used Digital Particle Imaging Velocimetry (DPIV) in conjunction with Kulite dynamic pressure transducers to simultaneously capture transient velocity and pressure measurements of the non-stationary flow field during a compressor surge. Successful DPIV results were obtained for the high-speed centrifugal compressor. The high frequency response pressure transducers were used to reconcile the DPIV image data. The DPIV measurements provided instantaneous snapshots of the complex-flow field which helped understand the pre-surge phenomena occurring within the compressor.

Young and Day [3] used high-speed fast pressure transducers spaced over the six stages of the air compressor to conduct signal analysis in order to understand the irregularities or underlying disturbances associated with stall or surge for both low and high mass flow compressors. Gannon and Hobson [8] used two types of stagnation

pressure probes one set for steady state performance, and the other set to capture high-speed data when studying low-frequency instabilities.

A recent patent by Kulite [18] presented a pressure transducer assembly for measuring pressures in high temperature environments that use elongated tubes terminated at one end by an acoustic micro-filter. The micro-filter damper operated to absorb acoustic waves impinging on it with limited or no reflection. Hot gases propagated through the elongated tube and the transducer measured corresponding pressures. The acoustic filter operated to absorb acoustic waves resulting from the hot gases, therefore enabling the pressure transducer to be mainly responsive to high frequency waves associated within gas turbine operations. Future investigations could replace pressure transducers with this pressure transducer assembly.

Numerical prediction methods: LES, KES, using RANS

Simulations and models are also used to evaluate gas turbine engine stability and monitor stall detection. Teolis et al. [19] used signal-processing algorithms to perform gas turbine engine blade diagnostics and high cycle fatigue prognosis using minimal blade tip monitoring sensors. Teolis et al. [19] demonstrated detection of stall cell precursors using a single Eddy Current Sensor (ECS) in real-time tests on a NASA Rotor 67, single stage axial-flow compressor. Zaki et al. [7] used a hybrid of Reynolds-Averaged Navier-Stokes (RANS) and Kinetic-Eddy Simulation (KES) for stall predictions in a NASA stage-35 compressor as a representation of a modern compressor stage. This approach satisfactorily predicted the on-set of stall.

D. NPS RESEARCH ON NON-AXISYMMETRIC FLOW DISTURBANCES

Past NPS theses that focused on unsteady non-axisymmetric flow characteristics include works from Rodgers, Koessler, and Davis.

Rodgers [20] reestablished unsteady pressure measurements and test procedures for 60%, 70%, and 80% design speeds, for a Sanger [2] designed transonic compressor rig at NPS. Using the Sanger rotor, Rodgers established and developed data acquisition

and reduction systems. Rodgers' low speed results were compared to computational predictions and reasonable agreement was obtained.

Koessler's thesis [21] used Fast Fourier Transforms (FFTs) to signal process steam ingested data captured by pressure transducers to look at three frequencies within the flow field: the blade passing frequency, the once-per revolution frequency, and a pre-cursor frequency. The development of the stall pre-cursor frequency was clearly illustrated as the throttle setting moved closer to stall. Koessler also noticed the once-per-revolution frequency disappeared then reappeared as the throttle moved from the peak efficiency point to the near stall point at 90% compressor design speed. The presence of a stall precursor in both the steady state and steam-induced stall data suggested that active control of stall could be possible.

Davis' [22] research analyzed the behavior of a transonic compressor when approaching stall for the rotor and full stage configuration using combinations of frequency and time domain analysis for different operating ranges; these included sub-sonic, sonic, and transonic speeds. Davis' project revealed increased irregularity in the blade-passing signature, and dependence for both tip-clearance and eccentricity in the axial direction. For a compressor with a small, uniform, tip-clearance, the increase in blade passing irregularity accompanied a modest reduction in flow rate. When the tip-clearance was enlarged, a sharp rise in irregularity at all circumferential locations occurred. Davis concluded that a compressor with eccentric tip-clearance experienced increased irregularities, however his research was unable to detect where the irregularities originated within the passage.

E. LITERATURE REVIEW

Conclusions: Tip-leakage vortex, Eccentricity, and blade passing frequencies

Current studies have concluded that the underlying cause(s) behind the stall inception process depend on tip-clearance, eccentricity, and low-frequency instabilities. In an effort to understand when and why irregularities occur, Young and Day [3] concluded that the blade-passing signature of each rotor blade was different from its neighbor and from itself in the previous revolution. The blade passing irregularity and

proximity to stall was linked to tip-clearance and eccentricity. It was also noticed that the irregularity level increased with a large tip clearance. An identification of these discrete pre-stall propagating disturbances was found to be responsible for the irregularity observed in blade passing signatures. Furthermore, they concluded that blade passing signatures could not be used to provide a reliable stall warning system for a real engine.

Hah et al. [9] studied the unsteady flow phenomena due to tip clearance flow in a modern transonic axial single stage compressor rotor. The data was acquired using PIV and pressure transducers and both indicated the tip clearance vortex oscillating substantially near stall. The spectral analysis showed two dominant frequencies: one frequency between 40–60% of rotor rotation and the other dominant frequency was between 40–60% of the Blade Passing Frequency (BPF). The first frequency represented the movement of a disturbance spanning several consecutive blade passages against the rotor rotation. The second frequency represented the traditional tip flow instability, which was widely observed in subsonic compressors. LES simulations showed the second frequency was due to the movement of the instability vortex. They recommended a multi-passage analysis to analyze flow near stall operation and predict stall limits in compressor operations.

Yamada et al. [23] explained the tip clearance flow field in a transonic axial compressor rotor, especially at near-stall condition, and investigated the effect of the tip leakage vortex for near-stall flow. Using NASA Rotor-37, the tip-clearance flow field was investigated by unsteady RANS simulations and a method of identifying vortex structures based on the critical-point theory. Yamada's results indicated tip clearance flow fields were due to the breakdown of the tip-leakage vortex that led to a large blockage near the tip and within the rotor passage. The breakdown of the tip-leakage vortex yielded a rise to the fluctuations of the blade torque, the low energy fluid and the shock wave near the rotor tip. Yamada concluded that the large blockage effect and the unsteadiness due to the tip leakage vortex breakdown could be related to the spillage of the tip clearance flow at stall inception. Yamada recommended carrying out multi-passage simulations and further investigating relationships within single-passage simulations.

F. FINDING A PARTICULAR FREQUENCY WITHIN THE PASSAGE

This research effort pursues the better understanding of non-periodic flow characteristics of a forward swept axial transonic compressor rotor when operating near stall by developing comprehensive data reduction techniques that could be implemented in military gas turbine engines. The fan used for this study was a modern design, modified at the Varva Turbopropulsion Laboratory (TPL) at the Naval Postgraduate School (NPS) for research in compressor fan design optimization and near-stall operations.

Using pressure transducers embedded within the rotor wall casing, the data acquired was reduced by calibrating, processing, and analyzing each of the 70%, 80%, 85%, and 90% compressor speed lines. Within each design speed, various data sets were acquired for different throttle configurations: peak efficiency to near-stall.

Conversion from the time, to space, and finally to the frequency domains were necessary to achieve a final distribution map of the pressures. The mean pressures, from successive over plots of each passage pressure trace, were used to create the contour plots of the pressure field for each design speed at varying flow conditions.

Blade passage data sets from pressure fields were transformed from the time to the frequency domain using FFTs to provide frequency and amplitude maximums for each probe at discrete locations within the blade passage. The maximum amplitudes were located within the rotor passage and two distinct dominant frequencies were found. The first being the rotor rotational frequency and the second found at 56.5% of the rotor rotational frequency. A contour map of the amplitude for each frequency of interest was generated to illustrate correlations in frequency strength and its relationship with tip-leakage vortices, normal/oblique shocks, and passage-to-passage interaction throughout varying flow conditions.

Recent studies have shown a need for multi-passage simulations and full transient analyses near stall e.g., Young and Day [3], Hah et al. [14], and Yamada et al. [23] as a basis to understand the fluctuations in pressure within the rotor passage. This study uses effective instrumentation and robust data reduction techniques to successfully observe

modal periodic (multiples of rotor speed) and non-periodic (not related to rotor speed) disturbances. The 90% speed line was investigated in detail.

II. EXPERIMENTAL FACILITY AND PROCEDURE

All experiments within this thesis were conducted at the Vavra Turbo Propulsion Laboratory (TPL) test facility. The hardware includes the Transonic Compressor Rig (TCR), compressor fan, and pressure transducers. The software consists of steady state and high-speed data acquisition systems.

A. TEST FACILITY DESCRIPTION

Two opposed-rotor air driven turbine stages, supplied by a 12-stage Allis-Chalmers axial compressor drove the transonic compressor rig. The Allis-Chalmers air-compressor takes in air directly from the atmosphere and delivered it to the TCR to drive the rotor at the desired rotor speed. Figure 1 shows the TCR schematic, however the steam generator and accompanying equipment is not applicable to this thesis.

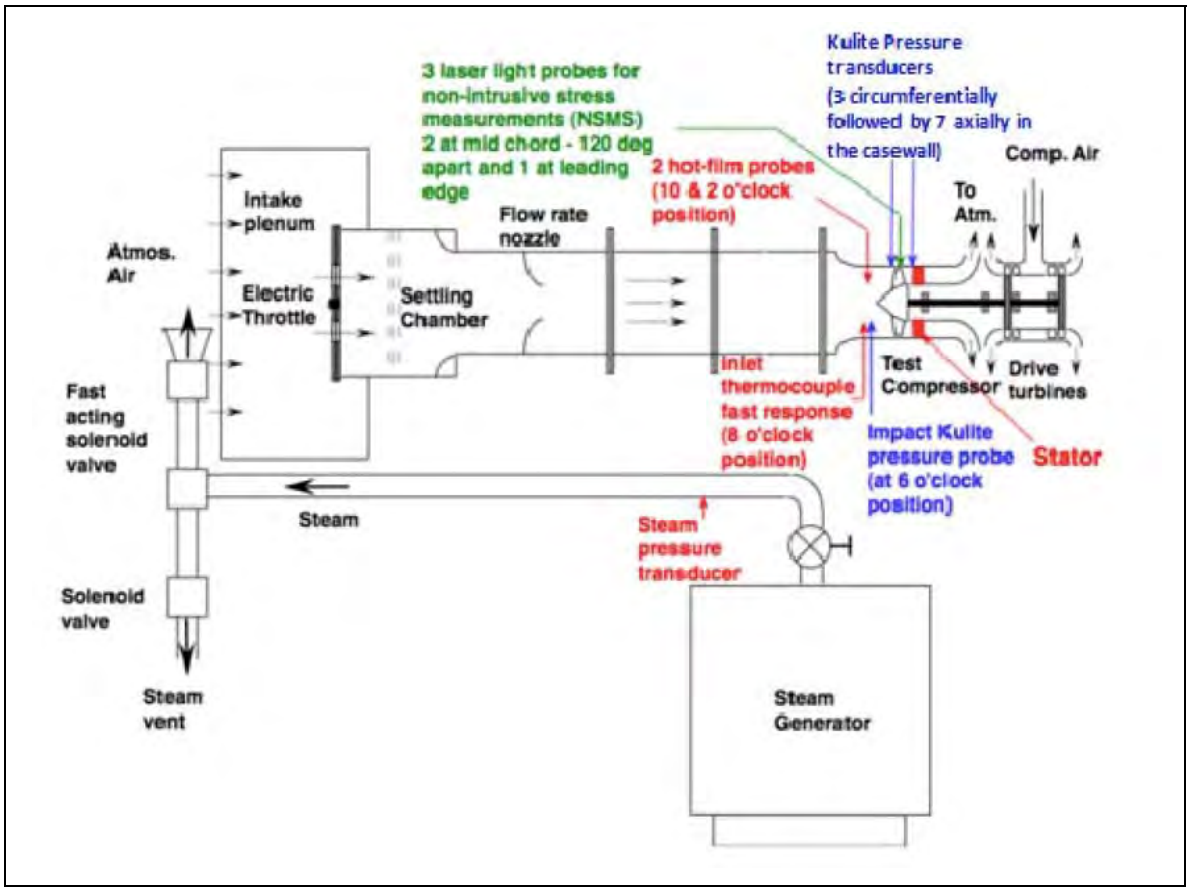


Figure 1. TCR Schematic [22]

Data acquired from the rotor was taken during steady-state operations for 70%, 80%, 85%, and 90% speed lines. The design speed of the TCR (30,000 RPM) was not tested due to the open-loop nature of the facility and the Allis-Chalmers compressors' power limitations. Table 1 shows the rotors speed profile.

Rotor Speed	RPM	Tip-Speed
90%	27,000	406 m/s
85%	25,500	383 m/s
80%	24,000	361 m/s
70%	21,000	316 m/s

Table 1. Rotor Speed Profile

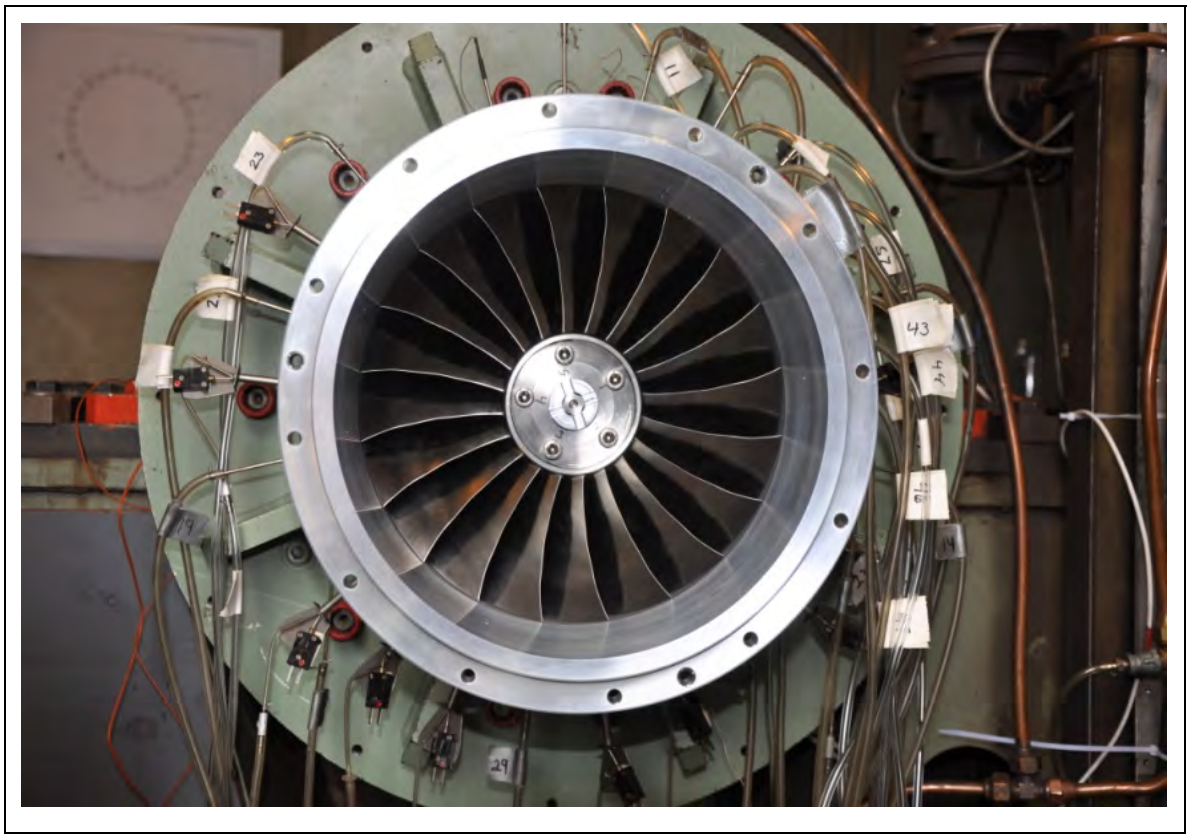


Figure 2. Transonic Compressor Rig with Forward Swept Fan

1. Test Fan Characteristics

The forward swept fan (shown in Figures 2 and 3) used in this experiment was designed to represent modern fans in operation in the military. This fan was modified at the Vavra Turbo Propulsion Laboratory for research in blade optimization and near-stall operation.

Figure 3. Modern Forward Swept Fan

The modern fans characteristics are listed in Table 2. Of note, this modern fan experiences a pressure ratio of 1.7 at peak efficiency at 90% speed, and has a higher blade loading than the previous Sanger rotor [2] researched at the TPL. A thin blade thickness design was used to block the effects caused by the fans relative high Mach number of 1.52. The fan's tip-clearance was negligible due to an abradable lining located within the casing wall that physically touched the rotor blade-tips during operating conditions. The abradable strip technique was passed from successful research experiments on the Sanger rotor, where the Sanger rotor experienced a minimal tip-gap clearance of 0.635 mm (0.025") at non-operating conditions and minimal clearance at

operating conditions [2]. Providing minimal tip-gap clearance was important to this research, since tip-gap clearance vortex breakdown has been considered a possible cause of stall initiation Hah et al. [24].

Fan Characteristics at 90% Speed	
Blade Number	20
Rotor Diameter	287 mm (11.3”)
Tip Clearance	0.635 mm (0.025”)
Nose Diameter	91.4 mm (3.6")
Casing Diameter	287 mm (11.3")
Tip Mach Number	1.52
Tip Speed	406 m/sec (1330’/sec)
Total Pressure Ratio	1.70 near stall
Mass Flow	10.14 kg/sec (22.35lbm/sec)

Table 2. Fan Design Specifications

The compressor pressure map shown in Figure 4 was produced by Descovich [25] from steady-state data also acquired using installed combination temperature/pressure probes ahead of and aft of the forward swept fan. Each speed line began at maximum mass flow for each desired speed. The marks on the speed lines indicate the points where data sets were taken. At each mark the mass flow was decreased while maintaining the desired rotor speed. The most left point on each speed line denotes pre-stall operations, which shows the minimum mass flow for each speed profile.

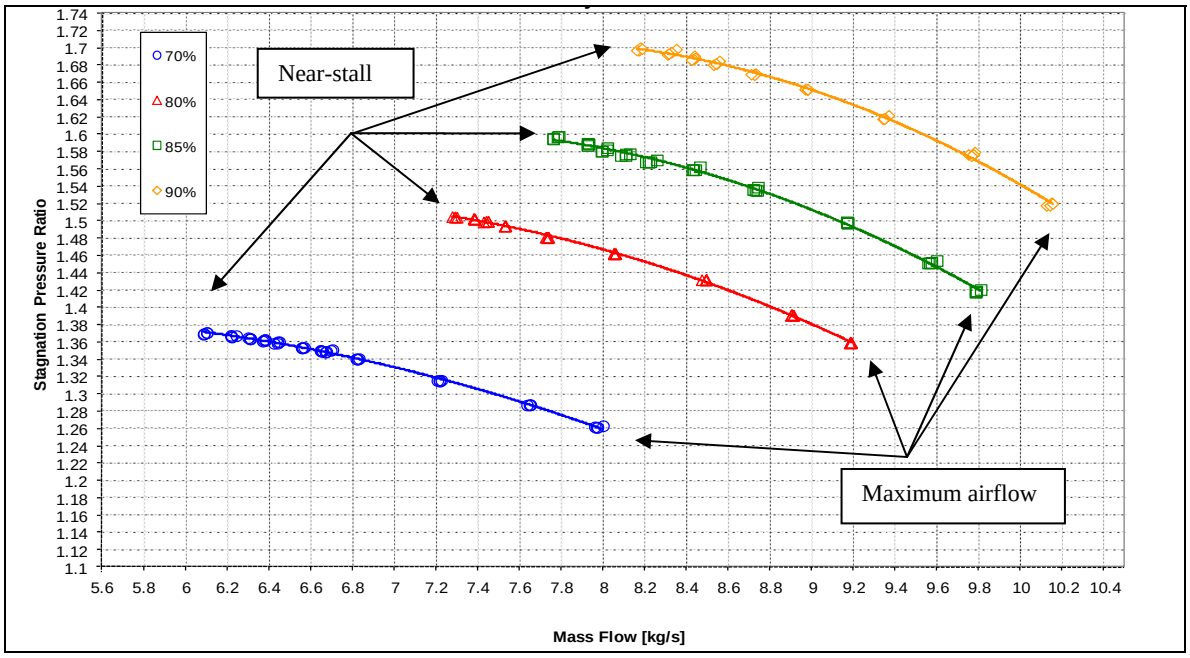


Figure 4. Compressor Pressure Map [25]

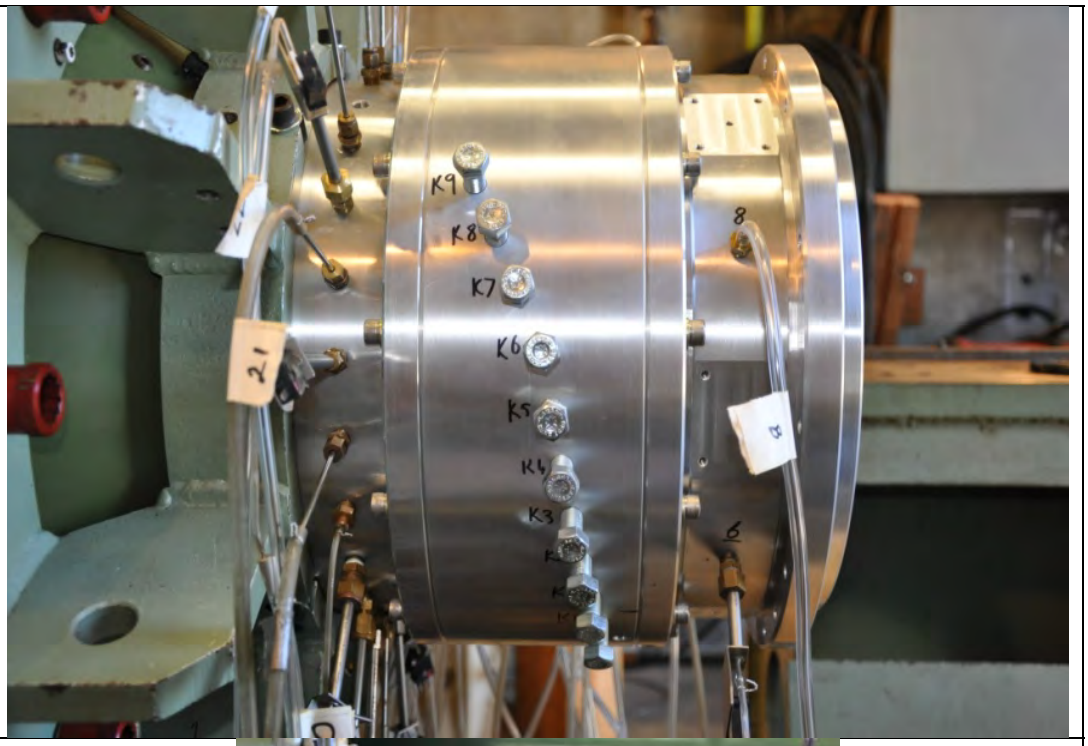
2. Instrumentation

Pressure transducers were used to record the instability distribution within the modern forward swept fan. Kulite Miniature IS Pressure Transducers (XCQ-080 Series) were chosen to capture the high-speed data. Their specifications are listed in Table 3. Figure 5 shows the TCR before and after Kulite pressure transducers were inserted.

Input Pressure Range	1.7 Bar (25 psi)
Diameter	2 mm (0.078”)
Natural Frequency	240 KHz
Input/ Output Impedance	1000 Ohms
Acceleration Sensitivity % FS/g (Perpendicular/Transverse)	5.0x10-4 FS/g and 6.0x10-5 FS/g
Operating Temperature Range	-65 F to +250 F (-55C to +120C)
Weight	0.3 Grams (Including Module and Leads)

Table 3. Kulite Miniature IS Pressure Transducer (XCQ-080 Series) Specifications [26]

A



B

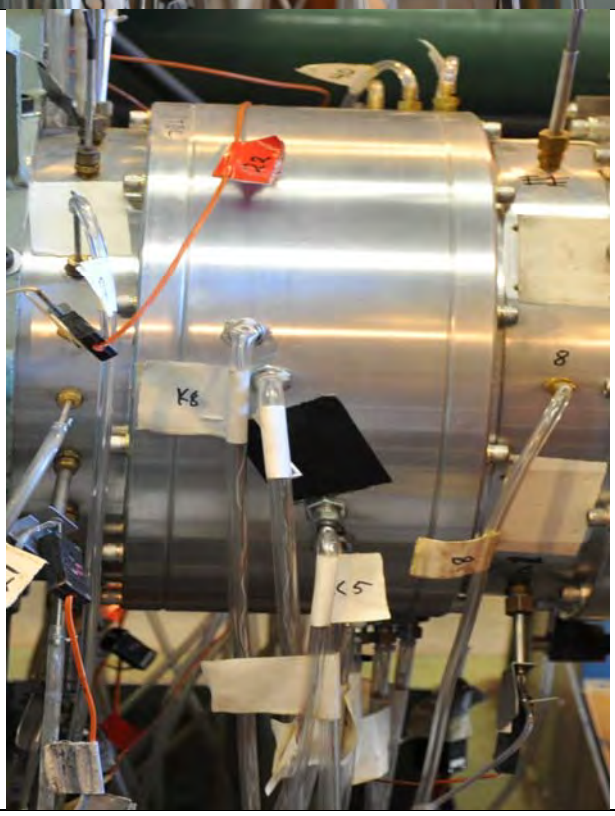
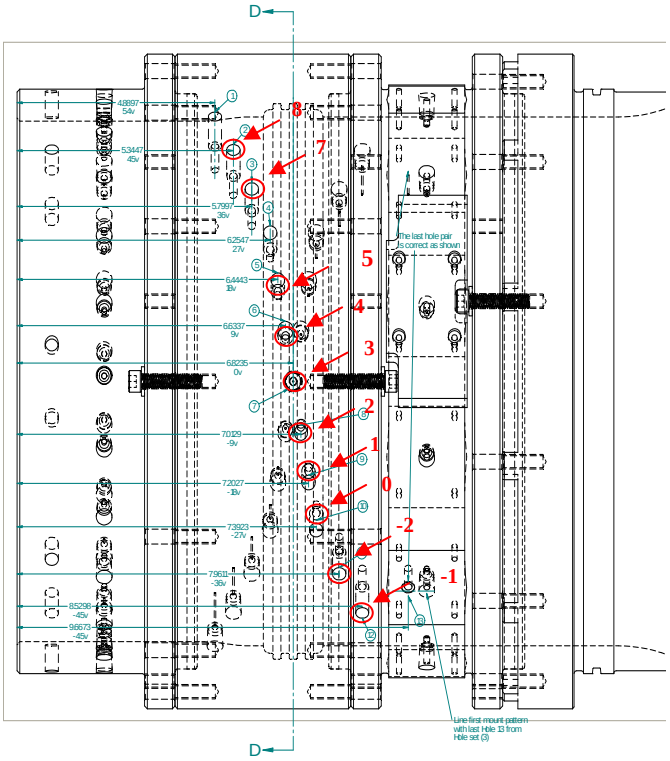


Figure 5. Instrumentation Position Outside of Rotor Casing

Pressure transducers were critical in achieving results for this experiment. The Kulite probes were embedded within the casing wall upstream, directly over blades, and downstream of the rotor to measure the pressure as the fan operated.

A total of 14 probes were inserted at 9° intervals around the rotors circumference and at varying radial stations shown in Figure 6 (A and B). An enlarged engineering drawing is located in Appendix F. This was an increase from five pressure transducers used for research on the Sanger rotor by Gannon and Hobson [8]. They concluded that more instrumentation was needed to better observe the flow with each passing blade. More instrumentation allowed better precision when plotting near-stall pressure instabilities. An impact probe was inserted upstream of the rotor to measure stagnation pressure changes upstream of the transonic compressor, however the position of the probe did not affect the rest of the data acquired.

A



B

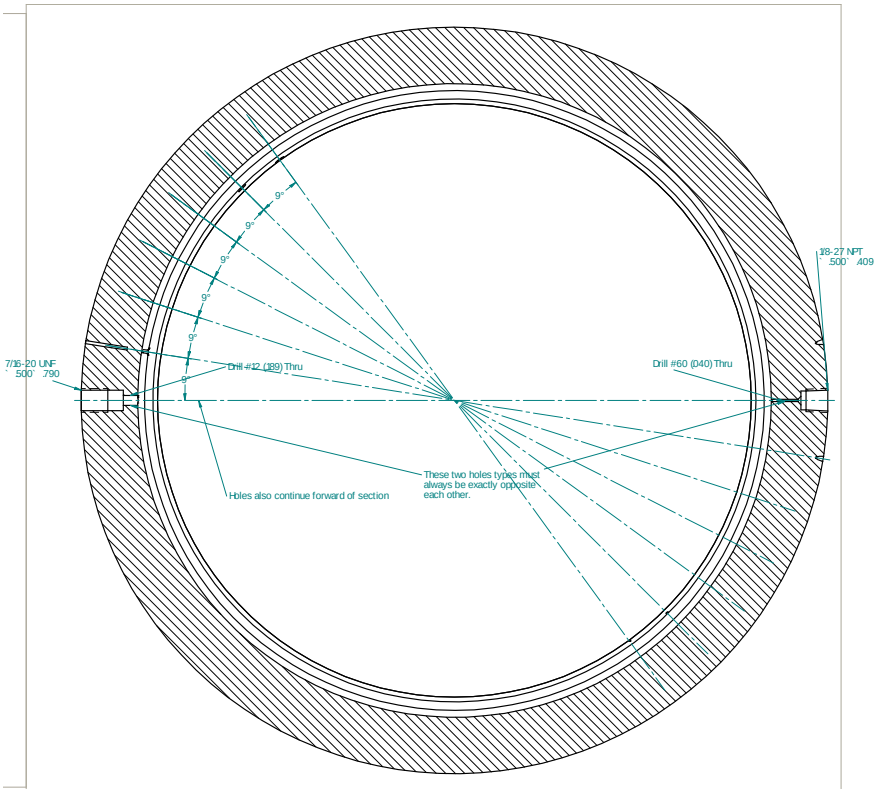


Figure 6. Pressure Probes Radial Stations and Dimensions

Of the 14 probes embedded, ten Kulite probes were used to acquire the data presented in this thesis. The Kulite probes were chosen because they were statically and dynamically capable. Most importantly these pressure transducers met the data acquisition sampling rate of 196,608 Hz with a natural frequency of 240 kHz. Figure 7 better illustrates the dimensions and diminutive nature of the Kulite Miniature IS Pressure Transducer.

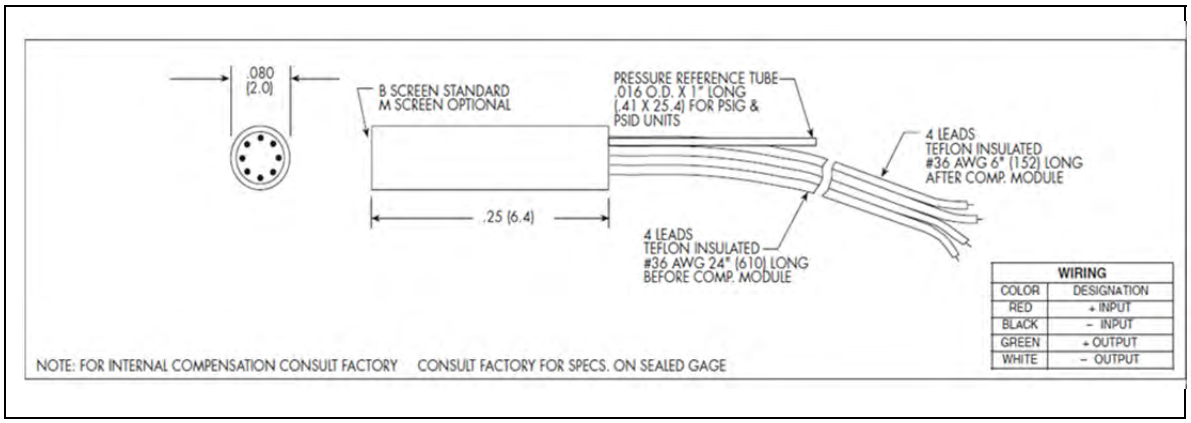


Figure 7. Kulite Miniature IS Pressure Transducer [26]

The DAC Express data acquisition system compiled the data acquired during the experiment. The DAC Express was capable of analyzing a maximum of 16 channels simultaneously for any length of time. This software sampled and stored data via its two mainframes. This thesis experiment recorded two-second samples at a speed of 196,608 Hz for different flow and operating conditions. Maintaining appropriate ranges for the instruments prevented data saturation and subsequent "clipping" of the signal and chopped endpoints. These settings had to be changed for different speeds and positions on the speed line. Data files were saved in comma delimited form (*.csv) for readability, and ease for exporting to Microsoft Excel and processing in MATLAB. Over half a million data sample points were acquired for each run, which required the user to ignore the export warning system stating "*the number of samples exceeds the maximum worksheet size of 65536 rows for Excel.*"

The calibration procedure consisted of taking 0.5 second samples for successive backpressures of varying pressures for 0, 5, 10, and 15 inches of Mercury on the barometric scale (0, 16930, 33860, 50790 Pa). This calibration technique was applied to the backside of the probe. All probes were calibrated before and after each TCR experiment to ensure that they had remained operational during the experiment. The negative voltage (Figure 8) was due to possible probe leads being switched.

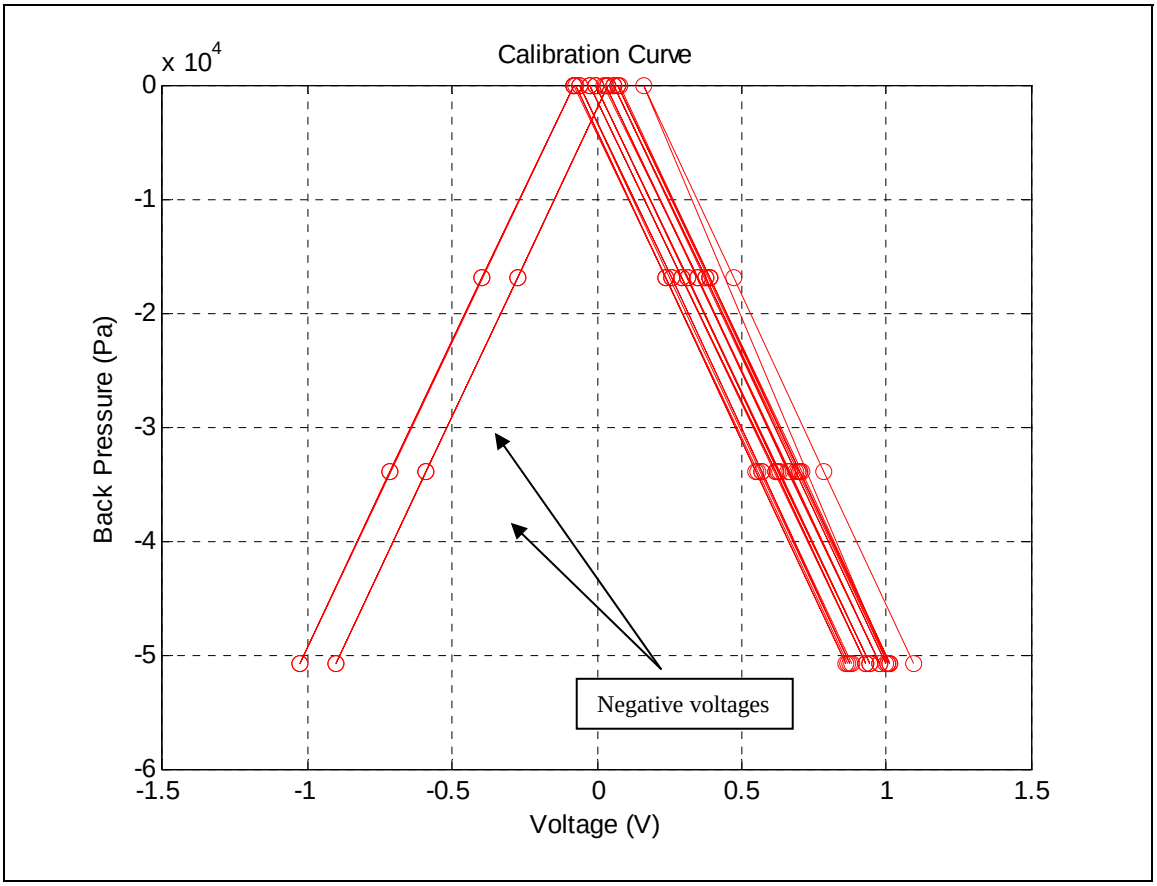


Figure 8. Calibration Curve

The additional impact probe located upstream of the airflow was used to measure inlet pressure transients, but was not used during data reduction nor did it affect the acquired data due to its location placement. The purpose of the impact probe was to measure the transient stagnation pressure, while the wall probes measured static pressure.

B. EXPERIMENTAL PROCEDURE

Two separately operated throttles run the TCR. The upstream throttle controls the air into the compressor and the other throttle controls the air that drives the turbines. The compressor speed was set at each speed line while decreasing air mass flow to approach stall. Table 4 indicates the throttle setting position at each point along the speed line. Full open throttle has a throttle setting of 0. The airflow is reduced by increasing the throttle position.

90% Speed (27000 RPM)	
Run 1-4	Full Open Throttle (Not at full speed due to power limitation)
Run 5-8	Throttle Setting 5.6
Run 9-12	Throttle Setting 6.25
Run 13-16	Throttle Setting 6.75
Run 17-20	Throttle Setting 6.9, near stall

Table 4. Speed Profile: 90%

Two-second samples were taken for each data sample in between speed lines. The 90% speed profile is shown in Table 4, while the rest of the design speed profiles are found in Appendix E.

III. DATA REDUCTION AND PROCESSING PROCEDURE

The data reduction process consisted of analyzing and processing each of the compressor design speed data lines. Post-processing focused on presenting the data relative to the blade passages.

A total of 17,000 blade passages were sampled in two-second worth of data. One hundred points were used to evenly divide a blade passage. The mean of the blade passage pressures were transformed into a pressure distribution map. This contour process consisted of taking the average of all the pressure signals at each of the 100 points across the passage to produce the smooth mean. The pressure averages for each Kulite data set were sequentially joined and interpolated to generate the final contour plot.

A. POST PROCESSING TECHNIQUES

The probes were positioned half a pitch (9°) apart along the rotor casing with “probe-0” at the leading edge and “probe 6” at the trailing edge of the blade (Table 5 and Figure 9).

Probe Number	**Radial Position**	**Axial Position**
Kulite '-2'	-45°	8.5298"
Kulite '-1'	-36°	7.9611"
Kulite '0'	-27°	7.3923"
Kulite '1'	-18°	7.2027"
Kulite '2'	-9°	7.0129"
Kulite '3'	0°	6.8235"
Kulite '4'	9°	6.6337"
Kulite '5'	18°	6.4443"
Kulite '6'	27°	6.2547"
Kulite '7'	36°	5.7997"
Kulite '8'	45°	5.3447"

Table 5. Kulite Probe Radial and Axial Positions

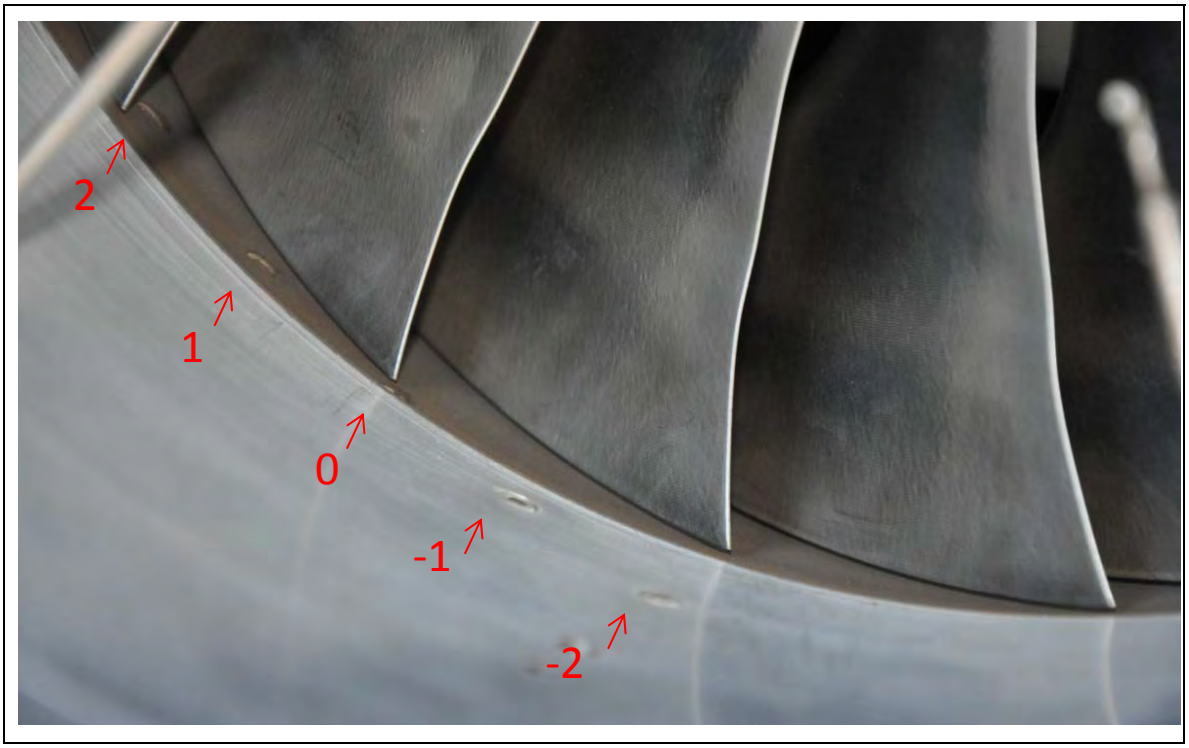


Figure 9. Probe Location with Respect to Blades

Figure 10 illustrates the physical location of the probes if they were unwrapped from the casing wall onto a flat surface. Figure 10 also shows the sequence of moves used to shift the position of the pressure probes from the casing onto the blade. This shift in position was done to simplify the post-processing steps that followed.

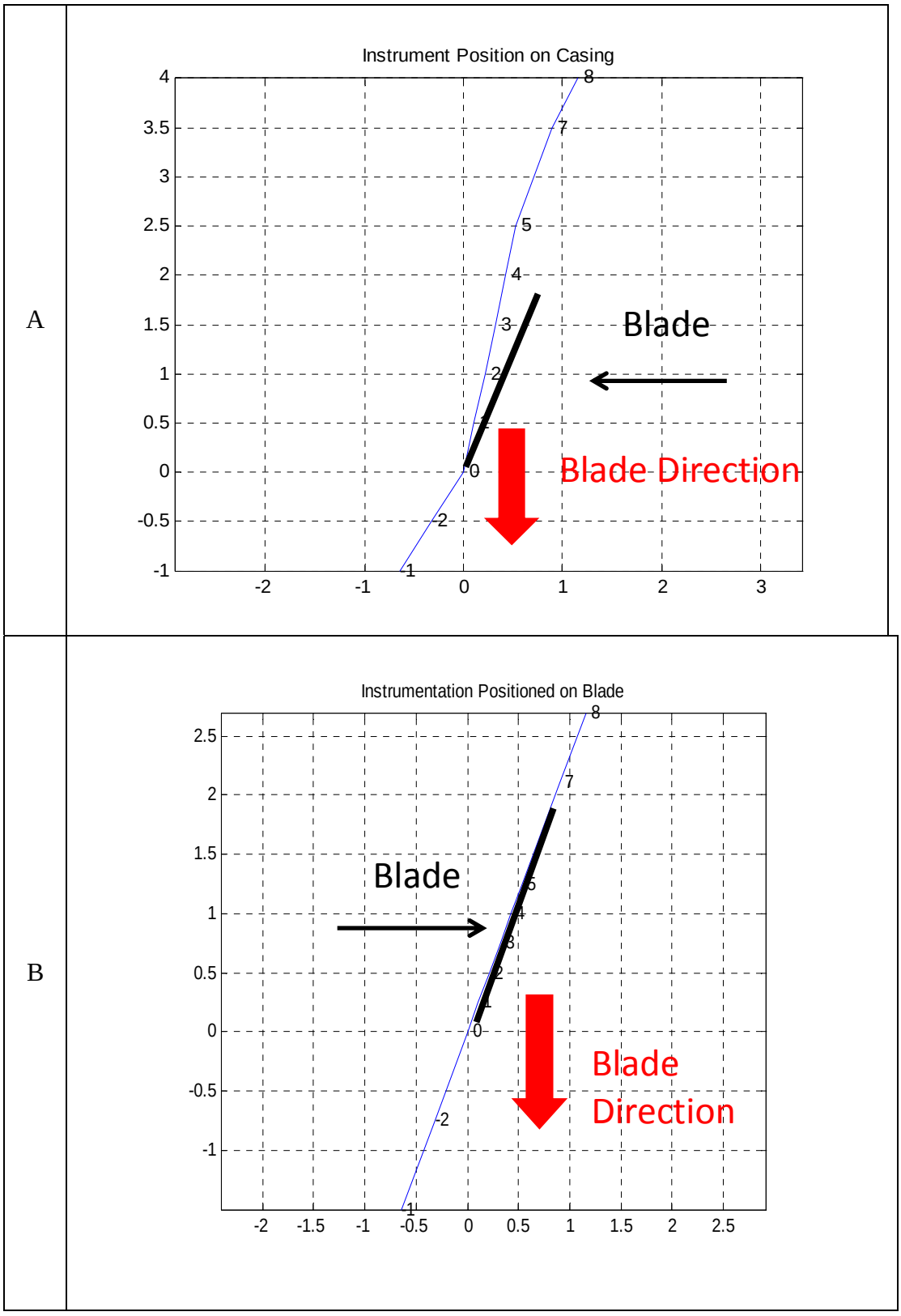


Figure 10. Pressure Probes Shifting from Casing to Blade

The pressure signals were non-dimensionalized with respect to the blade pitch, with one blade pitch being unity. By converting the rotors passage rotations from revolutions to single blade passages, each revolution signal was converted to the width of the passage. This simplified the post-processing of the signals and made it easier to verify that the data, from different Kulite probes, was combined correctly to produce meaningful contour plots. It was also useful is correcting for the rotor's speed which changed during sampling.

Figure 11(A) shows the trigger signal of each blade passage read from the once-per-rev pulse. This signal was plotted and analyzed because there was a position shift noticed through the sample. Converting from the time to the space domain (passage non-dimensionalization) successfully allowed this investigation.

Figure 11 (B) shows the trigger signal ahead from where it should be; this is due to inconsistencies in the electric motor, fluid coupling, 12-stage axial compressor, and the transonic compressor itself. Since the inconsistencies from the mechanisms could not be adjusted, the correction process was limited to manipulating the data already acquired.

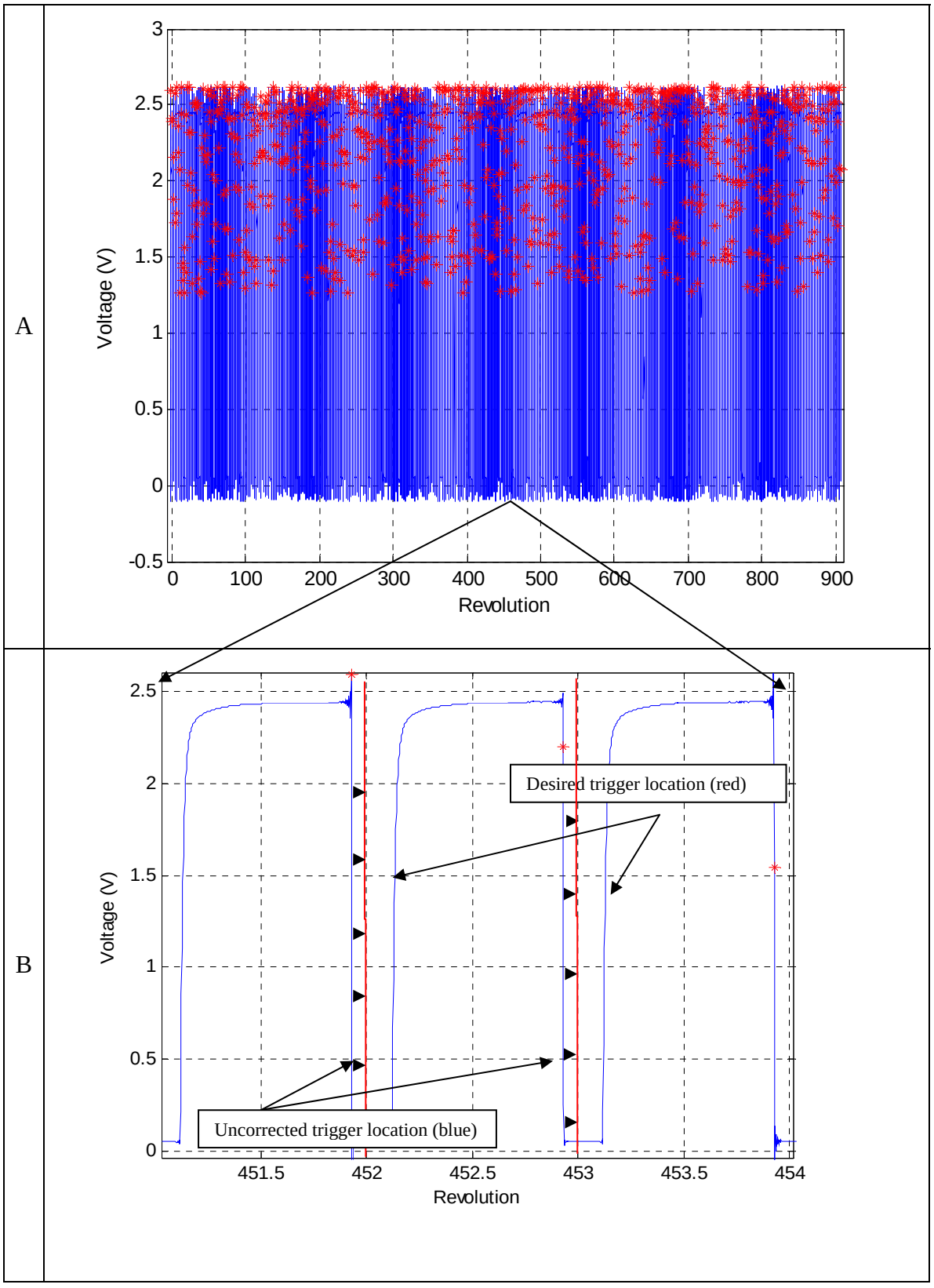


Figure 11. Uncorrected Trigger Locations

Although the acceleration error was small, it was still significant enough to create a problem when plotting individual probe pressure plots. The greatest error amounted to three complete passages. The correction process consisted of a 4-step procedure.

1. Plotted the error and the trigger position vector.
2. Smoothed the error curve in Figure 12 (A) using a robust quadratic fit with Matlab's 'rloess' function.
3. Interpolated the trigger error to equal the length of the position vector.
4. Corrected the once-per-rev signal by adding the trigger position error to the time data.

A visual inspection verified that the correction process was successful and that the triggers lay at integer values as shown in Figure 13 (A and B). The maximum error of 0.075% was found between the 200^{th} and 300^{th} rotation at the peak of Figure 12 (B).

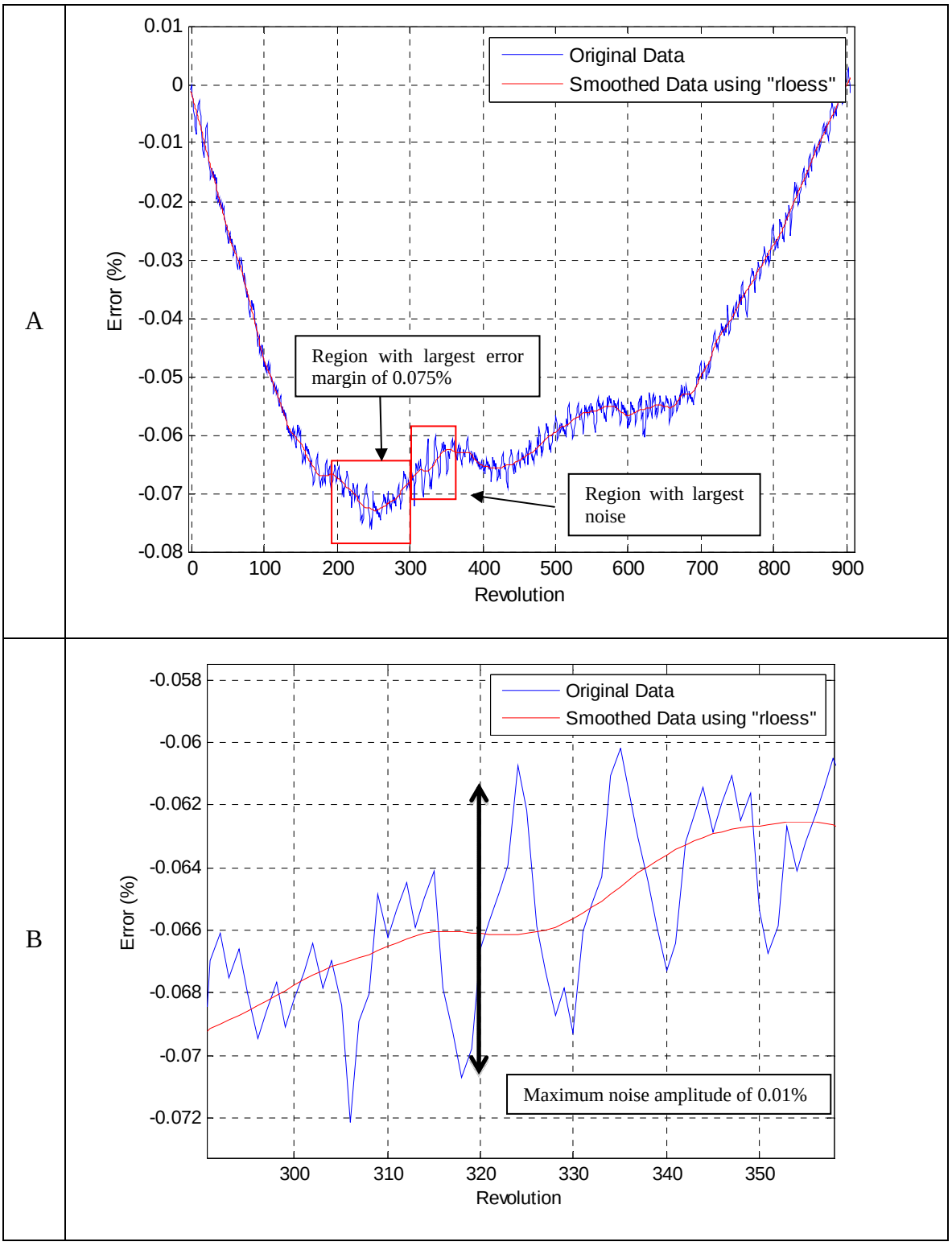


Figure 12. Smoothed RPM Error

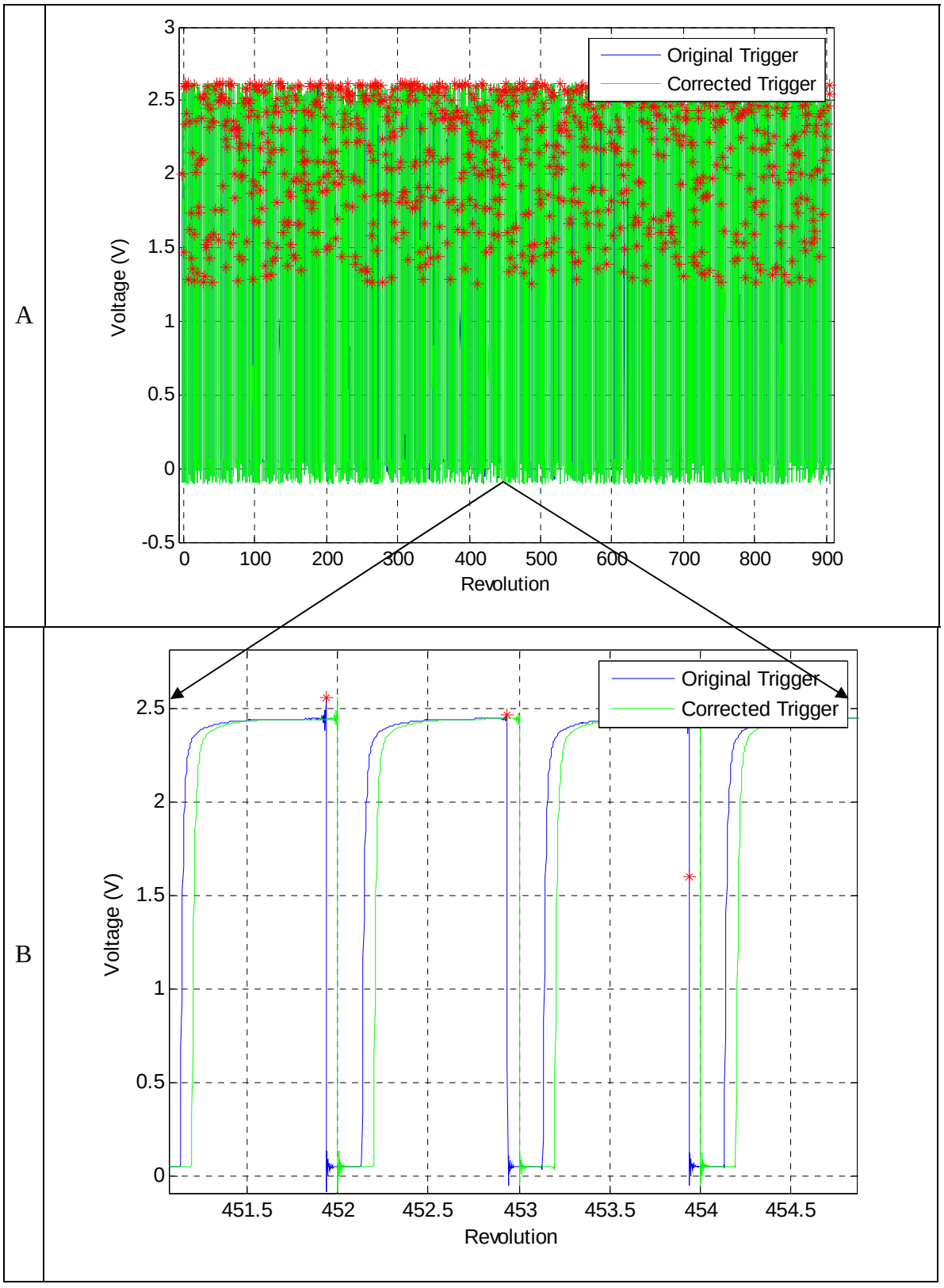


Figure 13. Corrected Trigger Plot

The complete data set of converted pressures into rotational passages is shown in Figure 14 (A). Figure 14 (B) is a zoomed portion of the raw data of a single passage of pressure data.

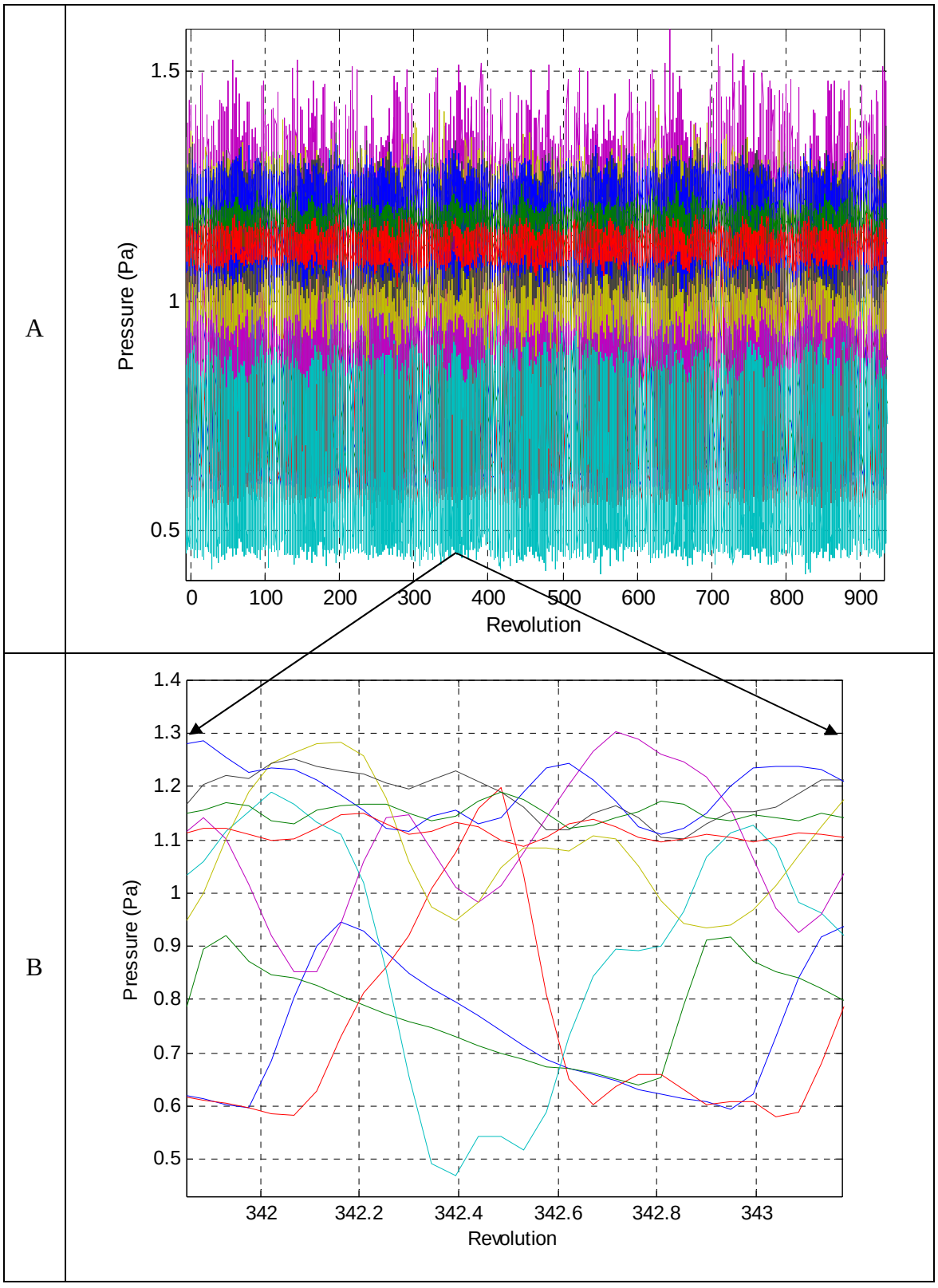


Figure 14. Pressure Profile for a Single Passage

Figure 15 shows the mean of all the passages; this pressure mean was used to generate the pressure contour map of the passage.

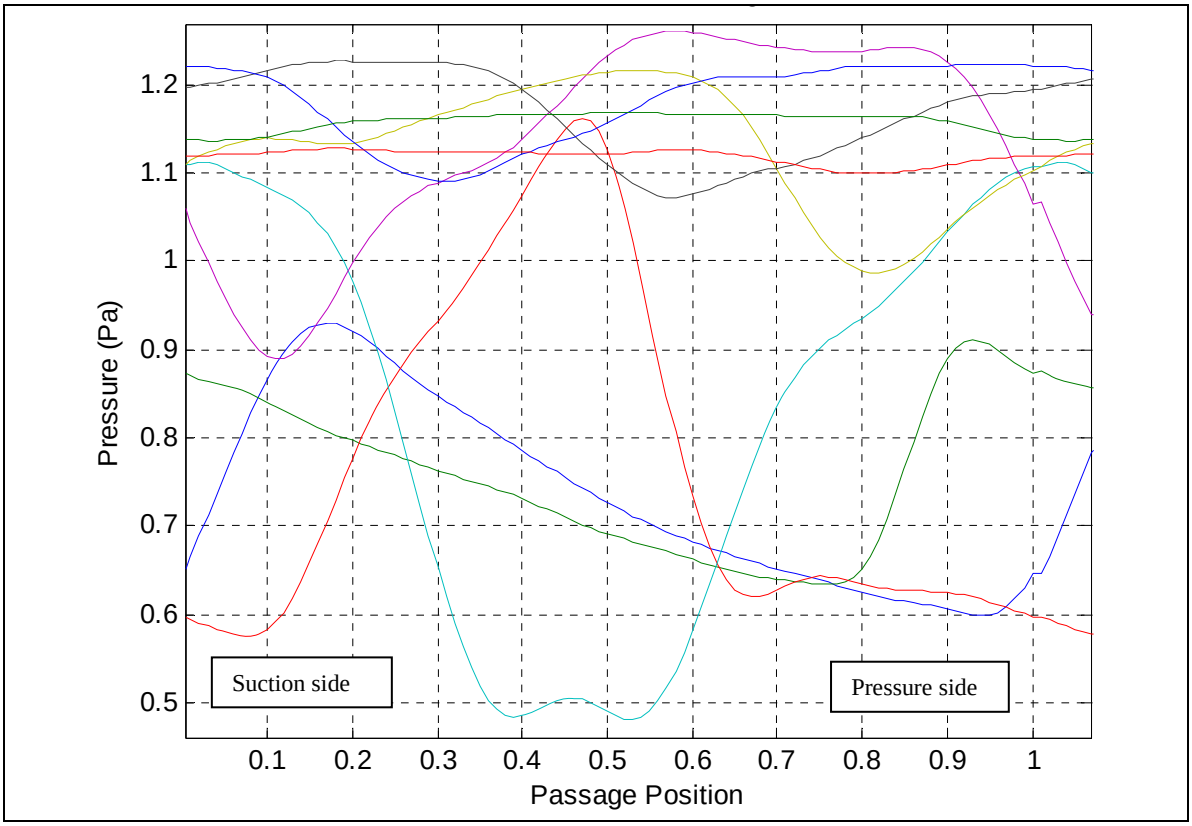


Figure 15. Passage Mean Pressure

B. DATA ANALYSIS

When designing compressors it had been assumed that the flow was periodic (or the same in each passage). This assumption was not supported by the sampled data after superimposing the successive blade passages for each Kulite. Young and Day also found the erratic nature in which the blade-passing signature of each rotor blade was different from its neighbor and different from itself in previous revolutions [2]. Figures 16–25 illustrate how individual passages varied for all pressure probes when plotted successively, 50 of the 17,000 passages calculated are shown.

The acquired data was sampled at regular intervals. Regular intervals allowed the data sets to be analyzed by converting them from the time to the position domain. This

process consisted of converting the rotors passage rotations from hertz to single blade passages as explained previously. These single blade passages were superimposed on each other. The 50 individual blade passages were evenly spaced using 100 points. Each point was used find the pressure variation throughout the data sample. The average pressure was calculated by taking the mean of the pressures at each of the 100 points for all the passages. This was done for all 100 points for each Kulite data set.

Kulites '-1' and '-2' (Figures 16 and 17) were located upstream of the rotor. These Kulite probes experienced fairly periodic flow with little change from one passage to the next. Figure 18 shows Kulite probe '0' which is located at beginning of the leading edge of the blade. This portion of the blade encounters the beginning of the normal shock formation and some inconsistent flow.

Figures 19–23 show Kulite probes 1–5, which measure the pressure along the entire length of the blade; these probes were spaced less than 5.08 mm (0.2") apart in the axial direction and 9° in the radial direction (see Table 5). It is visually apparent that the flow was becoming less periodic, but only at certain positions within the passage depending on the location of the probe with respect to the blade. Figure 20 is significant because Kulite '2' is positioned near the center of the blade. This Figure makes it clearly visible that this region experienced non-periodic flow for the majority of the passage.

Kulite probes '7' and '8' (Figures 24-25) were located downstream of the flow and directly after the rotor. It is evident that the flow in this region is significantly non-periodic. In order to improve pressure sampling in this region, more probes can be inserted, however this investigation is focused on the modal disturbances occurring along and near the leading edge of the blade as this is where the normal shock moves to when the rotor operates close to stall.

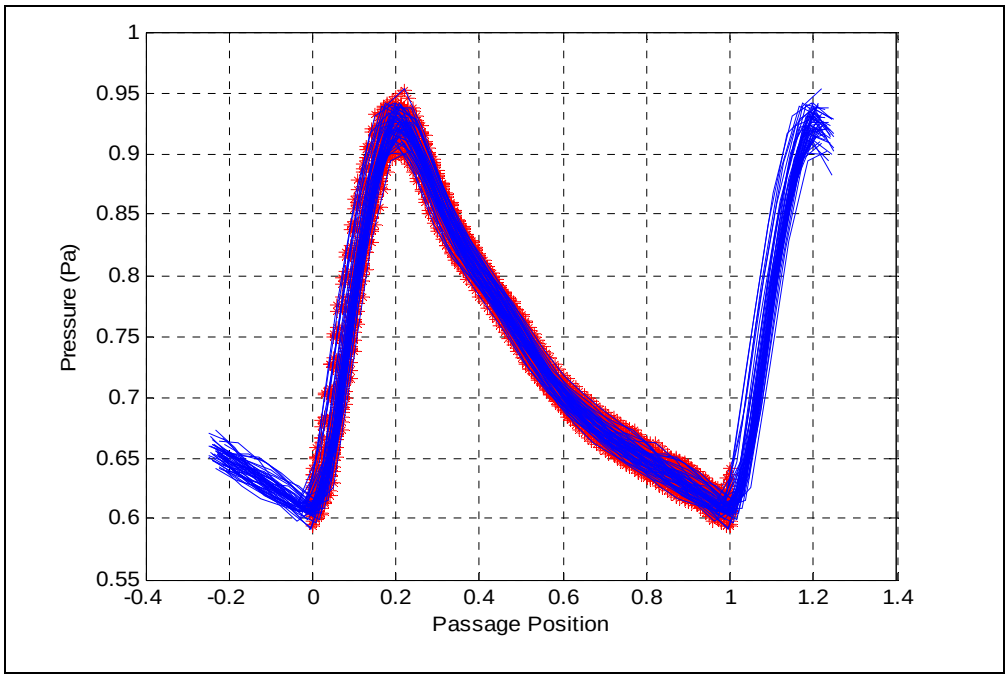


Figure 16. Kulite '-2' Pressure Passage Data

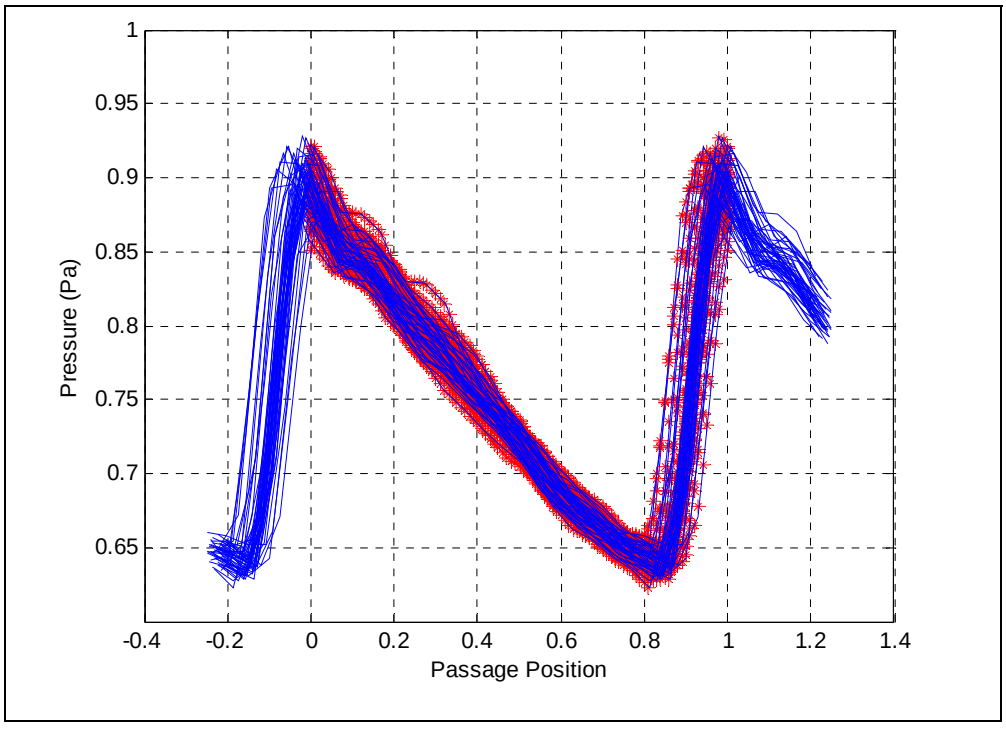


Figure 17. Kulite '-1' Pressure Passage Data

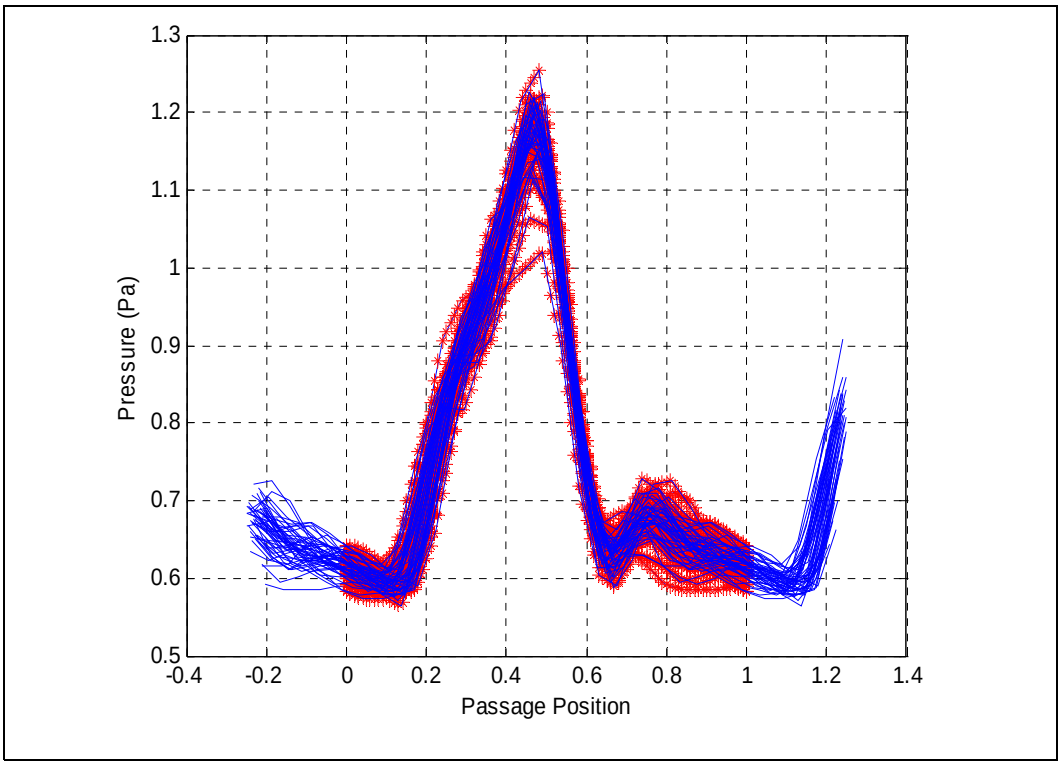


Figure 18. Kulite ‘0’ Pressure Passage Data

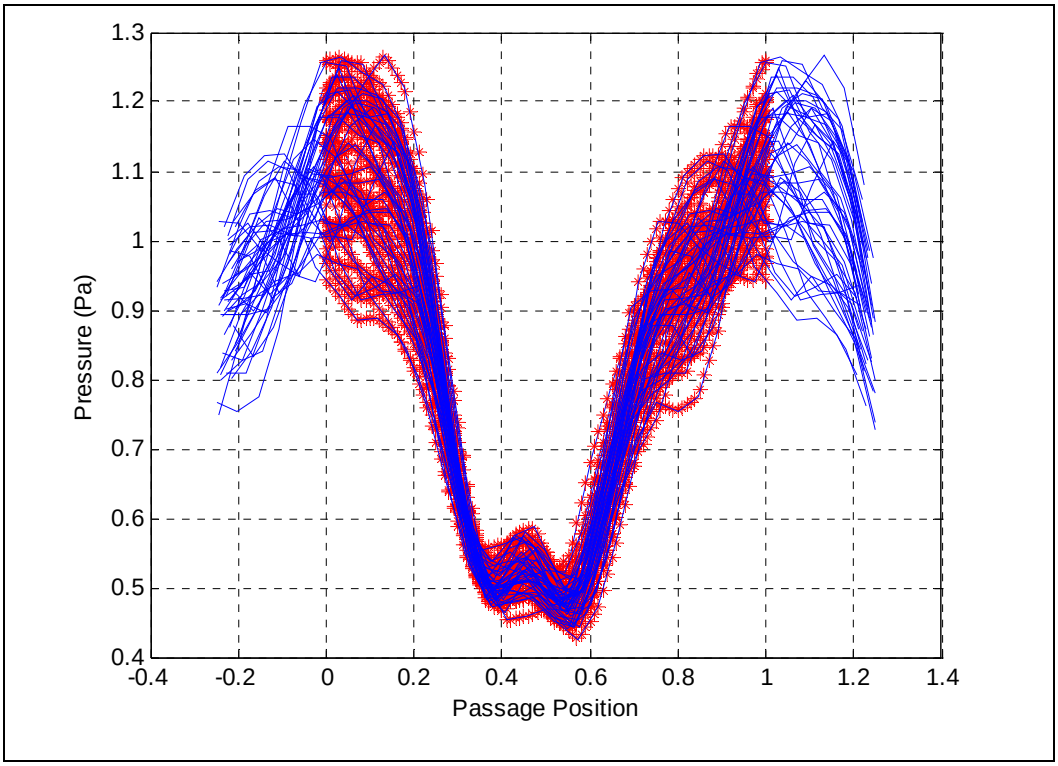


Figure 19. Kulite ‘1’ Pressure Passage Data

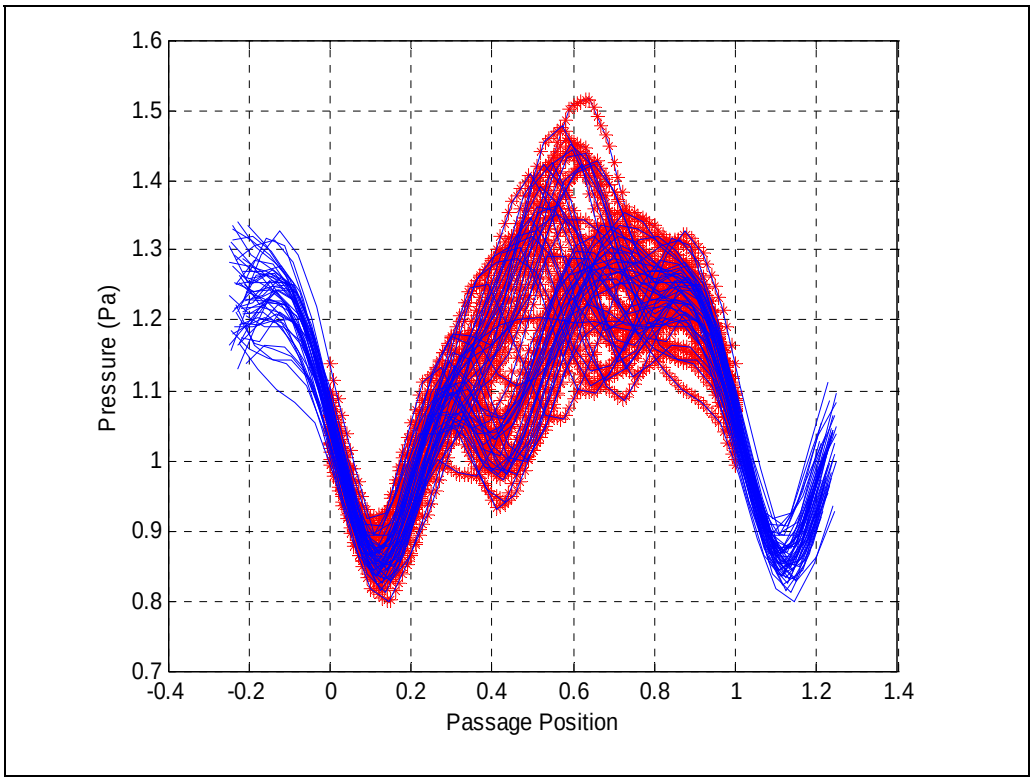


Figure 20. Kulite '2' Pressure Passage Data

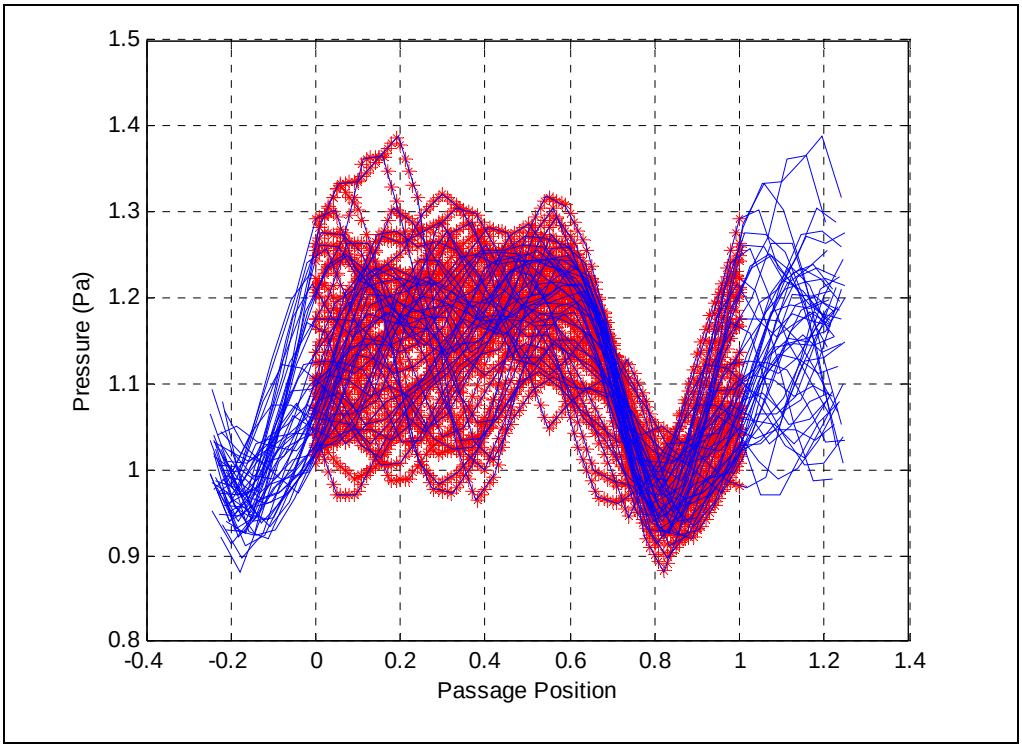


Figure 21. Kulite '3' Pressure Passage Data

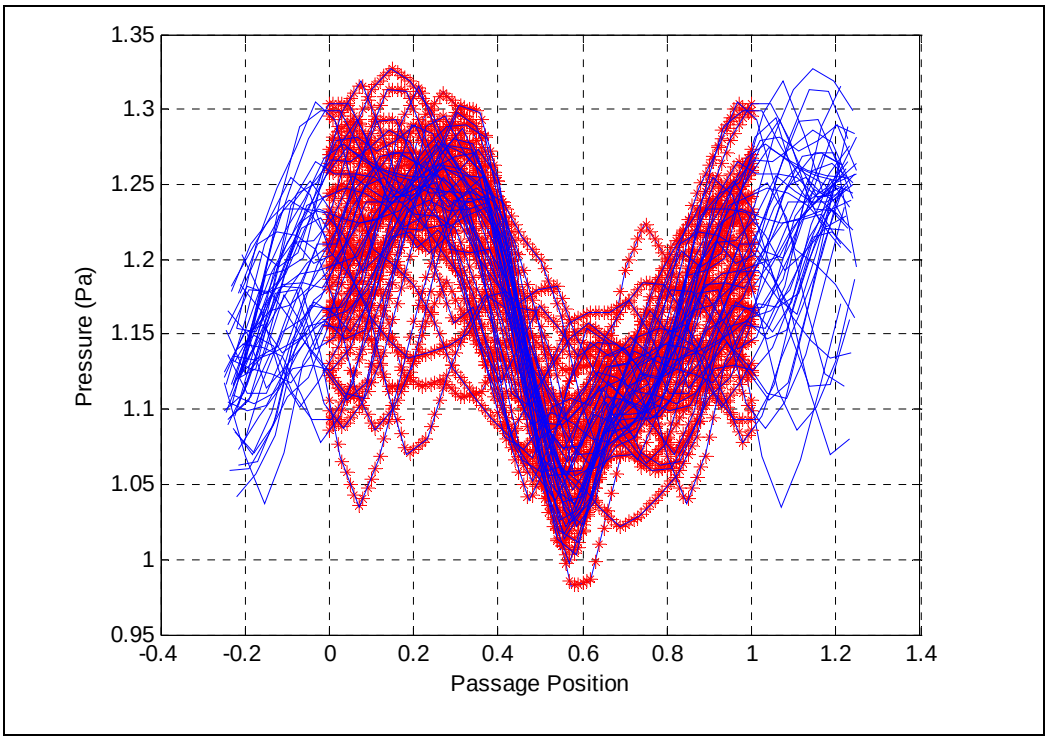


Figure 22. Kulite '4' Pressure Passage Data

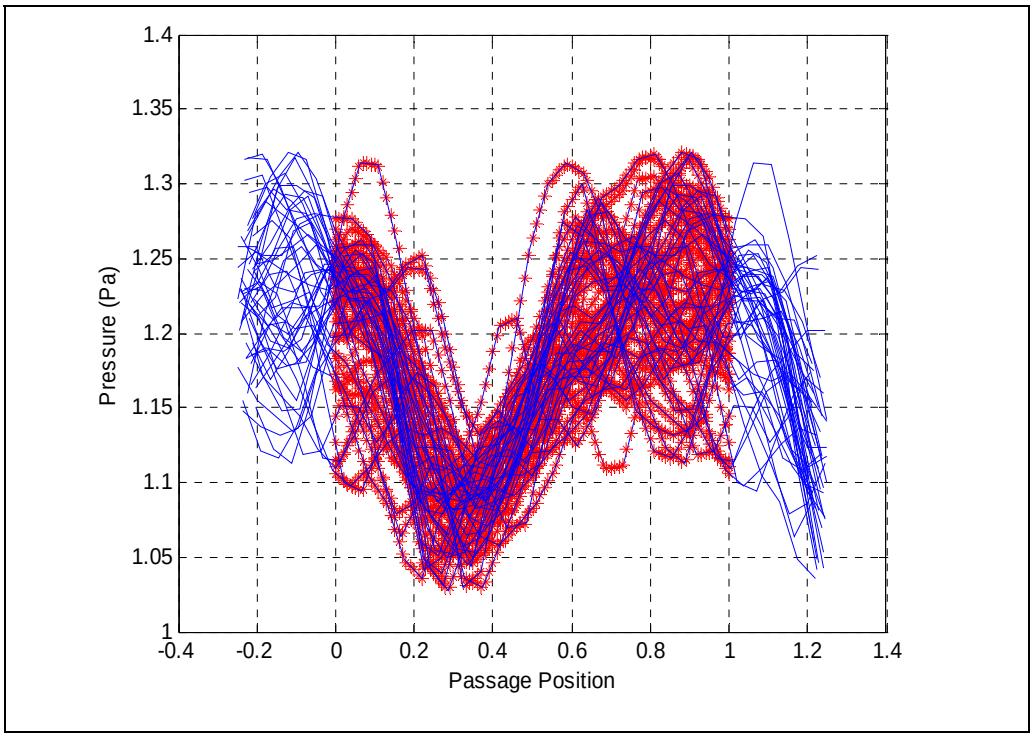


Figure 23. Kulite '5' Pressure Passage Data

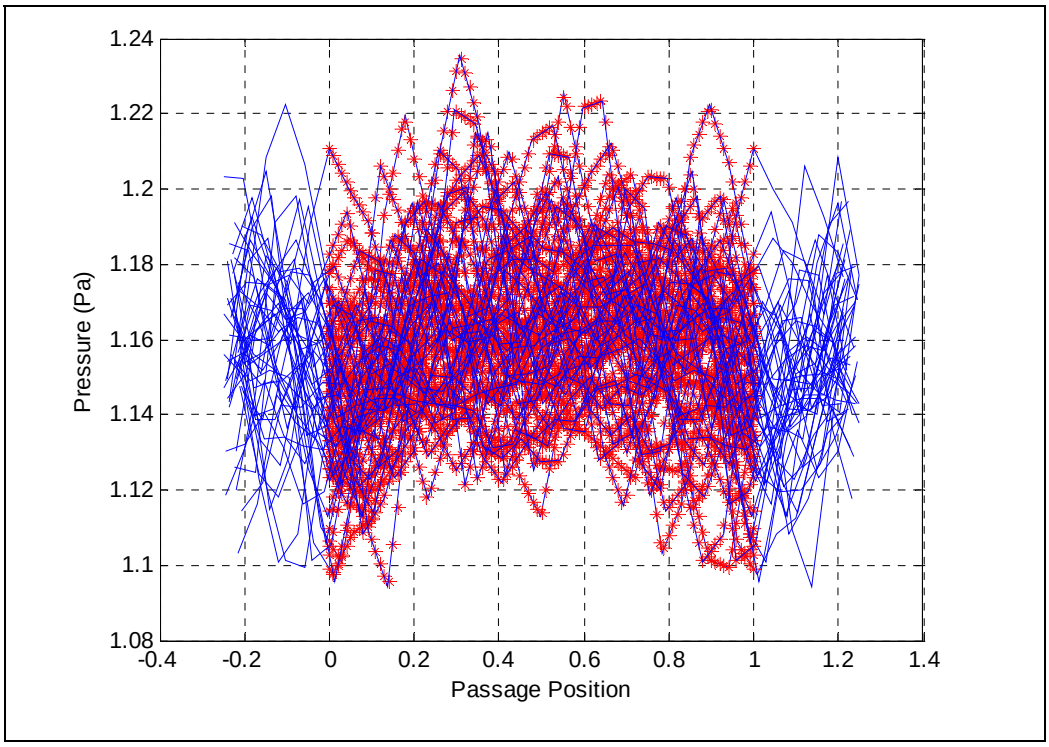


Figure 24. Kulite '7' Pressure Passage Data

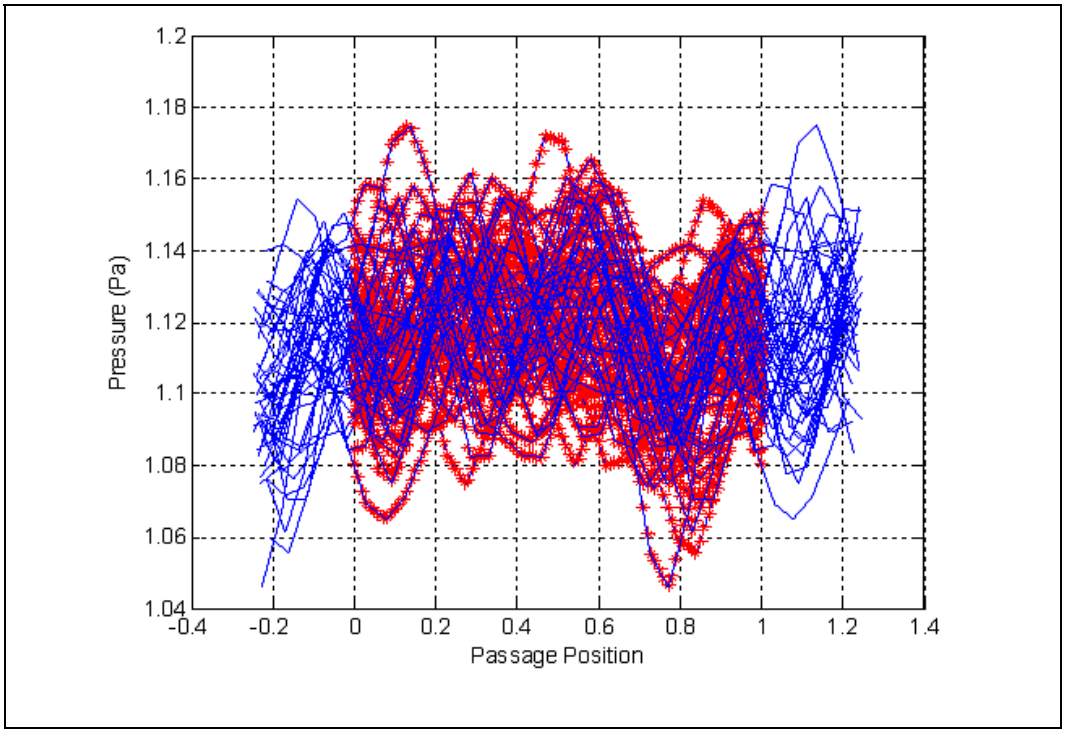


Figure 25. Kulite '8' Pressure Passage Data

C. CONTOUR PLOT GENERATION

The contouring process consisted of averaging each pressure signal at individual 100 points across the passage to produce a smooth mean signal. Each of the averages pressure signals repeated to make a string of 14 passages. The individual strings of passage signals were then matched to the Kulite probes geometry on the wall casing as shown in Figure 26.

Two interpolations were conducted, one along the oblique shock wave upstream of the rotor leading edge and the other along the blade passage downstream of the leading edge. The first three Kulite signal passages were interpolated along the oblique shock wave as shown in Figure 27. The last seven Kulite signals were interpolated along the blade as shown in Figure 29. These interpolations were done to better represent the shock interaction with respect to the rotor blade. The interpolation added 75 pressure traces to the acquired data set This allowed the contour plot function in MATLAB to smoothly join the pressure isolines in the blade and shock angle directions.

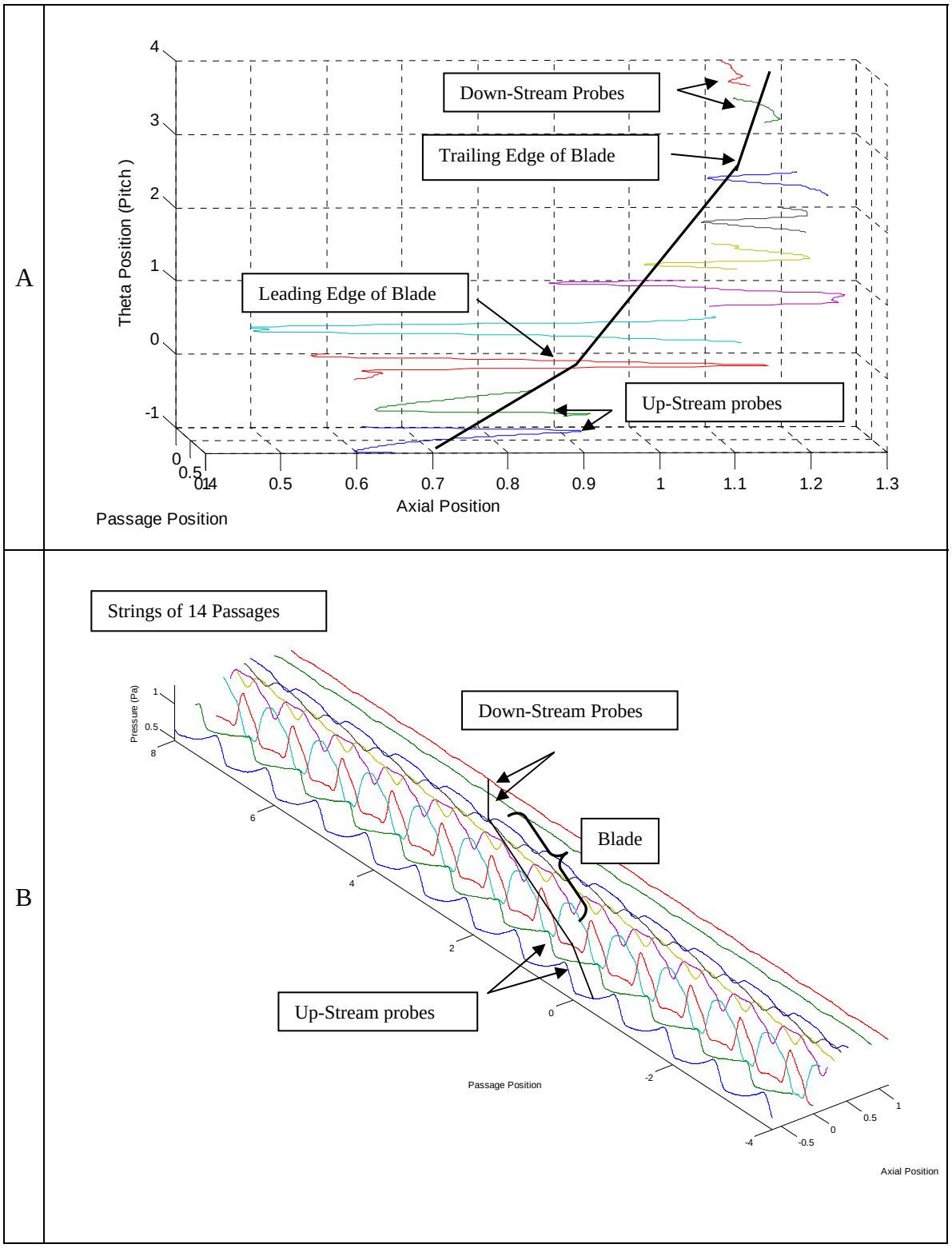


Figure 26. Passage Signal Strings Matching Kulite Geometry on Wall Casing (refer to Figure 10)

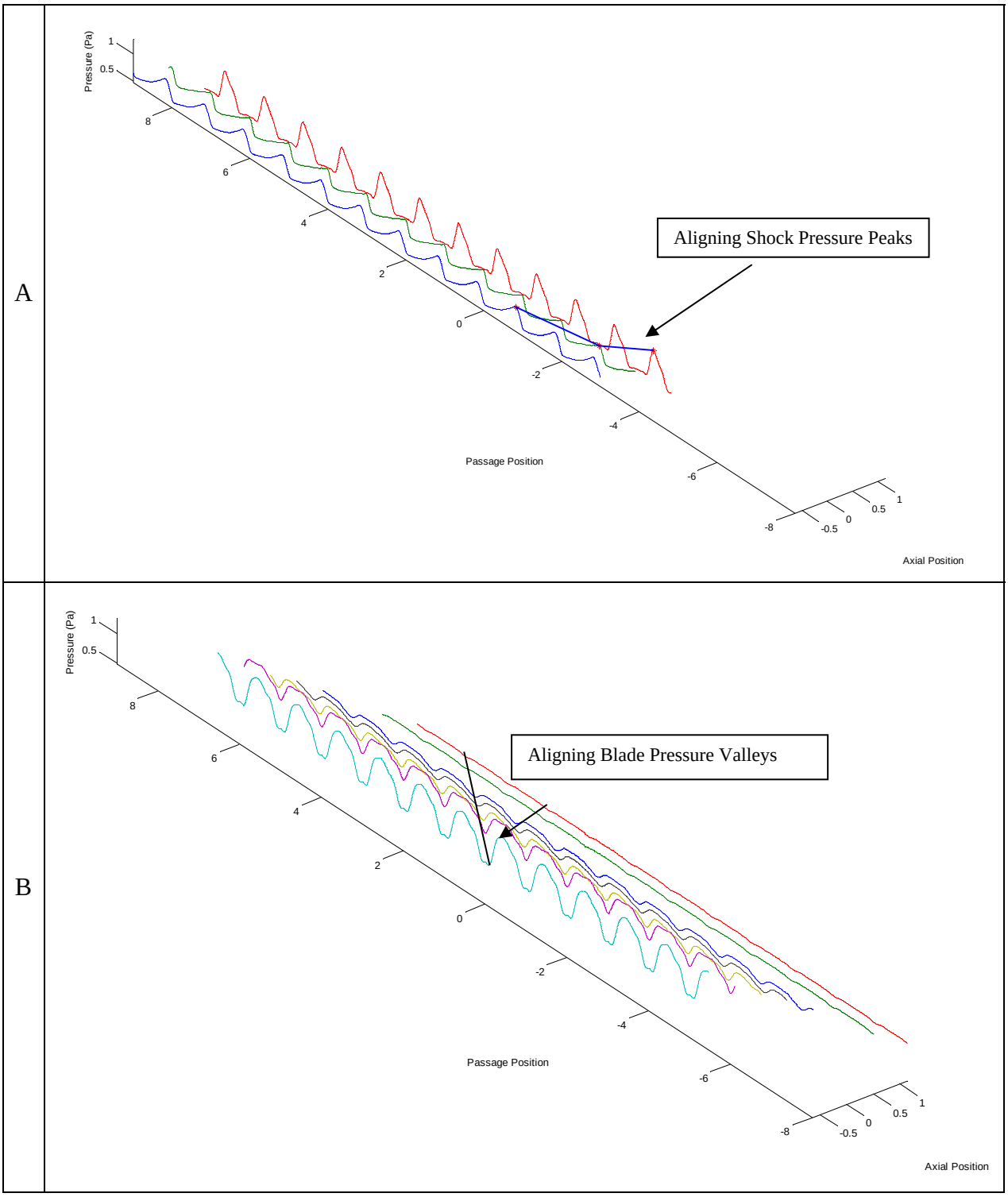


Figure 27. Interpolation Along the Shock Angle

1. Adding Mean Passages

A problem was encountered when the mean of each passage was repeated with itself as shown in Figure 28 (A). The double or repeated point created a discontinuity issue that was not compatible when using the contour plot function in MATLAB. This issue was avoided by creating an algorithm that ignored the first point of the copied passages.

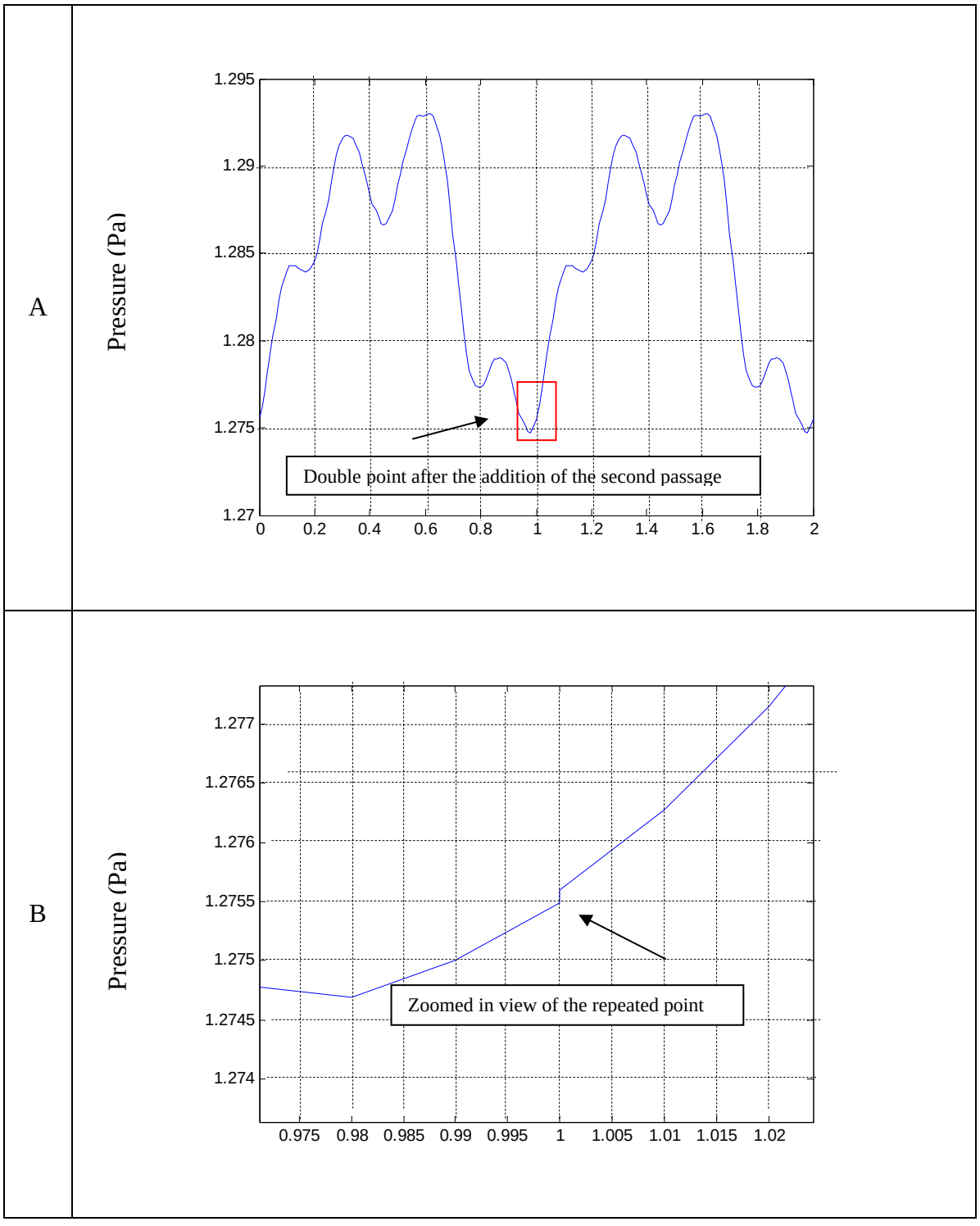


Figure 28. Adding Mean Passages to Each Other

2. Blade Interpolation

Before the interpolation along the blade occurred, the last seven signals within the passage were skewed to match the physical angle of the blade. This shift was done by aligning the probes throughout the passage using the known blade angle. Figure 29 (A) shows all 10 averaged Kulite pressure signals aligned to the passage. The blade alignment can be seen in Figure 29 (B) where the blade pressure valleys are in-line with one another and ready for interpolation. The first three lines do not align because those pressure lines represented the averages for the upstream probes for the shock wave. Figure 30 (A and B) is a three-dimensional representation of one signal passage; the dip marks the location of the blade. Figure 31 is a pressure distribution of the same transformed data as Figure 29 but using the contour function in MATLAB.

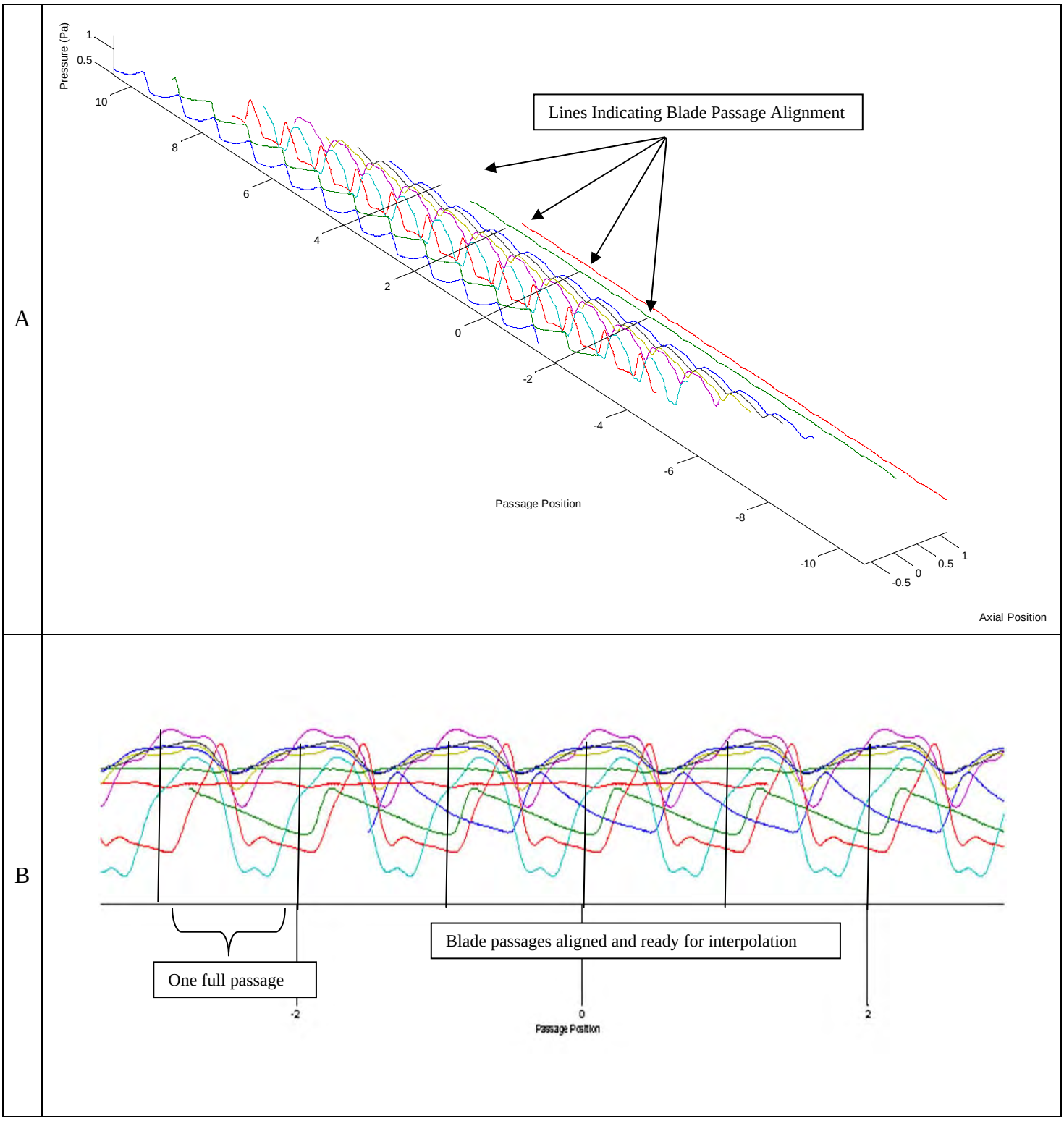


Figure 29. Passage Signal Strings Matching Kulite Geometry on Wall Casing

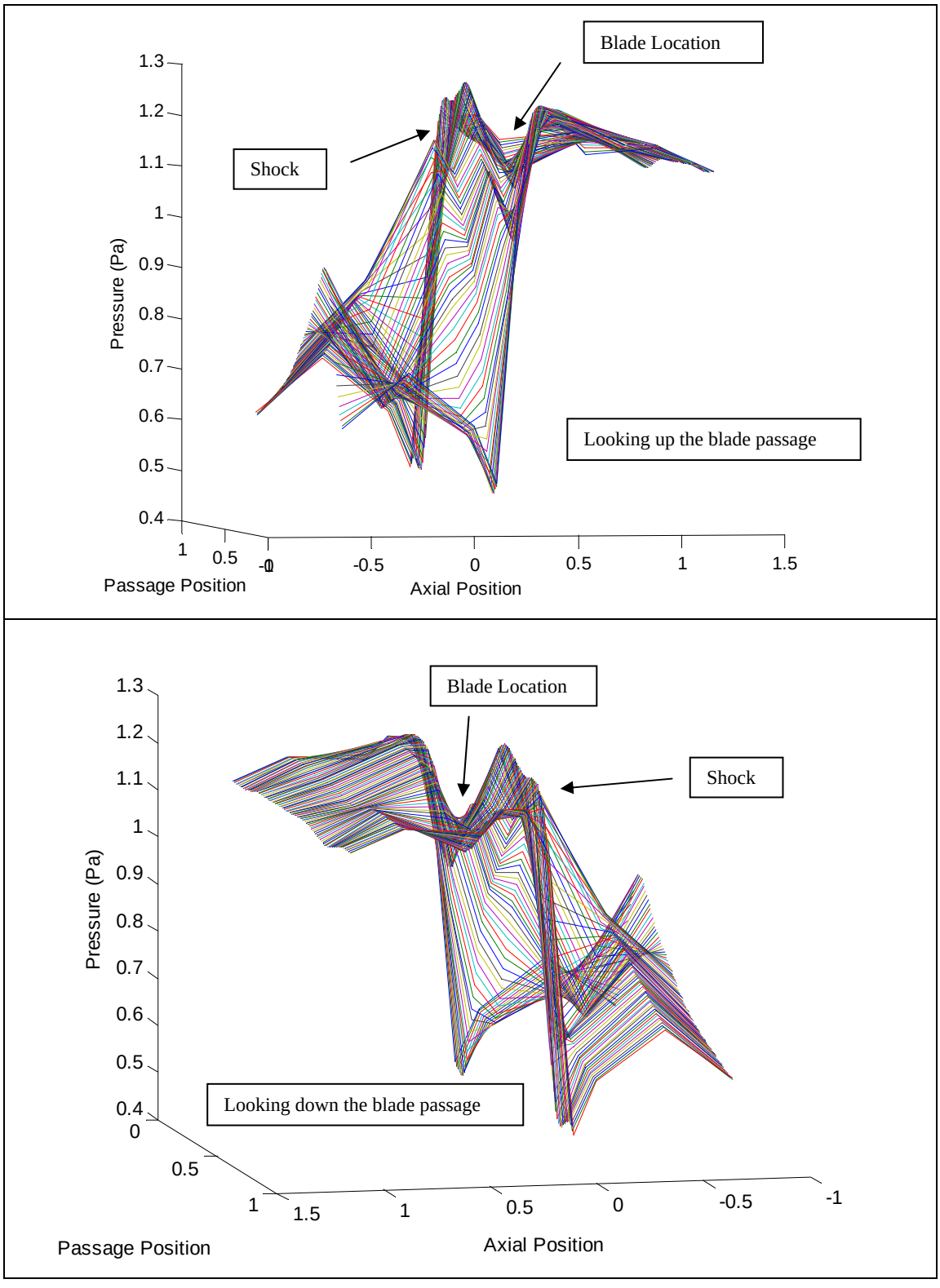


Figure 30. Three Dimensional View of the Blade Passage

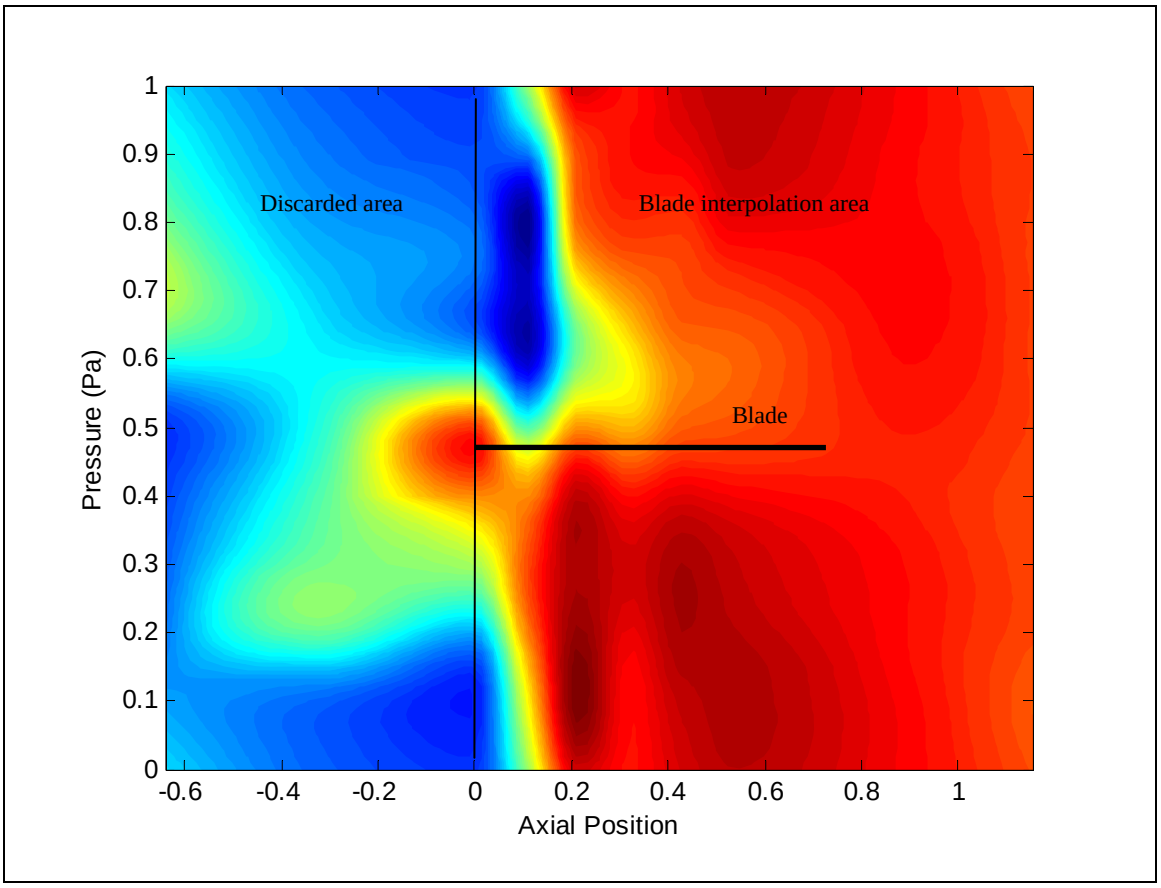


Figure 31. Pressure Contour Representation of Blade Interpolation

3. Shock Interpolation

The shock angle was calculated using shock tables for the inlet Mach number and deflection angle [27]. With a deflection angle, $\Theta=0°$, and a maximum Mach number of 1.29, a wave angle, $\beta=52°$ was calculated. Table 6 shows the calculations for all speeds investigated.

Speed	**70%**	**80%**	**85%**	**90%**
Diameter (in)	11.3	11.3	11.3	11.3
Radius (in)	0.144	0.144	0.144	0.144
Omega (rpm)	21000	24000	25500	27000
Omega (rad/sec)	2199.1	2513.3	2670.4	2827.4
Tip-speed (m/sec)	315.6	360.7	383.2	405.8
Gamma	1.4	1.4	1.4	1.4
*R (J/kg*K)*	287	287	287	287
Temp (K)	288	288	288	288
Sonic Velocity (m/sec)	340.3	340.3	340.3	340.3
Tip-Mach Number	0.93	1.06	1.12	1.19
Blade Angle (rad)	1.19	1.19	1.19	1.19
Inlet Mach Number	1.00	1.14	1.21	1.29
Deflection Angle (°)	0	0	0	0
Wave Angle, β (°)	**90**	**70**	**56**	**52**

Table 6. Shock Wave Calculations [27]

Once the shock angles were calculated, they were verified in MATLAB by finding and matching the peaks of the first three pressure lines as shown in Figure 32 (A). The shock peak angles aligned and highlighted in Figure 32 (B) are shown with the rest of the probe mean pressures.

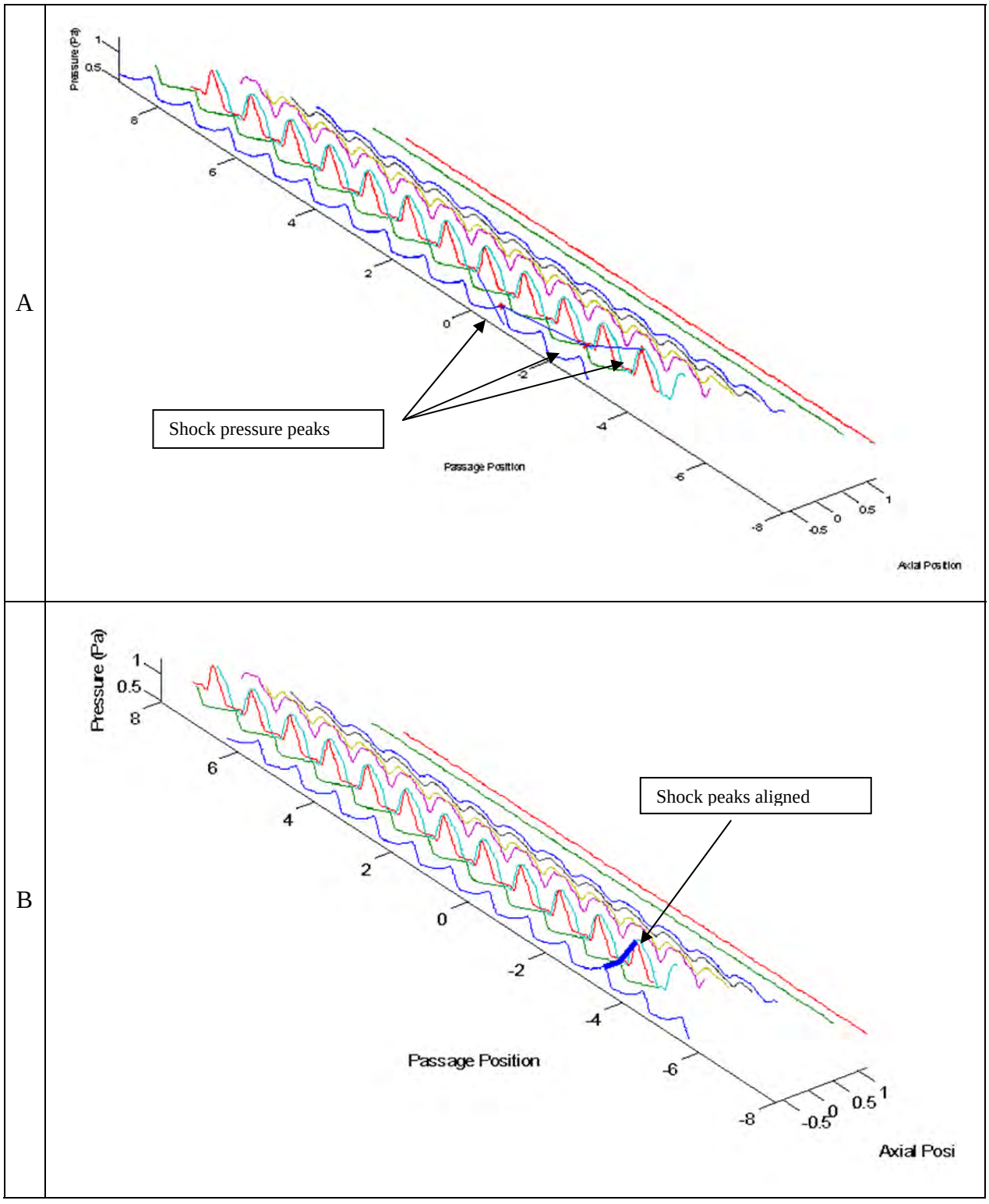


Figure 32. Passage Shock Interpolation

The following figures show the steps of how the final pressure contour plot was created step-by-step. Since a limited number of Kulite probes (10) were used, data was interpolated in the axial direction to improve the smoothness of the contour plots. Figure 32 shows the joining of the shock peaks.

A better illustration of what happened when the shock was interpolated is shown in a three-dimensional view in Figure 33 (A and B). Figure 33 illustrates the grouping of the shock angles needed for interpolation. Figure 34 (A and B) illustrates the same data represented in a pressure contour distribution map. Figure 34 (A) shows the angle that the shock was shifted down by, with the shock interpolation following in Figure 34 (B).

Figure 35 (A and B) shows a three-dimensional representation of how swept the shock is at 90% speed prior to stall. Only the shock angle portion of this figure was used for the final contour plot. Figure 36 (A and B) show the shock and the blade interpolations, the figure is shifted to match the Kulite probe locations on the wall casing. Figure 36 (A) shows the shock angle; the data to the right of the line is discarded. Figure 36 (B) shows the blade position; the data to the left of the line is discarded. Figure 37 joins the shock and blade angles in Figure 36 (A and B) at the correct shock angle and location.

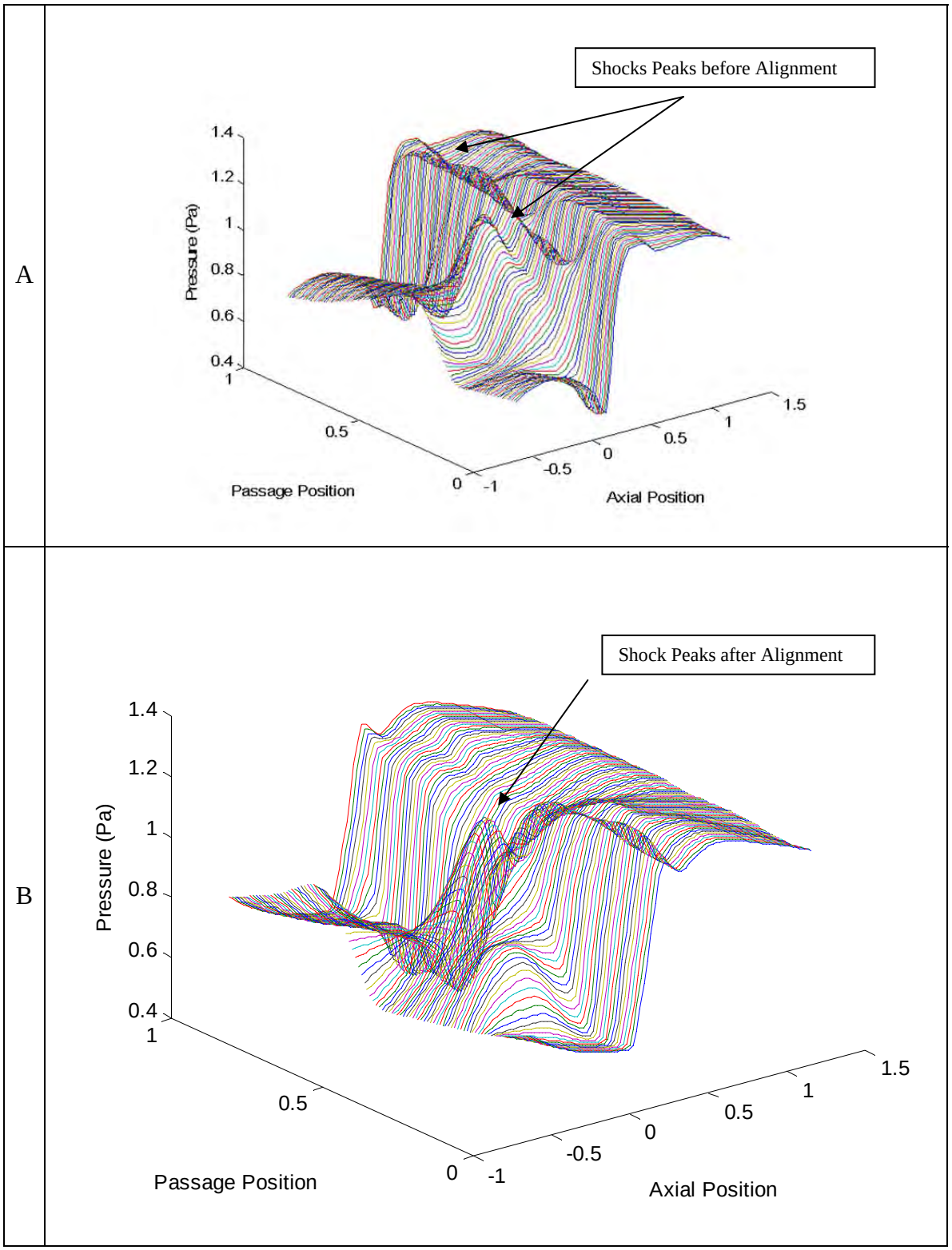


Figure 33. Shock Interpolation Three Dimensional View

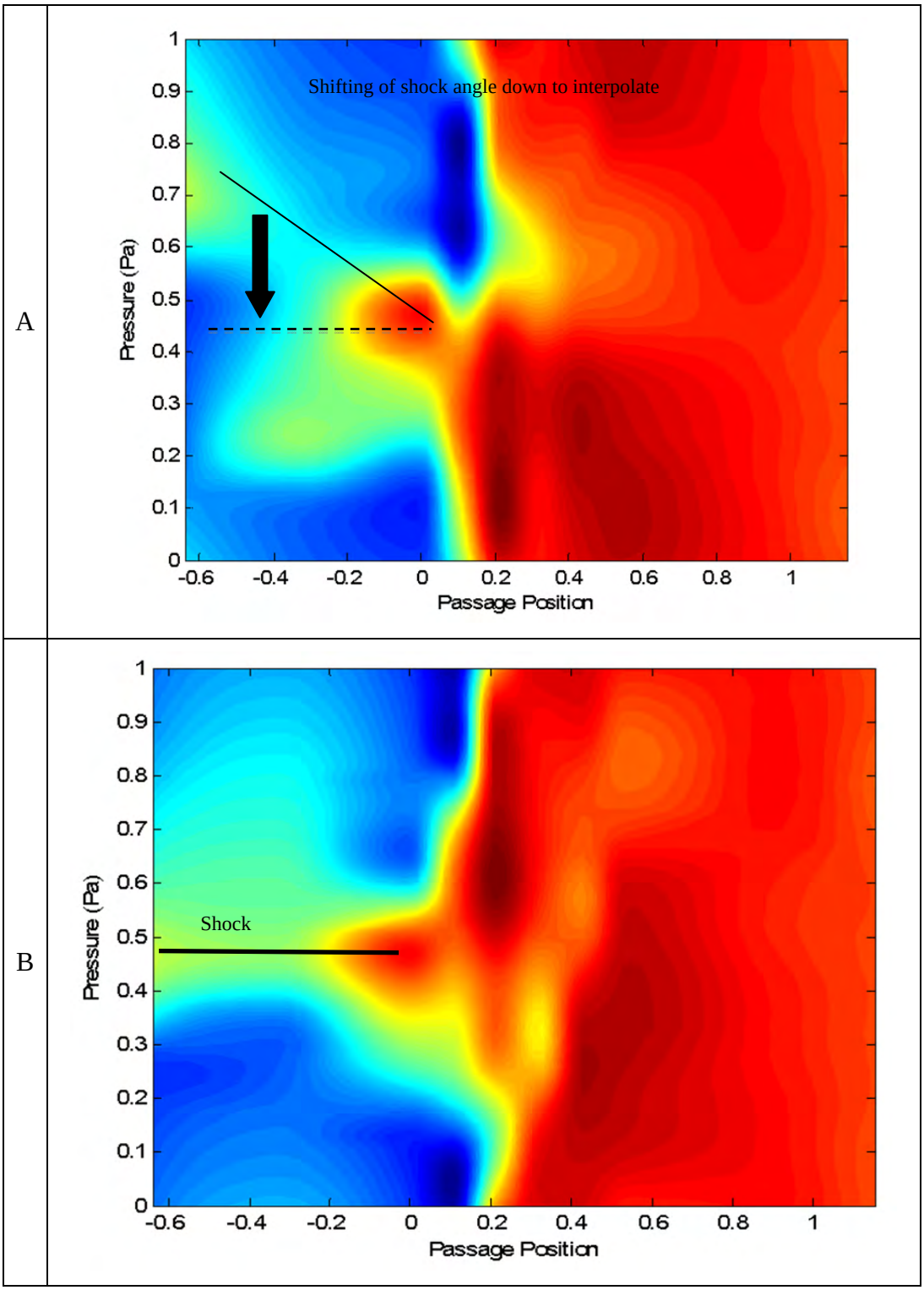
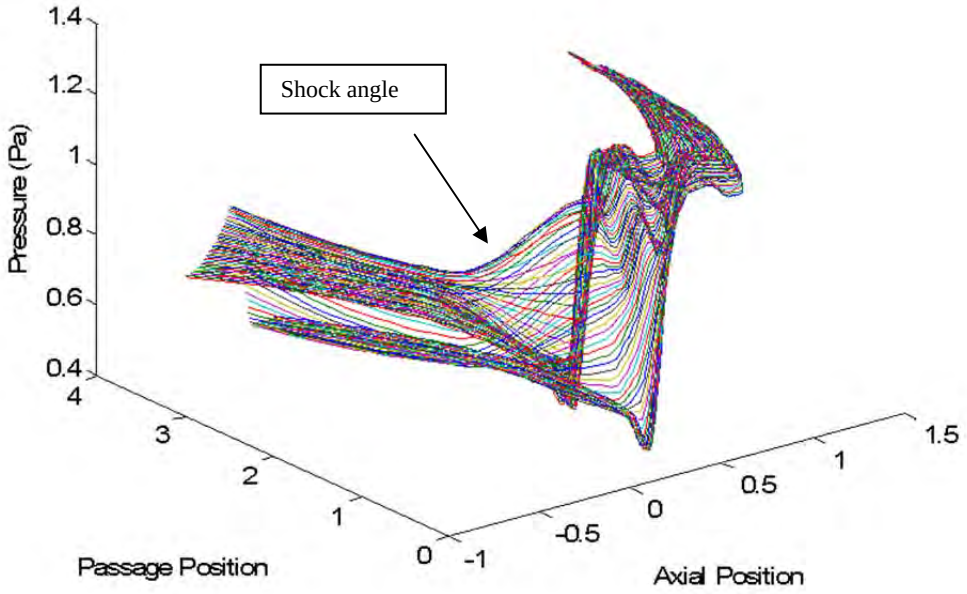


Figure 34. Contour Plot of Shock Passage Interpolation

A



B

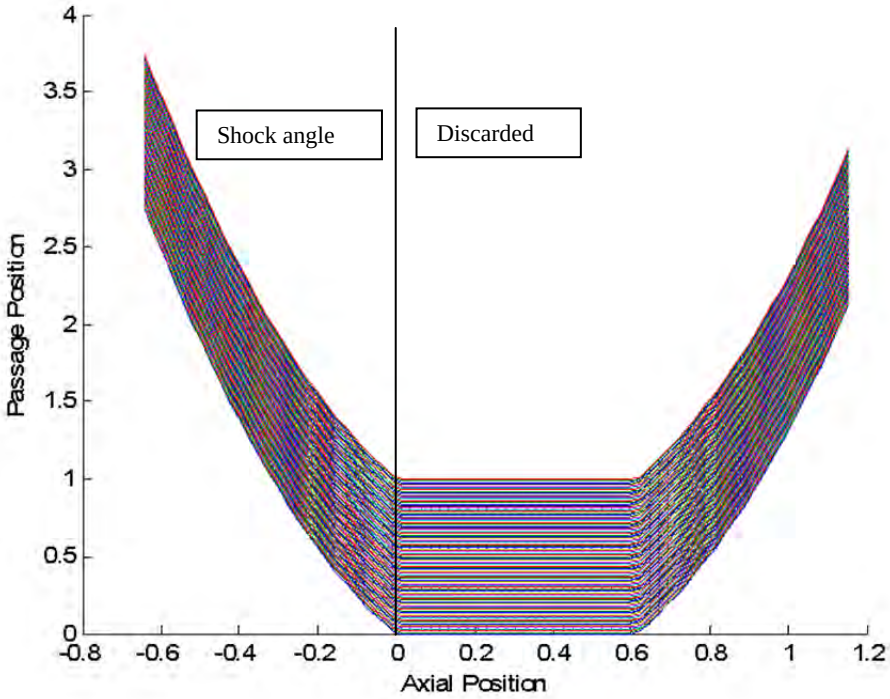
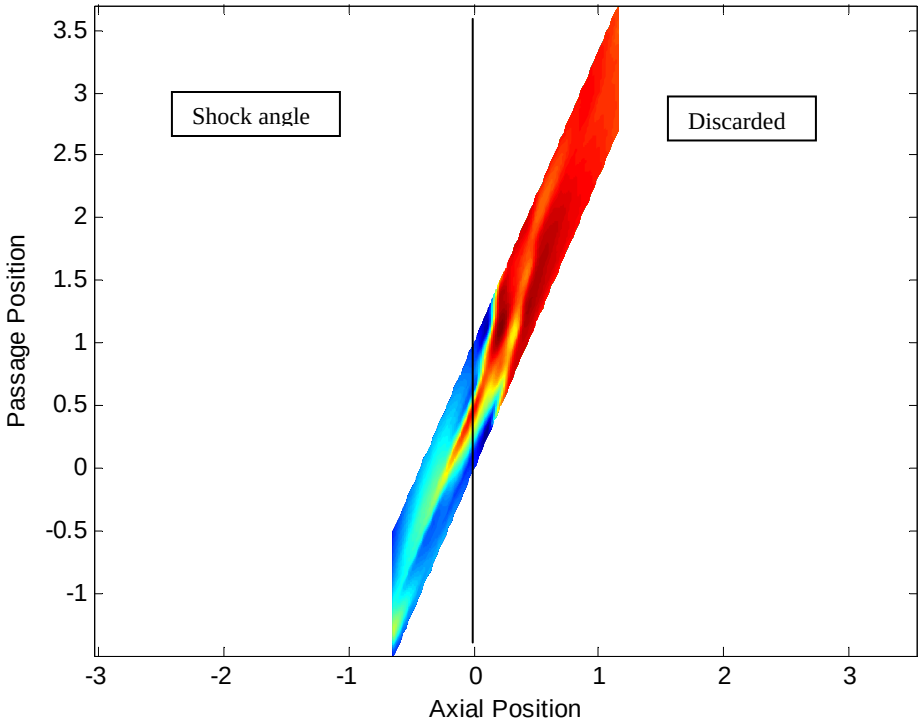


Figure 35. Shock Wave Angle

A



B

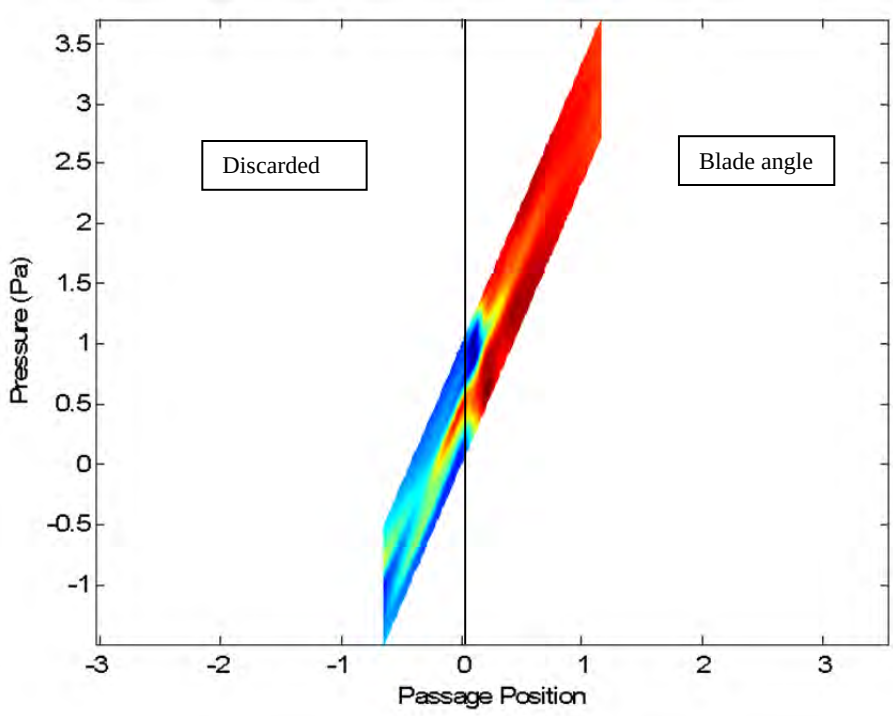


Figure 36. Adding Shock and Blade Angles

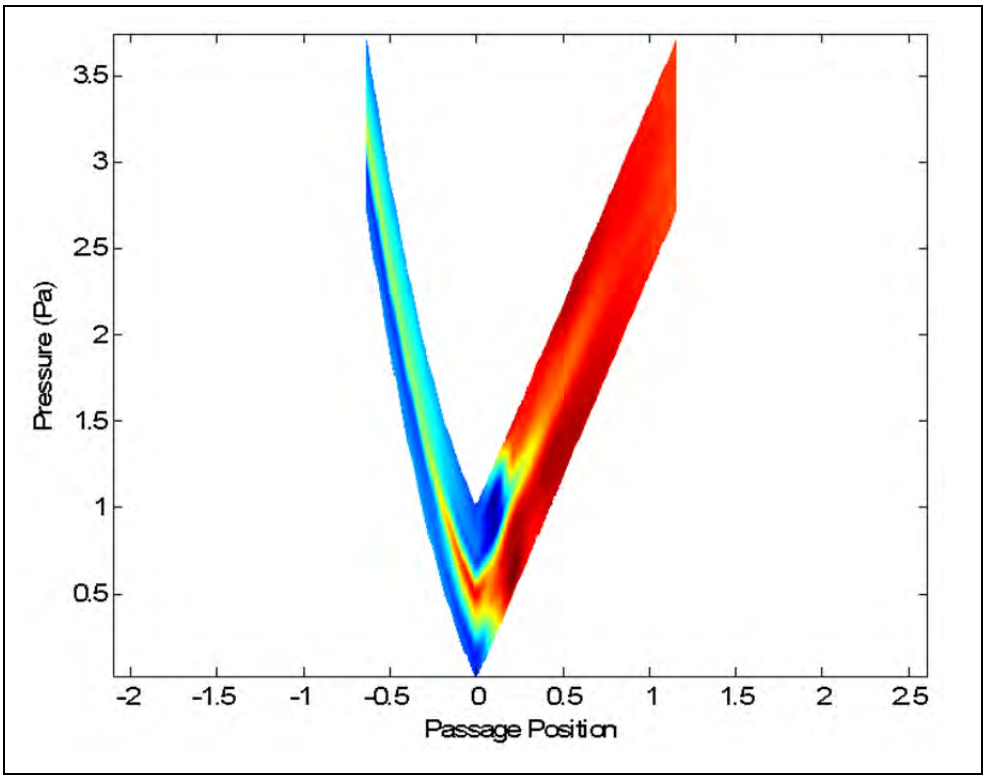


Figure 37. Shock and Blade Angle Added Together

The final pressure contour map is shown in Figure 38. The black angled lines denote the position of the blade; this helps to highlight the shock angle with respect to the physical blade. The colors in the contour map represent pressure values. High pressure is represented in red and low pressure is represented in blue. Regions of low pressure usually indicate supersonic flow, while regions of high pressure indicate subsonic flow relative to the blade. The shock wave ahead of the blade at 90% speed is oblique (~52°). The tip-leakage is also evident as is distorts the passage shock between the blades. Pressure distribution maps for 70%, 80%, and 85% were also produced for varying flow conditions and are found in Appendix B.

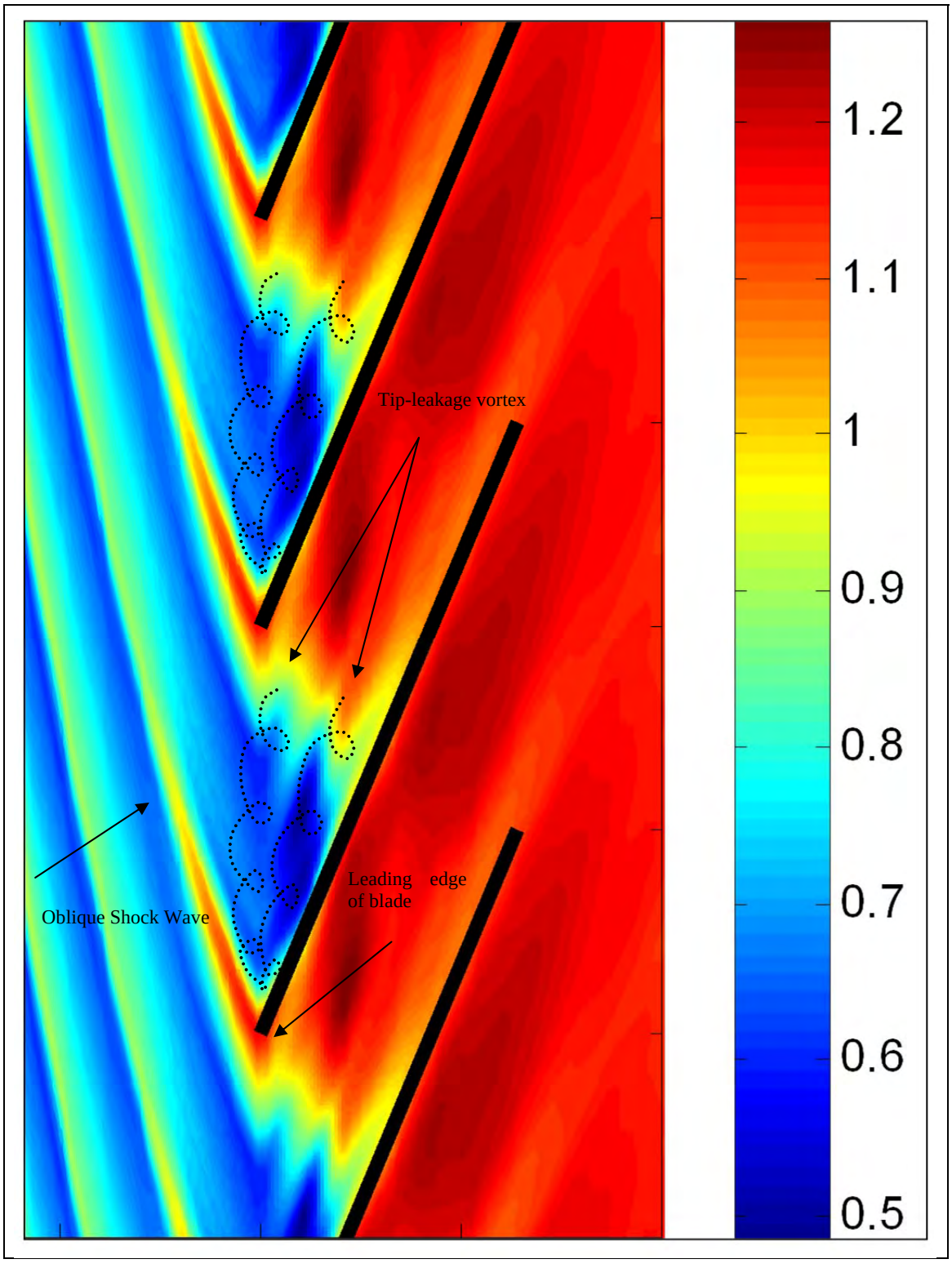


Figure 38. Near-Stall 90% Speed Pressure Contour Map

D. FFT DATA ANALYSIS

The FFT data analysis consisted of interrogating each of the 100 points for all ten Kulite signal passages that formed the contour plot. This process located the maximum amplitudes of non-periodic flow phenomena and their corresponding frequencies. Each data point contained 17,000 sub-points, which were transformed to the frequency domain using an FFT. A total of 1,000 FFT's were generated and analyzed. Figure 39 illustrates the location of the 10 Kulite probe lines and the location of two sub-points used to illustrate the method to find the passage distribution.

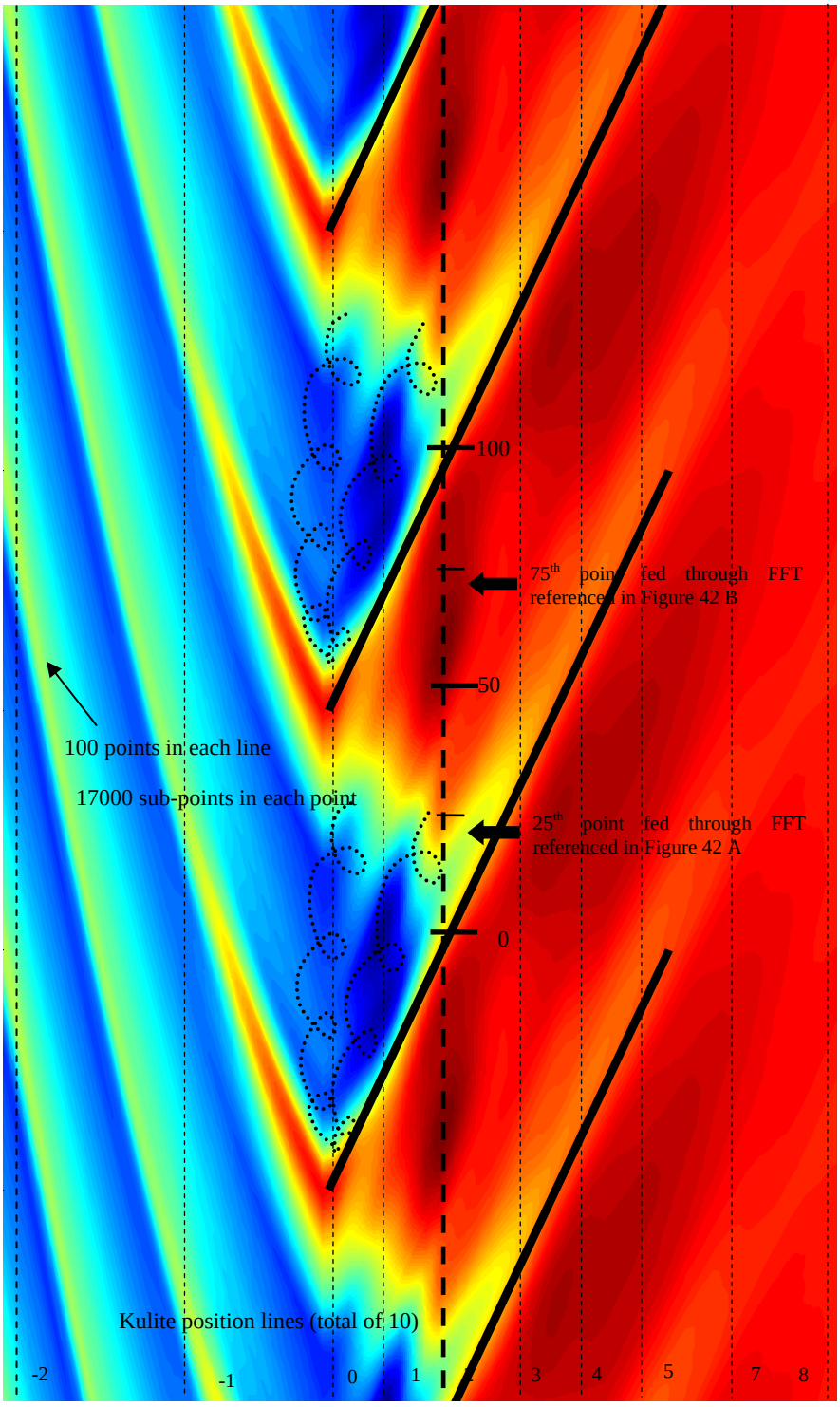


Figure 39. Kulite Position Lines, Points, and Sub-Points

Figure 40 (A) shows the superimposed pressure signal for Kulite 2. The data analyzed to produce Figure 40 is located along near the center of the blade. Figure 40 (B) is a zoomed view of the strings of data points within the superimposed pressure signal.

Figure 41 (A) shows the signal from a sub-point for 900 passages of data in blue. Overlaid is the signal in red passed through a Blackman window to avoid convolution errors. Figure 41 (B) shows the full spectrum of frequencies that were initially investigated. Multiples of the Once-Per-Revolution (OPR) frequency was observed, but frequencies after the first once-per-rev were omitted in this research. Figure 41 (B) shows the range of frequencies investigated. The small frequency after the 7^{th} once-per-rev was noticed but not investigated.

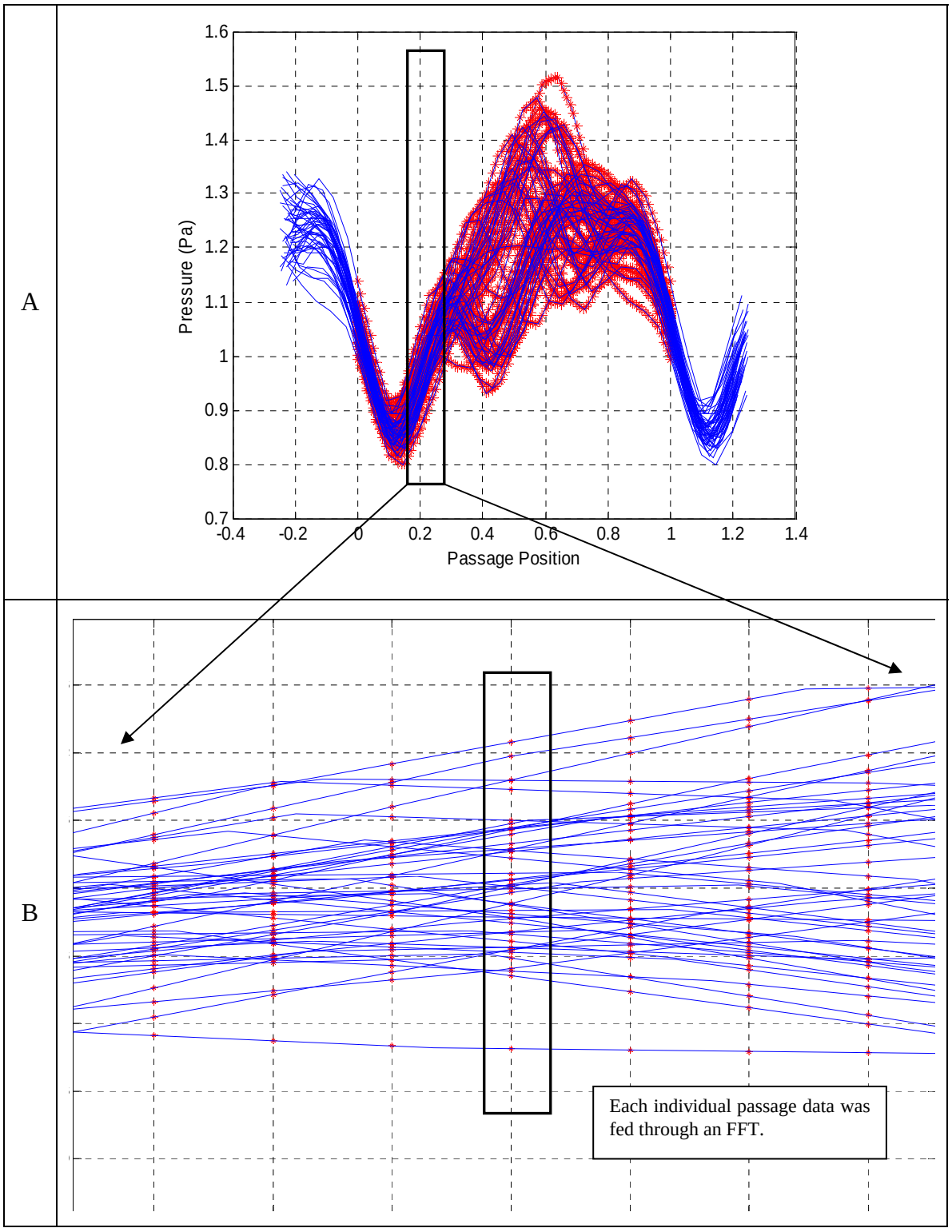


Figure 40. Close-up view of Kulite '2' Data Passage

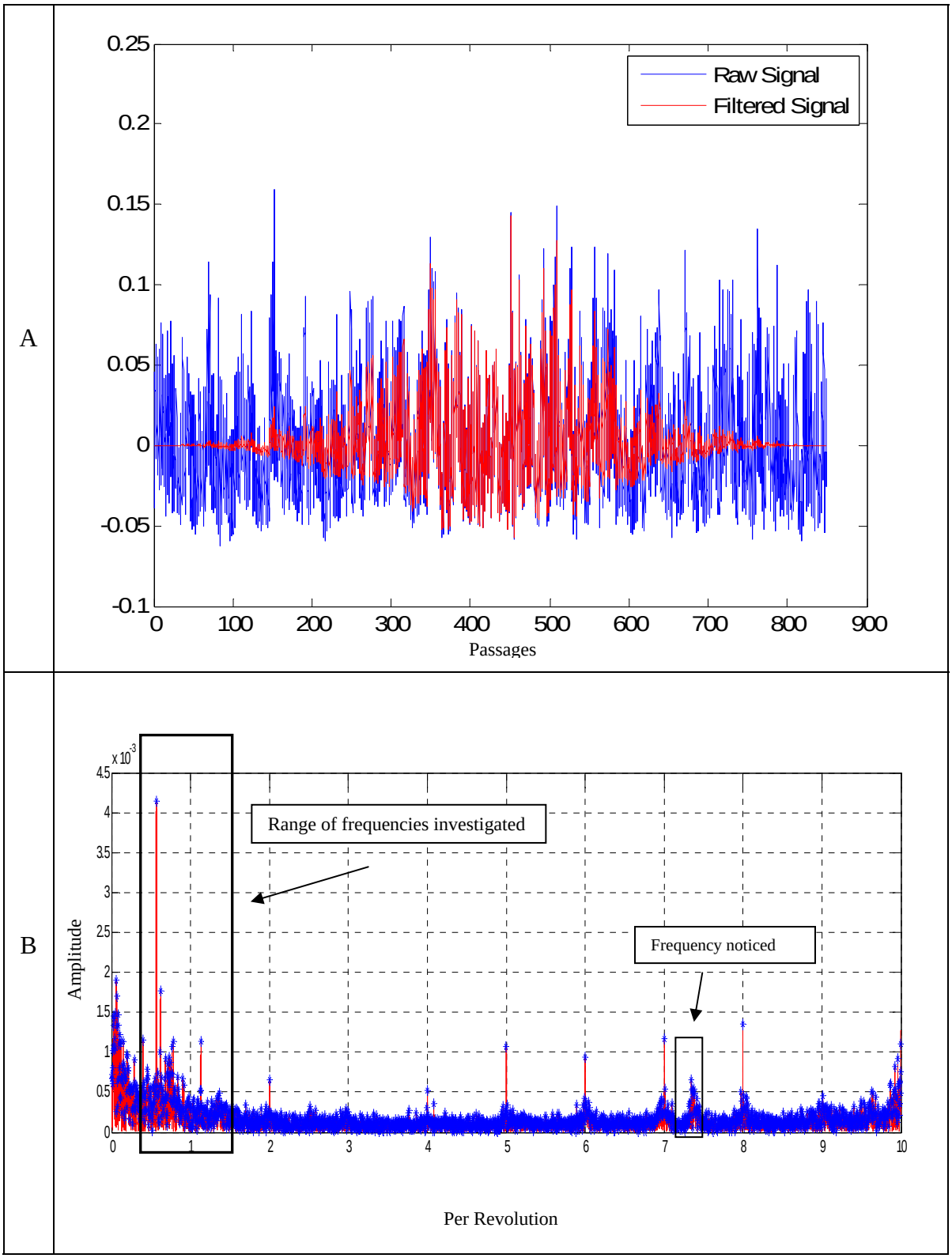


Figure 41. Blackman Window and FFT of Raw Signal

Figure 42 is the output of a signal transformed to the frequency domain using an FFT. Figure 42 (A) is a snapshot of the FFT of the 25th sub-point of Kulite probe 2. Figure 42 (B) shows the 75th sub-point of Kulite probe 2 (refer to Figure 39). Both the once-per-rev frequency and the non-periodic frequency (56.5% of BPF) are dominant in the 25th sub-point frequency analysis shown in Figure 42 (A). Figure 42 (B) illustrates that the non-periodic disturbance, 56.5% of BPF, is solely dominant in the 25th sub-point frequency analysis of Kulite probe 2

The blade passing frequency and the non-periodic disturbance was also noticed in research from Gannon et al. [28], Hah et al. [4], and Yamada et al. [23]; however an amplitude distribution plot of those frequencies of interest has not been produced. The relationship between the two frequencies of interest is still being further investigated by mapping the phase shift of these frequencies but is beyond the scope of this thesis. A preliminary contour map of the phase for each frequency was produced however; it requires more attention.

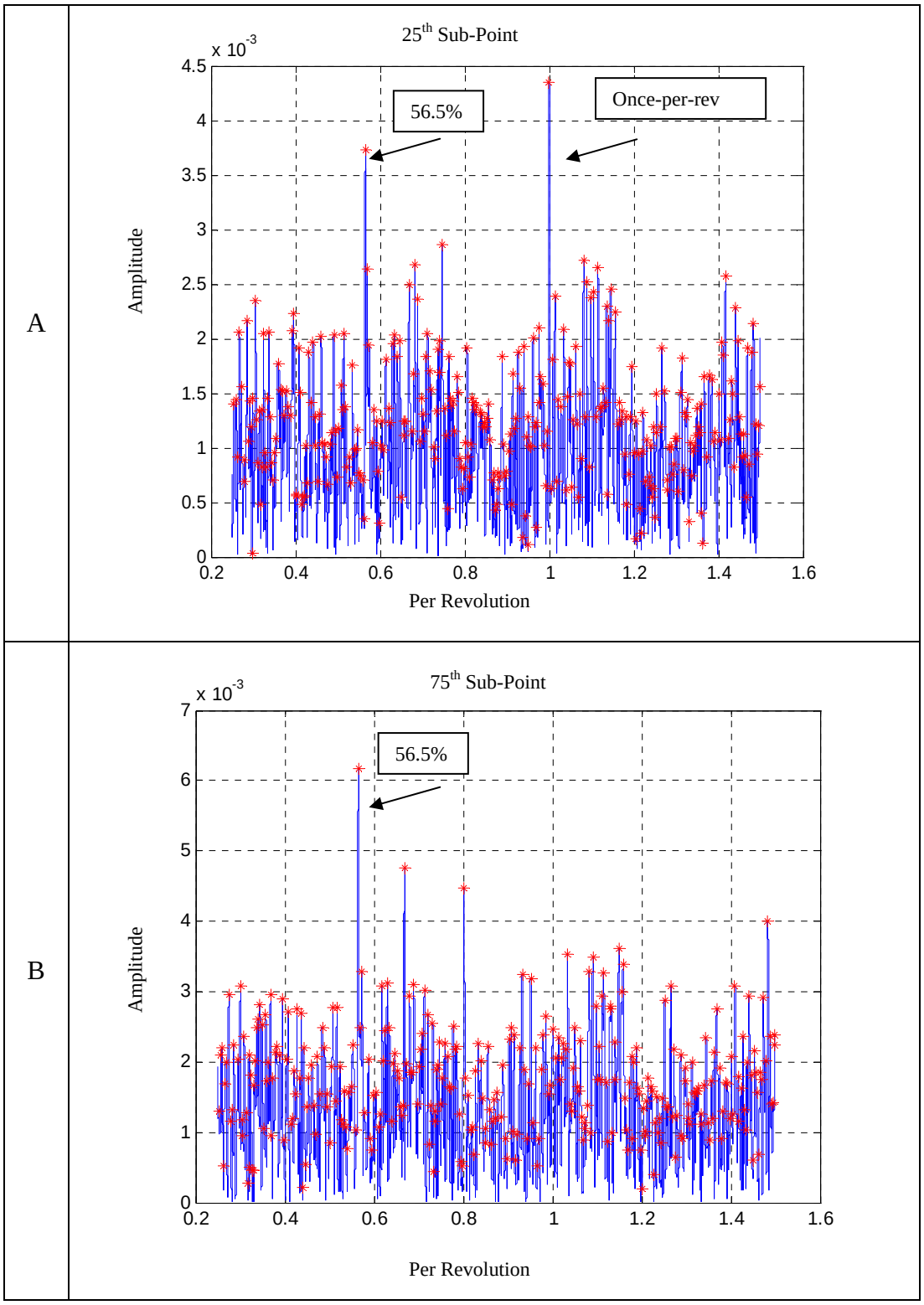


Figure 42. FFT for the 25th and 75th Sub-Points within the Passage of Kulite Probe '2' (refer to Figure 39)

E. FFT CONTOUR PLOT GENERATION

Figure 43 and 44 are the amplitude distributions of the frequencies of interest for both the periodic and non-periodic disturbance. These plots were created using the same steps covered in section C of Chapter 3. Using the linear scale of these disturbances the naked eye could discern where each frequency prevailed throughout the passage. Figure 43 shows that the blade passing frequency is prevalent along the blade, and most dominate at the leading edge of the blade. Figure 44 shows that the non-periodic frequency (56.5% of BPF) is prevalent along the shock wave.

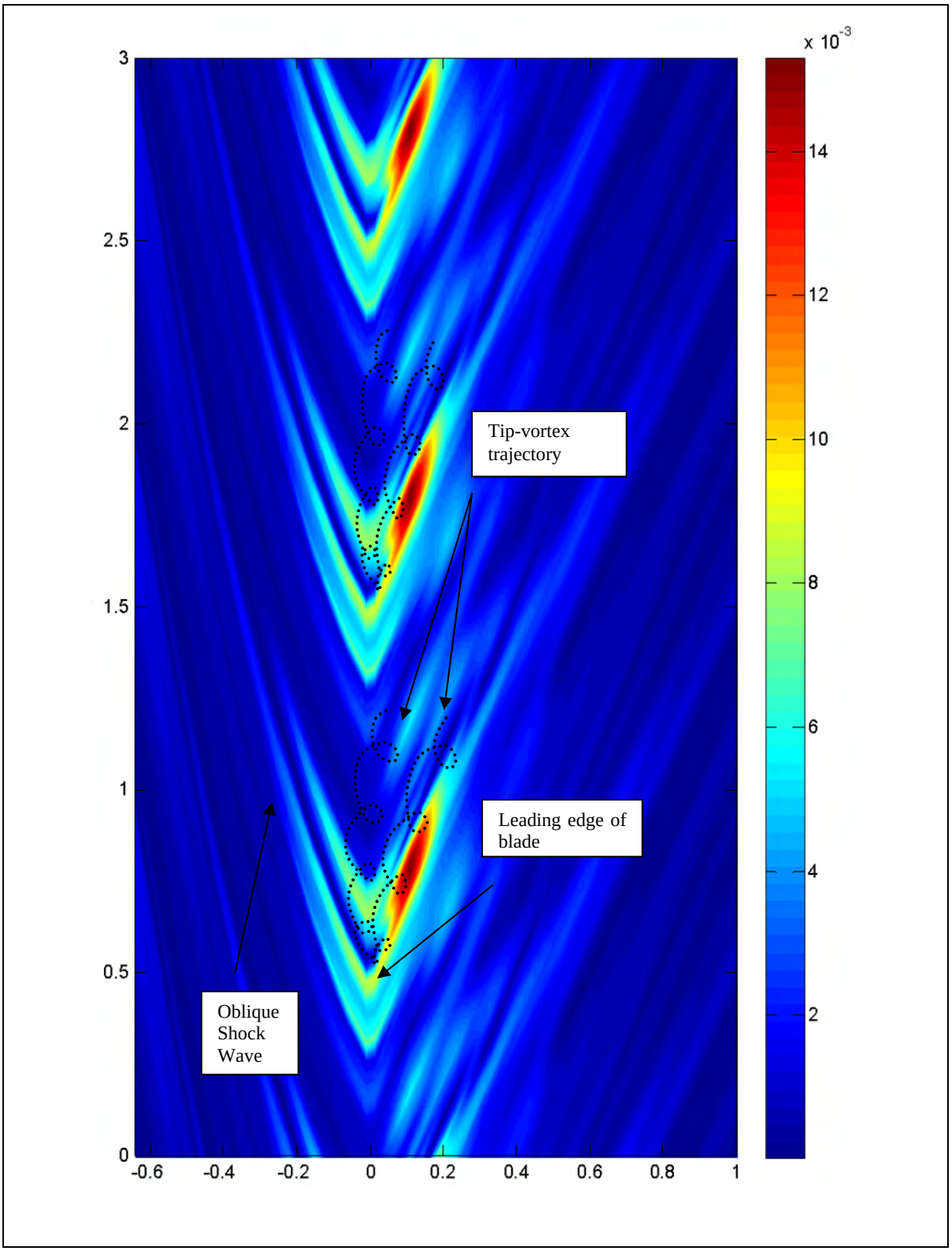


Figure 43. Contour OPR FFT

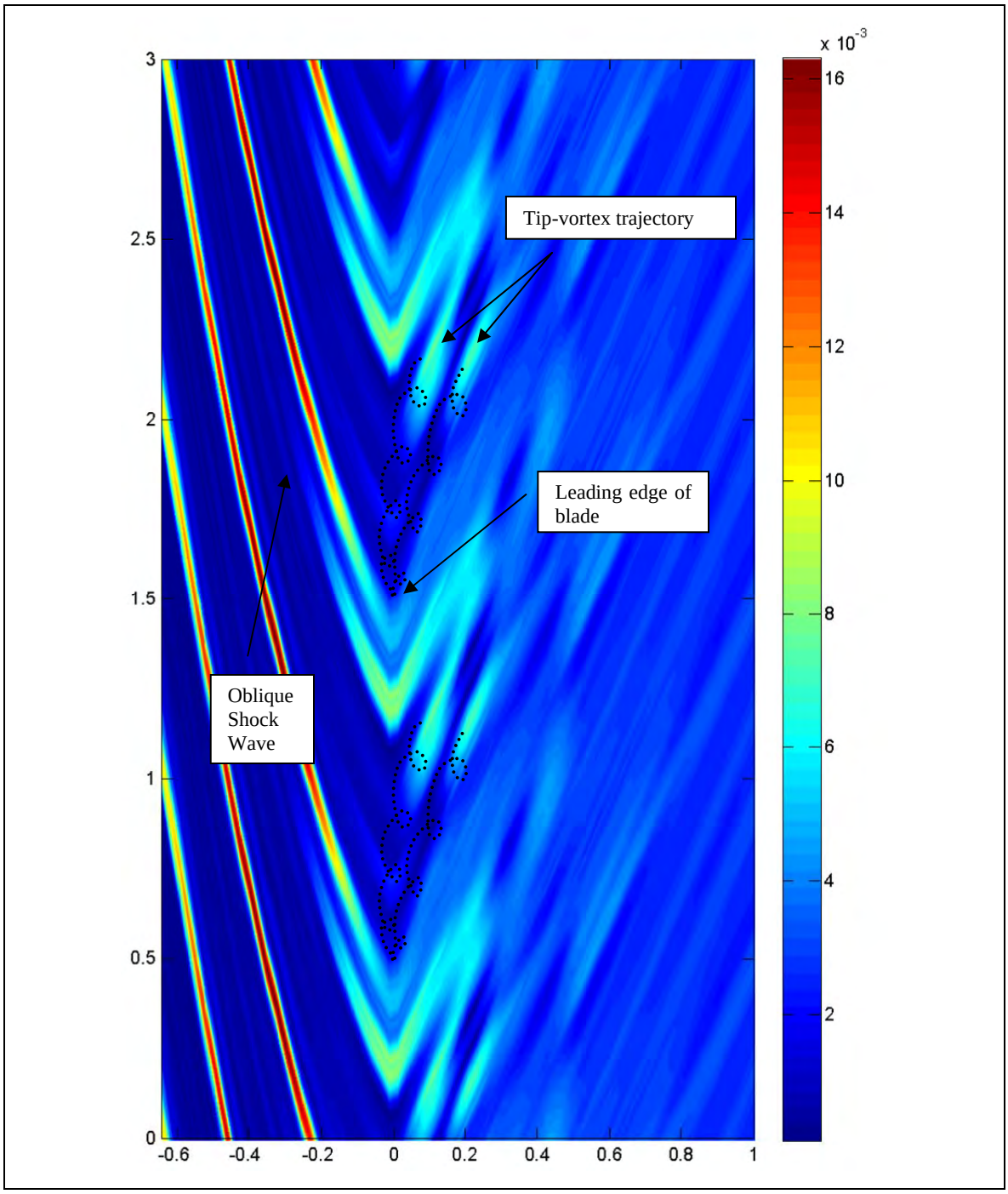


Figure 44. Contour Plot of the 56.5% OPR FFT

IV. RESULTS AND ANALYSIS

Full pressure distribution maps are shown for 70%, 80%, 85% and 90% design speeds. Pressure distribution maps for peak efficiency to pre-stall operations are listed in Appendix B.

A. PRESSURE MAPS FOR 70%, 80%, AND 85% DESIGN SPEEDS

Figures 45- 48 show the obtained projection of the pressure contours by the rotor casing for 70%, 80%, 85%, and 90% design speeds. The normal shock, passage shock, and its tip vortex trajectory are illustrated for all speeds.

As the speed increases the normal shock wave turns into an oblique shock wave as shown in Figures 46-48 for the 80%, 85%, 90% speeds. The passage shock and the tip-leakage vortex also become more prominent with increased rotor speed. The passage shock can clearly be seen in all figures (Figures 46-48). Although the rotor had minimal tip-clearance, at near-stall a strong cross flow develops where the passage shock interacts with the blade boundary layer due to the tip-leakage vortex. This boundary layer experiences an accumulation of low momentum fluid after several passages creating a flow blockage near the pressure suction side of the leading edge preventing incoming flow into the passage [22] shown in Figures 46-48.

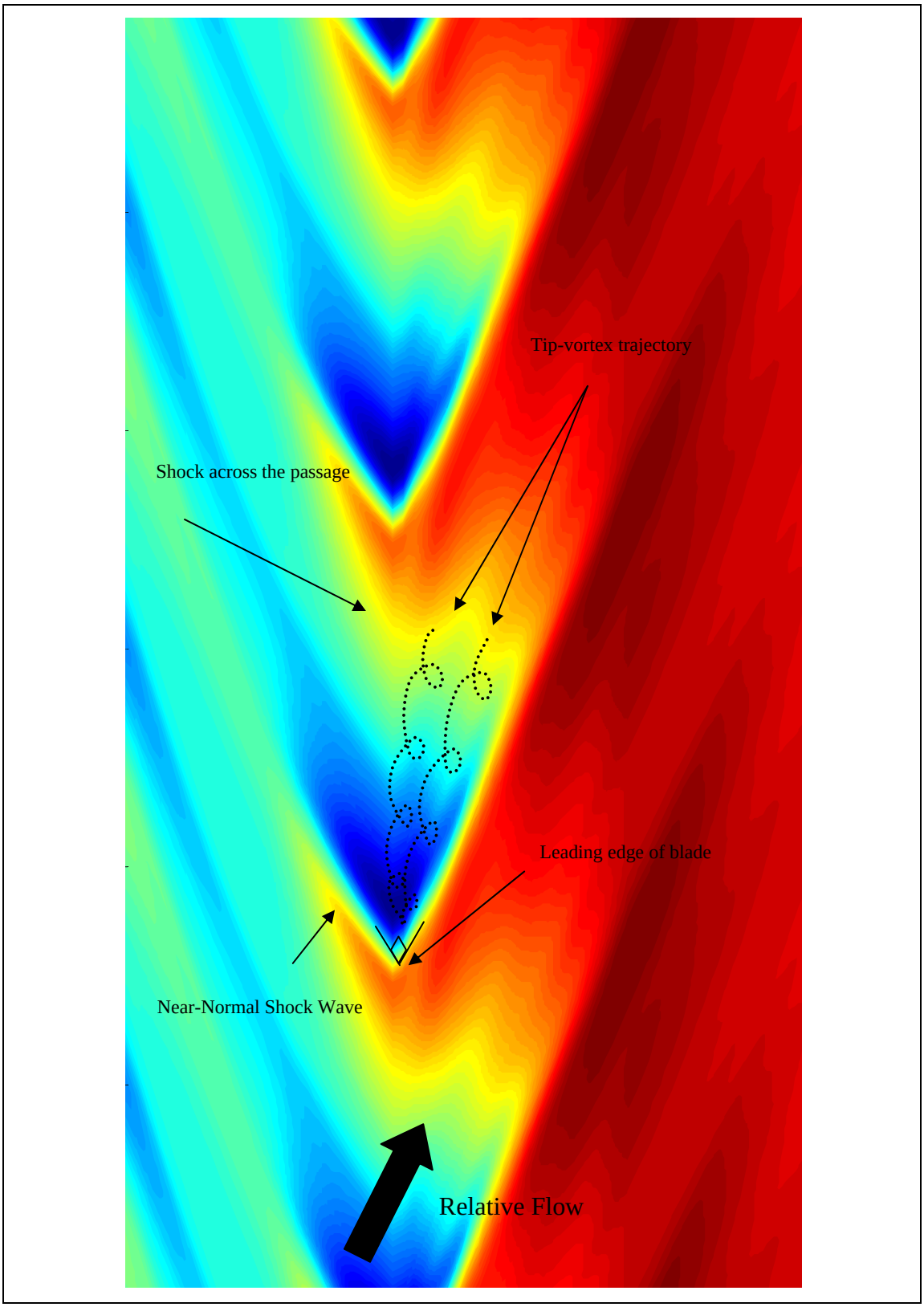


Figure 45. Near-Stall for 70% Speed

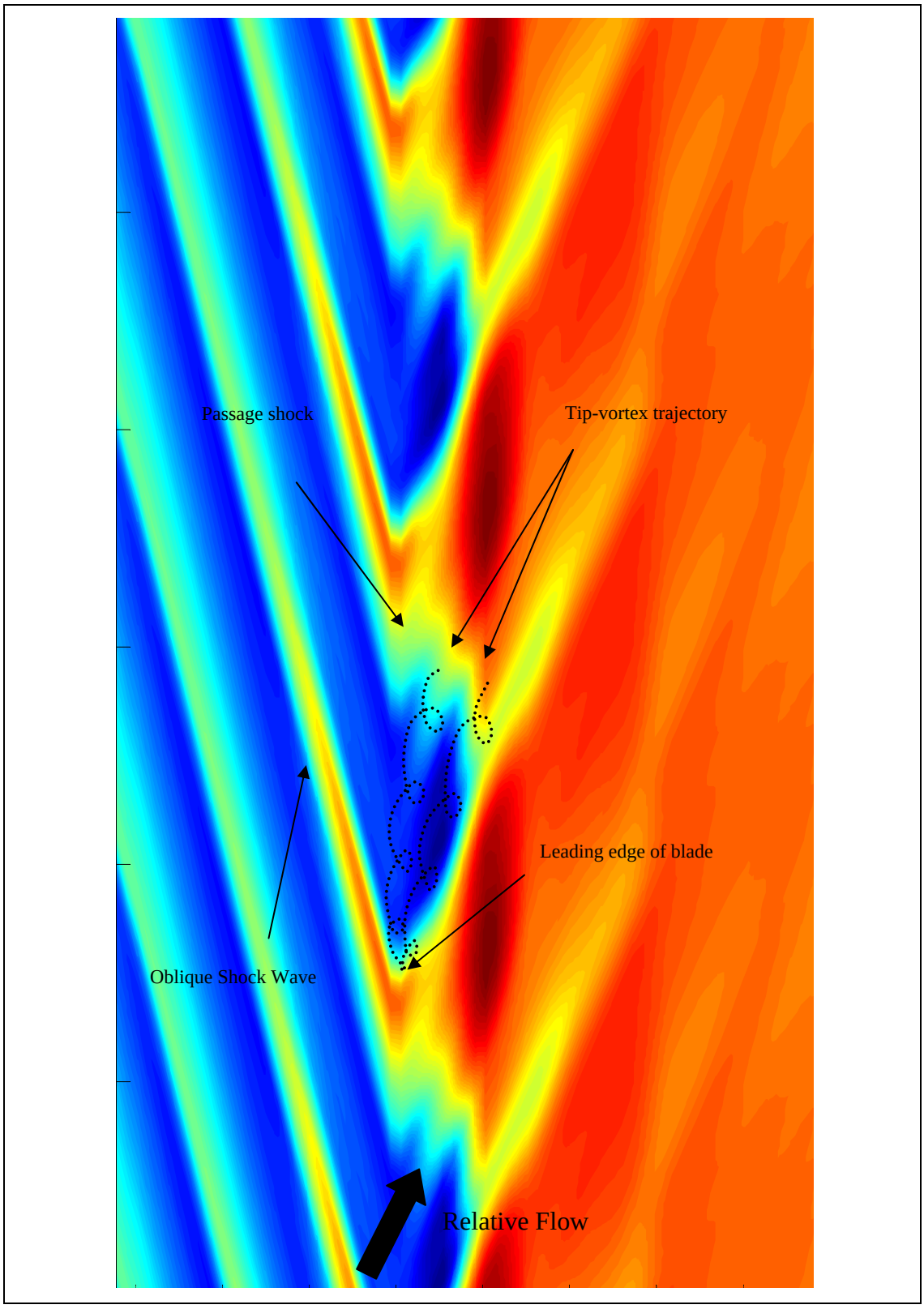


Figure 46. Near-Stall for 80% Speed

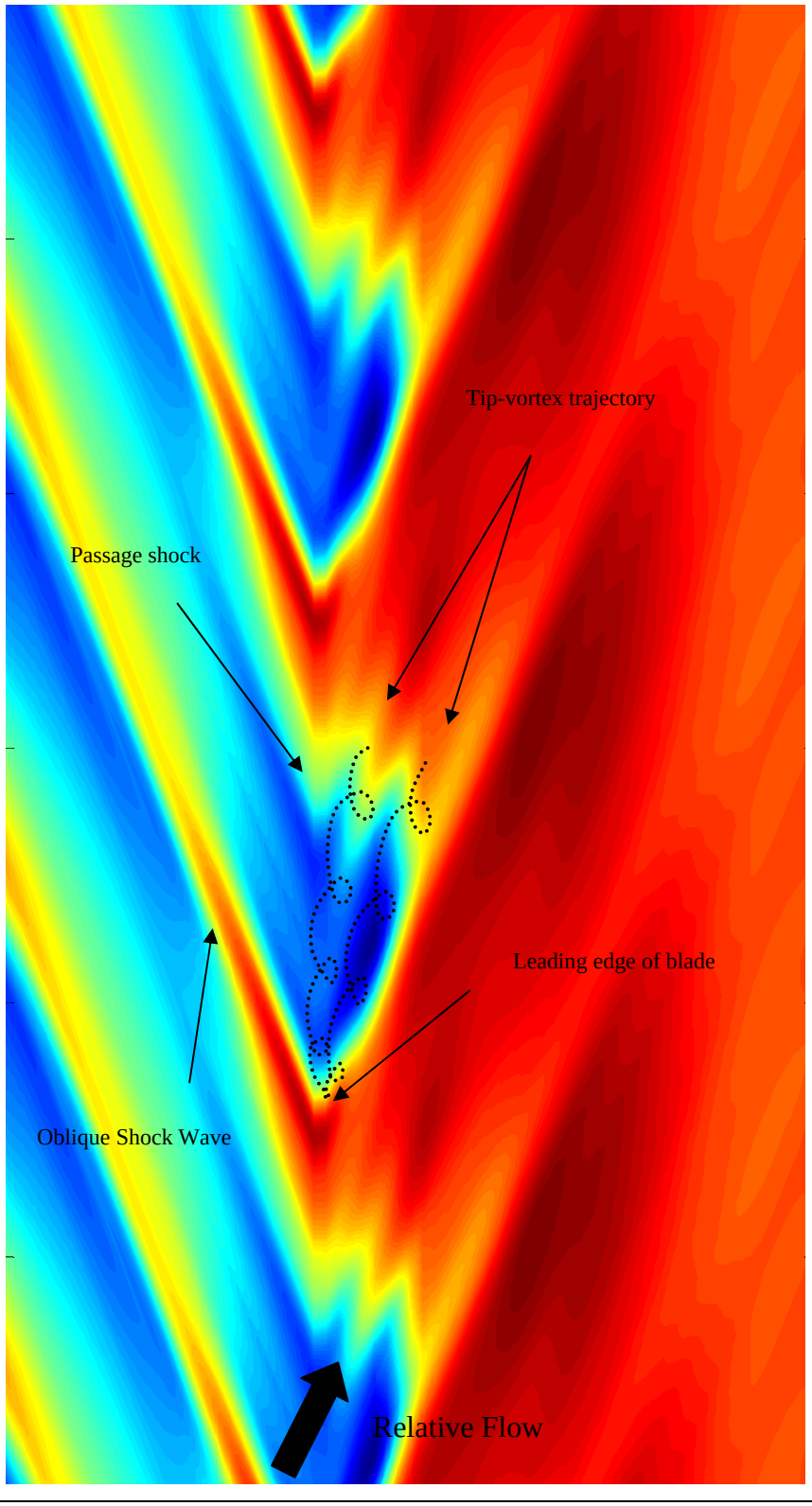


Figure 47. Near-Stall for 85% Speed

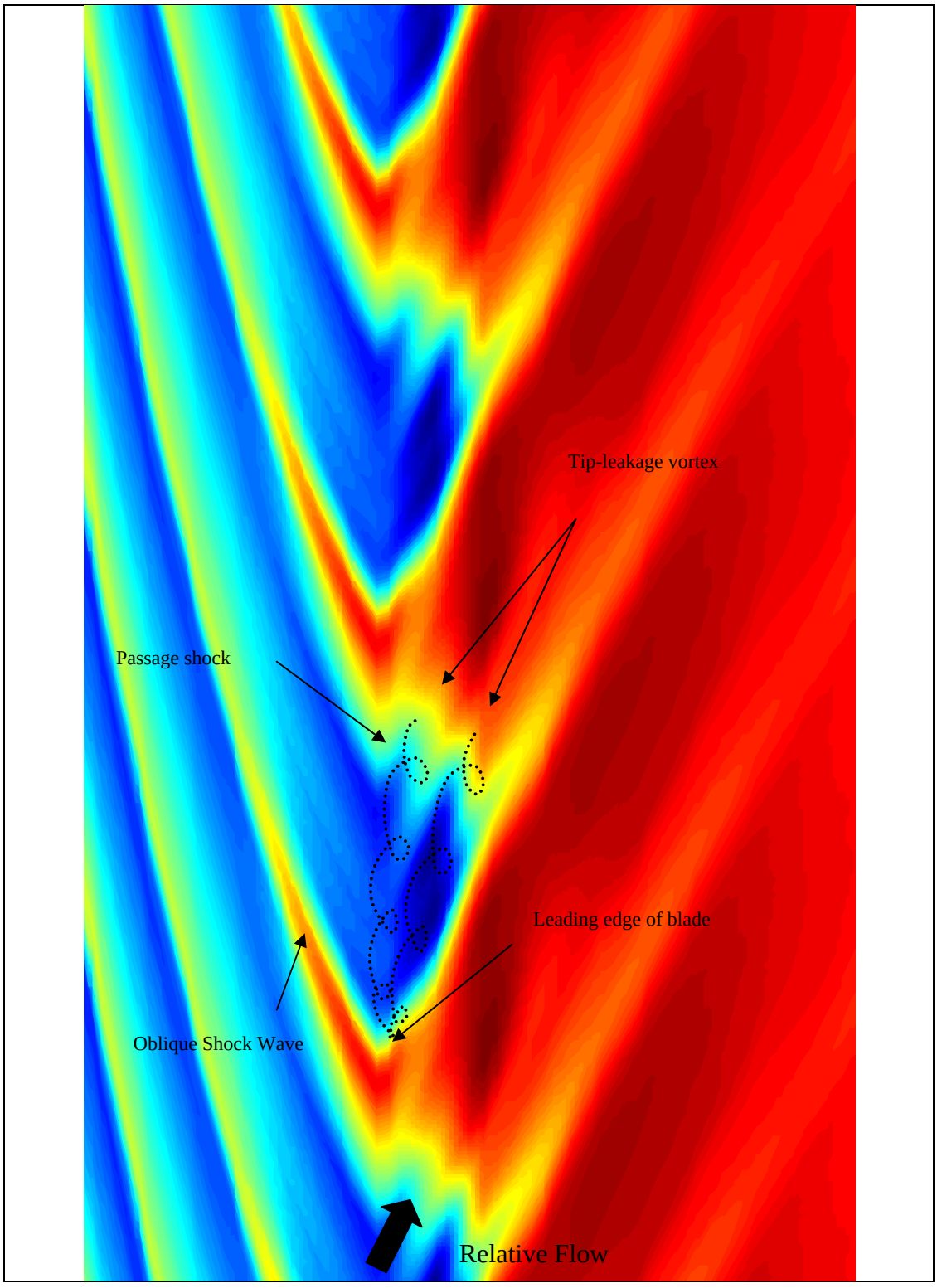


Figure 48. Near-Stall for 90% Speed

B. AMPLITUDE MAPS FOR FREQUENCIES OF INTEREST

Amplitude distribution maps for non-periodic and periodic disturbances are analyzed in Figures 49 and 50 using a logarithmic scale. Figures 43 and 44 showed the data using a linear scale. Figure 49 illustrates the non-periodic disturbance, and shows that the 56.5% BPF prevails in the area where the shock wave and the passage shock interact. The pockets of low amplitude correspond with the tip-vortex trajectory, but further interaction was not investigated.

Figure 50 illustrates the distribution of the periodic disturbance, which is prevalent in the area around the blade. The leading edge of the blade is most affected, as well as the beginning of the development of the shock wave. There is also a small area of high amplitude on the low-pressure side of the blade near the beginning of the tip-vortex trajectory.

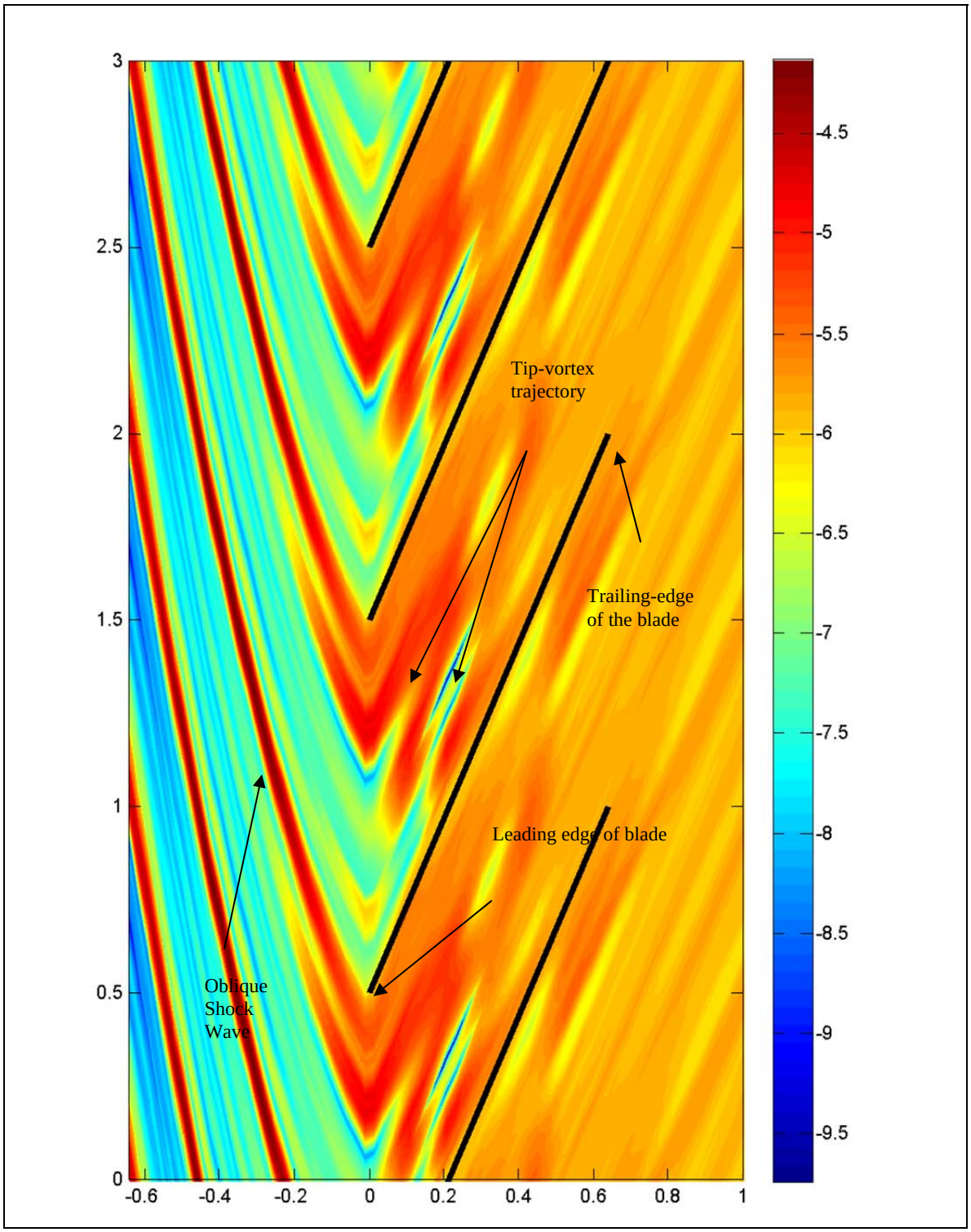


Figure 49. Log Scale of 56.5% OPR

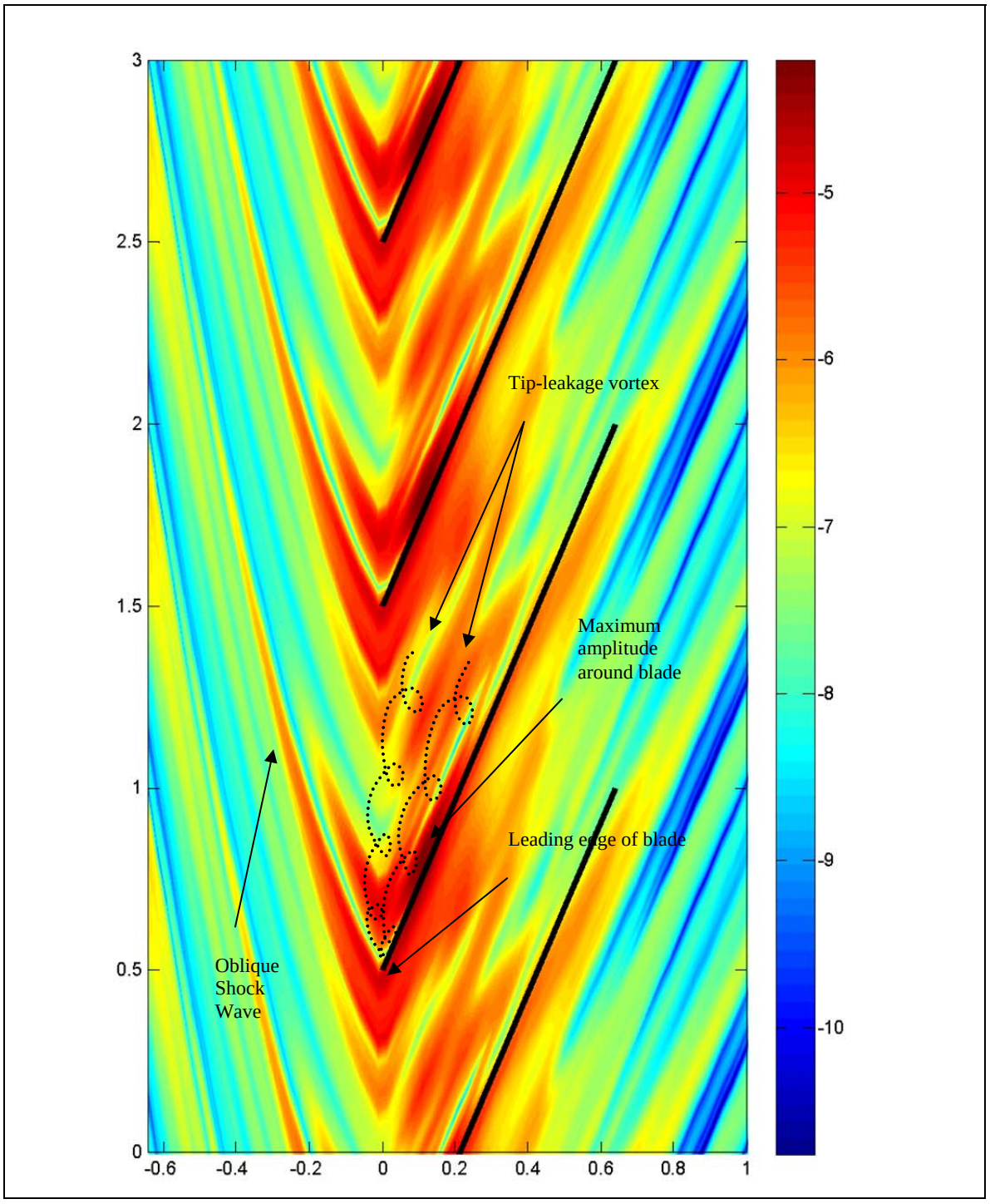


Figure 50. Log Scale of OPR

V. CONCLUSION

In conclusion a method adaptable to other compressors has been developed to identify periodic and non-periodic disturbances within the tip region of a transonic compressor. This research used effective yet discrete instrumentation in hopes that this data acquisition method could be easily transferred and implemented into an everyday operated military gas turbine engine.

Findings in the thesis include:

- A correction method for speed variation errors found within the TCR mechanisms.
- Irregular flow within a transonic compressor, as evidenced by differences in individual rotor blade passages from their neighbors and from themselves in previous revolutions.
- Periodic and non-periodic disturbances observed at the 90% rotor speed.
- A periodic (or once-per-revolution) disturbance was identified and prevalent over the blades.
- A non-periodic disturbance (identified at 56.5% of the rotor speed) was prevalent along the oblique shock wave and passage shock region.
- Another disturbance was identified after the 7^{th} OPR between 40-60%.

Plotting the phase for each frequency of interest can discover the location of the low dominant frequency and its interaction within the passage, which was partially done but in this thesis. This study uses effective sampling techniques, instrumentation, and appropriate numerical methods to isolate and filter low frequency instabilities using FFTs, as well as presented passage-to-passage distributions within the rotor blade passage prior to stall.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] A.J. Gannon and G.V. Hobson, “Pre-Stall Modal Instabilities in a Transonic Compressor Rotor,” International Symposium of Air Breathing Engines. Beijing, China, September 2007.

[2] N.L. Sanger, 1996, "Design of a low aspect ratio transonic compressor stage using CFD techniques," *ASME Journal of Turbomachinery*, 118, no.3, pp 479-491.

[3] A. Young and I. Day, “Stall warning by blade pressure signature analysis,” presented at the ASME Turbo Expo 2011, Vancouver, Canada, 2011.

[4] C. Hah, J. Bergner, and H. Schiffer, “Short length-scale rotating stall inception in a transonic axial compressor – criteria and mechanisms,” in *Proceedings of ASME Turbo Expo: Power for Land, Sea, and Air,* 2006, pp. 1–10.

[5] B. Hoss and L. Fottner, “Experimental setup, measurement and analysis of the onset of compressor flow instabilities in an aeroengine,” Insitut fur Strahlantriebe, Universitat der Bundeswehr, Munchen, Germany, Technical Report, pp. 117–129.

[6] C.S. Tan, I. Day, S. Morris, and A. Wadia, “Spike-Type Compressor Stall Inception, Detection, and Control,” in *Annual Review of Fluid Mechanics,* vol. 42, pp. 275–300, November 2010.

[7] M. Zaki, L.N. Sankar, and S. Menon, “Hybrid Reynolds-averaged Navier-Stokes/Kinetic-Eddy Simulation of stall inception in axial compressors,” in *Journal of Propulsion and Power*, vol. 26, no. 6, pp. 1276–1282, November-December 2010.

[8] A.J. Gannon, and G.V. Hobson, “Pre-stall instability distribution over a transonic compressor rotor,” in *ASME Journal of Fluid Engineering,* vol. 131, pp. 1–11, May 2009.

[9] C. Hah, M. Voges, M. Mueller, and H.P. Schiffer, “Characteristics of tip clearance flow instability in a transonic compressor,” in *Proceeding of ASME Turbo Expo 2010: Power for Land, Sea, and Air*, Glasgow, UK, 2010.

[10] H.J. Weigl, J.D. Paduano, L.G. Frechette, A.H. Epstein, E.M. Greitzer, “Active stabilization of rotating stall and surge in a transonic single stage axial compressor,” *ASME Journal of Turbomachinery*, 120, no 4, pp. 625-637.

[11] S.T. Bailie, W.F. Ng, and W.W. Copenhaver, “Experimental reduction of transonic fan forced response by inlet guide vane flow control,” *ASME Journal of Turbomachinery,* vol. 132, pp. 1–8, April 2010.

[12] J.N. Chi, “Modeling for the development and evaluation of strategies for controlling flow instabilities in transonic compressors,” *in Proceedings of IMECE2007, 2007 ASME International Mechanical Engineering Congress and Exposition*, Seattle, Washington, 2007.

[13] N. Gourdain and F. Leboeuf, “Unsteady simulation of an axial compressor stage with casing and blade passive treatments,” *ASME Journal of Turbomachinery*, vol. 131, pp. 1–12, April, 2009.

[14] I. Wilke and H.P. Kau, “A numerical investigation of the flow mechanism in a high pressure compressor front stage with axial slots,” *ASME Journal of Turbomachinery*, vol. 126, pp. 339–34, July 2004.

[15] B.K. Jha. and A.N.V. Rao, “Development of instrumentation to capture unsteady and flutter phenomena in the fan rotors of gas turbine engines,” *in Instrumentation and Measurement Technology Conference, 2011IEEE,* pp. 1–8, May 2011.

[16] M. Voges, R. Schnell, C. Willert, R. Monig, “Investigation of blade tip interaction with casing treatment in a transonic compressor – Part 1: Particle Image Velocimetry,” *ASME Journal of Turbomachinery*, vol. 133, pp. 1-13, January, 2011.

[17] M.P. Wernet, MM. Bright, and G.J. Skoch, “An investigation of surge in a high-speed compressor using digital PIV,” NASA/TM, December, 2002.

[18] Kulite, “Infinite Tube Sensor” U.S. Patent 7,975,552, July, 12, 2011.

[19] C Teolis, D Gent, and C. Kim, “Eddy Current Sensor Processing for Stall Detection,” in *IEEEAC paper # 1255*, Version 3, pp. 1–17, December 2004.

[20] M.W. Rodgers, “Unsteady pressure measurements on the case wall of a transonic compressor,” M.S. Thesis, Naval Postgraduate School, Monterey, CA, 2003.

[21] J.J. Koessler, “Experimental investigation of high-pressure steam induced stall of a transonic compressor,” M.S. Thesis, Naval Postgraduate School, Monterey, CA, 2007.

[22] W.L. Davis, “Stall analysis in a transonic compressor stage and rotor,” M.S. Thesis, Naval Postgraduate School, Monterey, CA, 2003.

[23] K. Yamada, K. Funazaki, and M. Furukawa, “The behavior of tip clearance flow at near-stall condition in a transonic axial compressor rotor,” in *Proceedings of ASME Turbo Expo 2007: Power for Land, Sea, and Air, Montreal, Canada, 2007.*

[24] C. Hah and D.C. Rabe, “Role of tip-leakage vortices and passage shock in stall inception in a swept transonic compressor rotor,” *in Proceedings of ASME Turbo Exp: Power for Land, Sea, and Air*, Vienna, Austria, June 2004.

[25] G.L. Descovich, "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.

[26] Kulite, “Miniature IS Pressure Transducer” U.S. Patent 7,484,418 B2, February 3, 2009.

[27] F.M. White, *Fluid Mechanics 4th Edition*. New York, NY: McGraw Hill International, 1998.

[28] A.J. Gannon, G.V. Hobson, and W.L. Davis, “Axial transonic rotor and stage behavior near the stability limit,” *ASME Journal of Turbomachinery,* vol 134, pp. 1–8, January 2012.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MATLAB CODE

The code was written in several parts. The main files and their function codes are summarized below. *If any or all of this code is used, this Thesis must be referenced.*

- Main Calibrate Load Calibration
- Main Analyze (Parts 1, 2, and FFT): Kulite Calculations, Voltage Calibration, Pressure Read, Process Kulite Data
- Contour Plots
- FFT Test Case: FFT Calc
- FFT Analyze
- Contour Plots FFT

Main Calibrate is used first. This code calibrates the probe pressures and ensures that all future calculations will be correct. *Main Analyze* has three parts (*1, 2, and FFT*). The first two files are used one right after the other. *Main Analyze 1* is used to convert from the time to the space domain. *Main Analyze 2* is used for generating the superimposed pressure plots for each Kulite probe. After the mean of the superimposed pressure signals is calculated, *Contour Plots* is run to create the contour plots. This code shifts the data to match the probe casing position, and interpolates along the blade position and the shock angle.

Main Analyze FFT is used to generate the same information at *Main Analyze 2* but the outputs are stored in a way to ease file handling. *FFT Test* case reads *Main Analyse*s' files and feeds each file through an FFT leaving a total of a 1,000 files to be analyzed. *FFT Analyze* interrogates each signal to find the maximum amplitude and its corresponding frequency throughout the signal. The output of *FFT Analyze* is the frequencies of interest, which are then fed into *Contour Plots FFT*, which generates a contour plot for the amplitude distribution for each frequency.

Main Calibrate

```
kulite.numbering =  [-1  -2   0   1   2   3   4   5   7   8];% go with the
flow, upstream to downstream

calibrate.filenames = cellstr(strvcat('Dx2011_1004_cal_0.csv',...
                            'Dx2011_1004_cal_5.csv',...
                            'Dx2011_1004_cal_10.csv',...
                            'Dx2011_1004_cal_15.csv',...
                            'Dx2011_1005_cal_0.csv',...
                            'Dx2011_1005_cal_5.csv',...
                            'Dx2011_1005_cal_10.csv',...
                            'Dx2011_1005_cal_15.csv'));

% Kulite calibration levels in inches of Mercury for each file
calibrate.pressure = [0 5 10 15 0 5 10 15]';

% Kulite calibration levels in Pascals ...
% (Negative due to being at rear of probe), converting from Hg mm to Pa.
calibrate.pressure = -101325*calibrate.pressure/29.921261;

% Reads the files in based on the names in filenames
for ii = 1:size(calibrate.filenames,1)
    disp(['Reading file ' char(calibrate.filenames(ii,:)) '...']);  % Displays
which file is being read
    [ NUMERIC_RAW, TXT(:,:,ii), ~ ] = xlsread(char(calibrate.filenames(ii,:)));
% reads the file in
    NUMERIC_MEAN(ii,:) = mean(NUMERIC_RAW,1); % Mean of each column is stored
as this is all that is needed for the calibration case.
end

%TXT = r x s cell string array containing text cells in RANGE.
% This assigns the voltages

for ii = 1:size(calibrate.filenames,1)        % The number of input files
    kk = 1;                           % Keeps a tally of the number of probes
    for jj = 1:size(TXT,2)            % The number of columns

calibrate.kulite = char(TXT(3,jj,ii)) == 'K';
calibrate.voltage = char(TXT(5,jj,ii)) == 'V';
       if ~isempty(calibrate.kulite) && calibrate.kulite(1) &&
calibrate.voltage % program still runs
            calibrate.volts(:,kk) = NUMERIC_MEAN(:,jj);% Only takes data if the
header is 'V' for voltage
            kk = kk+1;
         end
    end % for jj
end % for ii

% This calculates the constants required for calibration
figure(1); close; figure(1);
for ii = 1: size(calibrate.volts,2)
    [P(ii,:),S(ii),MU(:,ii)] =
polyfit(calibrate.volts(:,ii),calibrate.pressure,1);

    % figures are plotted
    plot(calibrate.volts(:,ii),calibrate.pressure,'-or'); hold on; grid on;
end % for ii

xlabel('Voltage [V]'), ylabel('Pressure [Pa]'), title('Calibration Curve'),
```

```
save kulite_calibrate P S MU
--------------------------------------------------------------------------------
function volts = load_calibration(kulite)

[NUMERIC,TXT,~] = xlsread(char(kulite.filenames(1,:)));
probes = 1;                        % Keeps a tally of the number of probes


for columns = 1:size(TXT,2)        % The number of columns
    volts.kulite = char(TXT(3,columns)) == 'K';
    volts.voltage = char(TXT(5,columns)) == 'V';
    volts.onceperrev= char(TXT(3,columns)) == 'O';

    if ~isempty(volts.kulite) && volts.kulite(1) && volts.voltage %
           volts.volts(:,probes) = NUMERIC(:,columns);
           probes = probes+1;

    elseif char(TXT(5,columns)) == 's'
           volts.time(:,1) = NUMERIC(:,columns);

    elseif ~isempty(volts.onceperrev) && volts.onceperrev(1) ...
               && volts.voltage % program still runs
           volts.trigger = NUMERIC(:,columns);

    end % if loop

    if kulite.status~= 1
        volts.volts(:,probes) = mean([ NUMERIC(:,columns-1)
NUMERIC(:,columns+1) ,2]);
    end
end % for jj
    % RPM is found from last column of data
volts.rpm = mean(NUMERIC(find(~isnan(NUMERIC(:,end))),end));
```

Main Analyze

```
% m file to load data from a Kulite use the previous calibration and output the
pressure levels from the Kulites
close all
clear all
% Load calibration data
load kulite_calibrate P MU S
% Filename to be analyzed
kulite.filenames = cellstr(strvcat('Dx2011_1004_90_13.csv'))
% File is read in
disp(['Reading file ' char(kulite.filenames(1,:)) '...']);
%Identify kulites
kulite.numbering =  [-1  -2   0   1   2   3   4   5   7   8];% go with the
flow, upstream to downstream
kulite.length = length(kulite.numbering);
disp('Number of Exisiting Kulites')
disp(kulite.length)
kulite.status = [1 1 1 1 1 1 1 1 1 1];
disp('Loading Volts, Time, Trigger, and RPM...')
volts = load_calibration(kulite);

%Manually Input Kulite Status

% Kulite Probe Measurements and Calculations
%Aug 10, 2011 Error found in kulite probe numbering of '-2' and '-1'
%dont know how long they have been mixed up and which data is affected. And
%how it will affect calibration, etc.

% Kulite 0 is the leading edge of the blade
% Kulite 6 is the trailing edge, unfortunately this kulite has been dead
% for all data gathering.
% Kulites '9' and '-3' will not be used, empty probes.

%Input total number of blades:
blade.number = 20;
blade.pitch = 27;%deg
%Input kulite angle and axial position along the casing:
kulite.data.theta.posit.raw = [ -45  -36  -27  -18  -9   0   9  18  36  45];
%degrees
kulite.data.axial.posit.raw = [ 8.5298    7.9611    7.3923    ...
    7.2027    7.0129    6.8235    6.6337    6.4443    5.7997  5.3447];
kulite.data.axial.posit.fix = -(kulite.data.axial.posit.raw -
(ones(1,kulite.length)*kulite.data.axial.posit.raw(3)));

%finds the leading edge kulite and sets the leading edge to zero
%upstream and then downstream!

%inches, kulite__axial_posit(1) is kulite '8' and kulite_axial_posit(11) is
%kulite '2' taking a look at the kulites placed on the casing.

%figure(31), plot(kulite_angle, fliplr(kulite_axial_posit))%shows you the top
view of reading the kulites, however, this is not
% the direction we want for our contour plot so we will leave it alone.

[kulite, contourdata, blade] = Kulite_Calculations(blade, kulite)

figure(1),
plot(kulite.data.axial.posit.corrected, kulite.data.theta.posit.corrected),
title('Instrument Position on Casing'),
text(kulite.data.axial.posit.corrected', ...
    kulite.data.theta.posit.corrected',num2str(kulite.numbering'))
```

```
axis equal,
grid on

figure(2),
plot(kulite.data.axial.posit.corrected, kulite.data.theta.posit.corrected),
hold on
plot(blade.axes.x,blade.axes.y)
title('Shifting of Instrumentation onto Blade'),
text(kulite.data.axial.posit.corrected',kulite.data.theta.posit.corrected',num2
str(kulite.numbering')),
text(blade.axes.x',blade.axes.y',num2str(kulite.numbering')),
axis equal,
grid on

figure(3),
plot(contourdata.blade.kulite.axial, contourdata.blade.kulite.pitch),
title('Instrumentation Positioned on Blade')
text(contourdata.blade.kulite.axial',contourdata.blade.kulite.pitch',num2str(ku
lite.numbering')),
axis equal,
grid on

figure(4)
plot(volts.time, volts.volts),
xlabel('Time (Sec)'),
ylabel('Volts (Volts)')
grid on,
title('Kulite voltage through time')

% Apply calibration to measured Voltages
pressure = Voltage_Calibration (volts, P, S, MU, kulite);

%add atm pressure from low speed data
% Saved txt file into xls so that its easier for MATLAB to read.
%[NUMERIC, TXT,
RAW]=xlsread('C:\Londono\90%NPSMF\90%_2011_02_18\Air_90_pc_Spreadsheet.xls')

% I did this to make sure that the program was correctly reading in the
% spreadsheet. My issue was that it loaded properly but the values all
% appeared as zeros. This is due, now I come to find after spending several
% hours of trying different methods of reading in 'txt' and 'xls' files
% that since 'Tinf6' column S is '9.91e37' it makes the rest of the values
% in the spreadsheet seem like zeros but once you call the individual row,
% column or cell out the correct value appears.
% Incorporate Low Speed Data, Pressures
pressure.lowspeed =
dlmread('C:\Londono\Transonic\TCR_Data_Files\70%NPSMF\70%2011_10_04\Air_70_pc_S
preadsheet.txt', '\t', 1,0);
%im having trouble running dlmread now.. not sure why the error implies that
the file is not in the directory....
pressure = Pressure_read(pressure)
%Loc is the 1st blade passage of each revolution
%trigger is the
loc = Process_Kulite_Data(volts)

loc.hertz = 1./diff(volts.time(loc.loc)); %units-hertz
loc.hertz.mean = mean(loc.hertz); %unit hertz
% Non-dimensionalize and plot Blade Passages vs Pressure, working in the
%position domian
% I want to make the variable 'blade_passage' a non-dimensional number
% begining with 0. So I take
loc.rev = volts.time(loc.loc); % time in seconds of each revolution pass
```

```
loc_rev = loc.rev;
loc.rev.difference = diff(loc_rev); %time difference between each revolution
loc_rev_diff = loc.rev.difference;
loc.rev.avg = mean(loc_rev_diff); %takes the average of the difference between
each revolution

%Once per rev plot..this is a TTL circuit plot... it is evident that the
%trigger is changing slightly... lets see after the non-dimentionization

% NON-DIMESIONALIZATION: Time to space domain

loc.offset = 2;
loc.posit.shift= volts.time- volts.time(loc.loc(loc.offset));%time shift to the
left
loc.posit.nondim = loc.posit.shift/loc.posit.shift(loc.loc(end+(loc.offset -
3)));
%non dimensionalized with the last trigger we wanted
loc.posit.fix= loc.posit.nondim*(length(loc.loc)+(loc.offset - 5));
%multiplied by the number of blade passages or revolutions, ...
%check the once per rev: it is in fact moving to the right, possibly due to
%acceleration.. miniscule position move due to the 2 second sample... you
%have a drive and an aerodynamic load so the acceleration should be constant
%and smooth throughout the course of the sample.

% the trigger is still moving, but not as bad as it was before. we need to
% fit a polynomial  to the actual rpm... if the line falls before the
% trigger its indicating high speed. if its right after the trigger it
% indicates low speed.
%diff(Time(Loc));%will yield the same time between the time of the trigger pass
%diff(Posit(Loc));% will not show the same position between trigger pass.. the
numbers range from .9965 to 1.0033
%clearly a change in position.
% Trigger Plot (without correction)
%
figure(4)
plot(loc.posit.fix, volts.trigger)
hold on
plot(loc.posit.fix(loc.loc), volts.trigger(loc.loc), '*r'), title('Trigger
Locations'),
xlabel('Position (Revolution)'), ylabel('Voltage (Volts)'),
xlim([-5 910]), grid on
%
% acceleration error calculated
loc_posit_fix = loc.posit.fix;
loc_loc = loc.loc;
loc.error = mean(diff(loc_posit_fix(loc_loc)))-1; % error in trigger
position... to the e-6... extrmly small

% Trigger Acceleration- PROBLEM!
%to fix need to polyfit and then polyval:

loc.p = polyfit((1:length(loc.posit.fix(loc.loc))-
1)',diff(loc_posit_fix(loc_loc))-1, 2) ;
loc.val = polyval(loc.p, 1:length(diff(loc_posit_fix(loc_loc)))-1);

%Method 1:
% figure(2)
% plot(diff(Posit(Loc))-1)
% hold on
% plot(val, 'r'), title('Change in RPM')

%Method 2: PROVED TO BE BETTER METHOD!
```

```
loc.error= ( loc_posit_fix(loc_loc)-
(round(loc_posit_fix(loc_loc(1))):round(loc_posit_fix(loc_loc(end))))' );
%plot(Loc_error), title('Trigger Position Error') % can see 0.1 error in passes
%the error is due to mechanisms.. there are a lot of inconsistencies in
%the electric motor, fluid coupling, 12 stage axial compressor, and
%compressor itself.
% add the error to the Locs ...

% Correcting the Error:
% Process:
% 1) Plot the error vs the Posit(Loc) and make sure the lengths are the same.
% 2) smooth the error curve (could be left out for now)
% 3) use interp make a vector the same length as posit from the much shorter
error vector.
% 4) Correct the entire posit vector
% 5) check that all the triggers now lie at integer values.
% Step 1: Plot the error vs the Posit(Loc)
%and make sure the lengths are the same. error (897X1)

loc.posit.final = loc_posit_fix(loc_loc); %(897x1)
loc_posit_final = loc.posit.final;
%plot(Posit_loc, Loc_error), title('Trigger Error at each Loc')
% Step 2: Plot Smoothed vs Error
loc.error.smooth.position = smooth(loc_posit_fix(loc_loc)-
(round(loc_posit_fix(loc_loc(1))):round(loc_posit_fix(loc_loc(end))))',0.05,
'rloess');
loc.error.smooth.location = ( (loc.offset-3): -
(loc.offset)+length(loc_posit_fix(loc_loc)-
(round(loc_posit_fix(loc_loc(1))):round(loc_posit_fix(loc_loc(end))))'));
loc.error.corrected = loc_posit_fix(loc_loc)-
(round(loc_posit_fix(loc_loc(1))):round(loc_posit_fix(loc_loc(end))))';
%check dimensions!
figure(5)
plot(loc.error.smooth.location, loc.error.corrected), title('Trigger Position
Error (Smoothed)') % can see 0.1 error in passes
hold on
plot(loc.error.smooth.location, loc.error.smooth.position, 'r'),
title('RPM Error Smoothed '), ...
%     %plot smoothed data on top of original data
legend('Original Data', 'Smoothed Data using "rloess"'),xlim([-5 910]),
grid on, xlabel('Position'), ylabel('Error')
%Step 3: Interpolate
%3) use interp make a vector the same length as posit from the much shorter
error vector.

lesl = loc.error.smooth.location;
lesp = loc.error.smooth.position;
lec = loc.error.corrected;
kulite.position.interp = interp1( lesl, (lesl-lesp'),loc_posit_fix, 'spline');
 %while running 90pc runs 1 and 5, MATLAB would not run Main_Analyze_2. The
 %reason is because there were multiple passages for the xdata. tracing
 %this down from kulite.position.correct to kpi to kulite.position.interp
 %to loc creation.... what is interesting is why the other runs for 70,80,
 %85 pc were not affected?
kpi = kulite.position.interp;
kulite.position.smooth = smooth((kpi(loc_loc)...
    -(round(kpi(loc_loc(1))))): ...
    round(kpi(loc_loc(end))))',0.05, 'rloess');

% Step 4: Correct Position Vector
kulite.position.correct =  kpi*blade.number; % total number of blade passings
```

```
% Trigger Plot with Error Correction
figure(6)
plot(loc.posit.fix, volts.trigger)
hold on
plot(kulite.position.interp, volts.trigger, 'g')
hold on
plot(loc.posit.fix(loc.loc), volts.trigger(loc.loc), '*r'), title('Trigger
Position with/out Error'),
legend('Original Trigger', 'Corrected Trigger'),
xlim([-5 910]), grid on, xlabel('Position'), ylabel('Voltage')

figure(7)
plot(kulite.position.correct, pressure.final),
title('Final Kulite Pressures After Non-dimensionalization'),
xlabel('Pressure (Pa)'),
ylabel('Position'), grid on, xlim([-10 915]), ylim([0.4 1.8])
%When it plots, DELETE the green that's a bad kulite...develop some sort of
buffer
%try maybe subplots, its starting with the first Kulite and then drawing over.
%verification put a line at each integer, to ensure that we did it right.

%plot(Posit(1:27), Press_F(1:27,:))plot of all the Kulites

save('kulite.mat', 'kulite')
save('pressure.mat', 'pressure')
save('blade.mat', 'blade')
save('contourdata.mat', 'contourdata')
save('loc.mat', 'loc')
save('volts.mat', 'volts')
--------------------------------------------------------------------------
function [kulite, contourdata, blade] = Kulite_Calculations(blade, kulite)

kulite.index.le = kulite.numbering(3); %leading edge of blade
%kulite.index.te = kulite.numbering(9); %trailing edge of blade, doesnt
%exist anymore because KUlite 6 is taken out of data.

%Rotor dimensions
rotor.diameter = 11.3; %inches
rotor.circumference = pi()*rotor.diameter; %inches
rotor.pitch.length = rotor.circumference/blade.number;%inches

%Kulite Positioning Theta Direction
kulite.data.theta.posit.nondim =
kulite.data.theta.posit.raw/(360/blade.number);%Kulite position
%along the theta direction, non-dimensionalizes the pitch axis
kulite.data.theta.posit.corrected = (kulite.data.theta.posit.nondim - ...
    kulite.data.theta.posit.nondim(find( kulite.numbering ==
kulite.index.le)));
%kulitge non dim finds the leading edge kulite and sets the leading edge to
zero.
%kulites are separated half a pitch apart- this is in the theta direction

%Kulite Positioning Axial Direction
kulite.data.axial.posit.corrected = kulite.data.axial.posit.fix/
rotor.pitch.length;
%kulite position along the z axis

%Blade Positioning
blade.pitch.length = blade.pitch/(360/blade.number);
%physical distance is 27 deg, now it is non dimensionalized.

% blade.axial.length = kulite.data.axial.posit.corrected...
```

```
%     (find( kulite.numbering == kulite.index.te) ) - ...
%      kulite.data.axial.posit.corrected(find( kulite.numbering == ...
%      kulite.index.le));  % 0.6409
blade.axial.length = 0.6409 - ...
     kulite.data.axial.posit.corrected(find( kulite.numbering == ...
     kulite.index.le));
blade.chord.length = sqrt(blade.axial.length^2 + blade.pitch.length^2);

blade.slope = (blade.pitch.length/blade.axial.length);
blade.angle.betaone = atan(blade.slope)*180/pi()

blade.axes.x = kulite.data.axial.posit.corrected;
blade.axes.y =
kulite.data.axial.posit.corrected*tan(blade.angle.betaone*pi()/180)

% figure(32),
% my_blade = line(x_blade, y_blade)
% hold on
% plot(kulite_axial_corrected_posit, kulite_corrected_theta)

%add kulite position to blade position for Contour Plots
contourdata.blade.kulite.pitch = -((kulite.data.theta.posit.corrected -
blade.axes.y) - ...
    kulite.data.theta.posit.corrected);
contourdata.blade.kulite.axial = blade.axes.x;
--------------------------------------------------------------------------
function pressure = Pressure_read(pressure)

% Saved txt file into xls so that its easier for MATLAB to read.
%[NUMERIC, TXT,
RAW]=xlsread('C:\Londono\90%NPSMF\90%_2011_02_18\Air_90_pc_Spreadsheet.xls')

pressure.atmosphere.one = mean(pressure.lowspeed(:,29));
pressure.atmosphere.two = mean(pressure.lowspeed(:,30));

%divide by stagnation press
pressure.atmosphere.avg = (pressure.atmosphere.one + ...
    pressure.atmosphere.two)/2;
pressure.stagnation = mean(pressure.lowspeed(:,26));

pressure.final = (pressure.calibrate + pressure.atmosphere.avg)/...
    pressure.stagnation;
--------------------------------------------------------------------------
function [pressure] = Voltage_Calibration (volts, P, S, MU, kulite)

pressure.calibrate = zeros(size(volts.volts));
for ii = 1: size(volts.volts,2)

    pressure.calibrate(:,ii) =
polyval(P(ii,:),volts.volts(:,ii),S(ii),MU(:,ii)); % Calibration is applied,
outputs pascals
end
for jj = 1:size(pressure.calibrate,2)        % The number of columns/kulites
     if kulite.status(jj) == 0
         pressure.calibrate(:,jj)= mean([pressure.calibrate(:,jj+1)
pressure.calibrate(:,jj-1)],2);
     end
end

 [B,IX] = sort(kulite.numbering);

pressure.calibrate = pressure.calibrate(:,IX);
```

```
--------------------------------------------------------------------------------
function loc = Process_Kulite_Data( volts )

volts_time = volts.time;
loc.samples = length(volts_time);

loc.trig = mean([min(volts.trigger) max(volts.trigger)]);

loc.loc = find( volts.trigger(2:loc.samples)<loc.trig & ...
    volts.trigger(1:loc.samples-1)>loc.trig );

if length(loc.loc) > 2
    if loc.loc(1) < 3
        loc.loc = loc.loc(2:end);
    end
end

loc.loc = loc.loc(1:(end-1));

save loc
```

Main Analyze 2

```
close all
clear all
load('kulite.mat')
load('pressure.mat')
load('blade.mat')
load('contourdata.mat')
load('loc.mat')
load('volts.mat')

% Single Kulite Plots
%assign storage for each vector
count.lines= 50;%lines
count.dots=100;%dots

passage.position.raw = zeros(count.lines, count.dots+1);

for iteration = 1:kulite.length

figure(iteration+7);

for cycles=1:count.lines; %number of cycles
    range(cycles) = find(kulite.position.correct > (cycles - 0.25) ,1,'first');
    range(cycles+1) = find(kulite.position.correct < ((cycles+1)+
0.25),1,'last');

    passage.position.raw = linspace(0,1,count.dots +1);

    h(cycles) = plot(kulite.position.correct(range(cycles):range(cycles+1))-
cycles, ...
                pressure.final(range(cycles):range(cycles+1),iteration));


           for points = 1: count.dots + 1

           passage.position.data(cycles, points) = interp1(get(h(cycles),
'Xdata'),...
                get(h(cycles),'Ydata'), passage.position.raw(points));
%            title('Kulite Corrected Data')
hold on
set(h(cycles), 'ZData',(cycles*(ones(size(get(h(cycles), 'XData'))))));
 passage.position.mean(points) = mean(passage.position.data(cycles,:));
           end

     plot3(passage.position.raw, passage.position.data(cycles,:),
iteration*ones(size(passage.position.raw)), '*r');
%      hold on
%
plot3(passage.position.raw,passage.position.mean,iteration*ones(size(passage.po
sition.raw)),'-og') %plots the mean of all the cycles
%
     cycles
end
title('Kulite Data for Each Blade Pass'),xlabel('Passage
Position'),ylabel('Pressure (Pa)'),
grid on,

if passage.position.raw(iteration) ~= 1
     kdp(iteration,:) = mean(passage.position.data,1);
else passage.position.raw(iteration) == 1
     kdp(iteration,:)= mean([kdp(iteration-1,:) kdp(iteration+1,:),2]);
```

```
end
end

figure(18)
plot([passage.position.raw (passage.position.raw + 1.01)],[kdp kdp]),
grid on, title('Kulite Pressure Averages'), %this is a correct plot- do not
touch
xlabel('Passage Position'), ylabel('Pressure (Pa)')
plot.threed.posit.axial = kulite.data.axial.posit.corrected'*
ones(1,size(kdp,2));%this yields a 11x101 matrix
plot.threed.posit.pitch = kulite.data.theta.posit.corrected' *
ones(1,size(kdp,2));%this yields a 11x101 matrix

figure(19)
plot3(passage.position.raw,kdp, plot.threed.posit.pitch) %this is a correct
plot- do not touch
xlabel('Kulite Data Passage Position'), ylabel('Kulite Data Passage'),
zlabel('Kulite Corrected Theta (Pitch Direction)')
grid on


scale.passage.position = [(passage.position.raw-4) (passage.position.raw-3)
(passage.position.raw-2) (passage.position.raw-1) ...
    (passage.position.raw) (passage.position.raw+1) (passage.position.raw+2)
...
    (passage.position.raw+3) (passage.position.raw+4) (passage.position.raw+5)
(passage.position.raw+6) (passage.position.raw+7)];
scale.theta.contour = [(ones(1, size(kdp,2))) (ones(1, size(kdp,2))) ...
    (ones(1, size(kdp,2))) (ones(1, size(kdp,2))) (ones(1, size(kdp,2)))...
    (ones(1, size(kdp,2))) (ones(1, size(kdp,2)))  (ones(1, size(kdp,2)))
(ones(1, size(kdp,2)))...
    (ones(1, size(kdp,2))) (ones(1, size(kdp,2)))  (ones(1, size(kdp,2)))];
scale.axial.contour = [plot.threed.posit.axial plot.threed.posit.axial ...
    plot.threed.posit.axial plot.threed.posit.axial plot.threed.posit.axial...
    plot.threed.posit.axial plot.threed.posit.axial plot.threed.posit.axial
plot.threed.posit.axial...
    plot.threed.posit.axial plot.threed.posit.axial plot.threed.posit.axial];
scale.kdp = [kdp kdp kdp kdp kdp kdp kdp kdp kdp kdp kdp];
scale.bladepassage.contour = (blade.axes.y' *[(ones(1, size(kdp,2))) ...
    (ones(1, size(kdp,2))) (ones(1, size(kdp,2))) (ones(1, size(kdp,2)))...
    (ones(1, size(kdp,2))) (ones(1, size(kdp,2))) (ones(1, size(kdp,2)))...
    (ones(1, size(kdp,2)))   (ones(1, size(kdp,2))) (ones(1, size(kdp,2)))
(ones(1, size(kdp,2)))...
    (ones(1, size(kdp,2)))]);

passage.position.fix = ones(iteration,1) *scale.passage.position;%11x808

plot.theta.contour = kulite.data.theta.posit.corrected'* scale.theta.contour
plot.axial.contour = scale.axial.contour;

%step one, shift the passage posit up to the kulite position on casing
passage.position.corrected = passage.position.fix  - plot.theta.contour ;
%shift contour plot on to the blade

plot.passage.contour = passage.position.corrected - scale.bladepassage.contour


% z direction is flow direction and theta is blade direction


save ('plot.mat')
save('passage.mat')
```

```
save('scale.mat')
save('kdp.mat')
```

Contour Plots

```
clear all
close all
% 2-1-
load passage scale kdp
load('plot.mat')

figure(25)
contourf(scale.axial.contour,passage.position.corrected , scale.kdp, 250,
'Linestyle', 'none')
axis equal,
xlabel('Kulite Axial Position'),
ylabel('Passage Position'),
zlabel('Kulite Data Passage')
% ylim([-1 2]), xlim([-1 1.5])

figure(26)
plot3(scale.axial.contour',passage.position.corrected' ,scale.kdp')
axis equal,
xlabel('Axial Position'),
ylabel('Passage Position'),
zlabel('Pressure (Pa)')


figure(27)
contourf(scale.axial.contour,plot.passage.contour , scale.kdp, 250,
'Linestyle', 'none')
axis equal,
xlabel('Axial Position'),
ylabel('Passage Position'),
zlabel('Pressure (Pa)')
%ylim([-1 2]), xlim([-1 1.5])

figure(28)
plot3(scale.axial.contour',plot.passage.contour' , scale.kdp')
axis equal,
xlabel('Axial Position'),
ylabel('Passage Position'),
zlabel('Pressure (Pa)')


%to find shock angle
%finding peaks for the first 3 kulite kdp's this will be find the shock
shock.peak.one.posit = max(scale.kdp(1,:))
shock.peak.two.posit = max(scale.kdp(2,:))
shock.peak.three.posit = max(scale.kdp(3,:))
shock.peak.one.row = find(scale.kdp(1,:) == shock.peak.one.posit)
shock.peak.two.row =find(scale.kdp(2,:) == shock.peak.two.posit)
shock.peak.three.row = find(scale.kdp(3,:) == shock.peak.three.posit)

figure(29)
for p = 1:1:3
        plot3(scale.axial.contour(p,:)',passage.position.corrected(p,:)' ,
scale.kdp(p,:)')
        hold on
        axis equal,
        xlabel('Axial Position'),
        ylabel('Passage Position'),
        zlabel('Pressure (Pa)')
end
hold on
```

```
plot3(scale.axial.contour(1,shock.peak.one.row(3)),passage.position.corrected(1
,shock.peak.one.row(3)), scale.kdp(1,shock.peak.one.row(3)),'*r')
hold on
plot3(scale.axial.contour(2,shock.peak.two.row(1)),passage.position.corrected(2
,shock.peak.two.row(1)), scale.kdp(2,shock.peak.two.row(1)),'*r')
hold on
plot3(scale.axial.contour(3,shock.peak.three.row(1)),passage.position.corrected
(3,shock.peak.three.row(1)), scale.kdp(3,shock.peak.three.row(1)),'*r')
hold on
line([scale.axial.contour(1,shock.peak.one.row(3)),scale.axial.contour(2,shock.
peak.two.row(1)),scale.axial.contour(3,shock.peak.three.row(1))],...

[passage.position.corrected(1,shock.peak.one.row(3)),passage.position.corrected
(2,shock.peak.two.row(1)),passage.position.corrected(3,shock.peak.three.row(1))
],...
    [scale.kdp(1,shock.peak.one.row(3)), scale.kdp(2,shock.peak.two.row(1)),
scale.kdp(3,shock.peak.three.row(1))])
% now adding blade to figure to visualize shock position and "see" if it
% looks right. ...verdict..it looks right! ha

figure(30)
plot3(scale.axial.contour',passage.position.corrected' , scale.kdp')
hold on
axis equal,
xlabel('Axial Position'),
ylabel('Passage Position'),
zlabel('Pressure (Pa)')

hold on
plot3(scale.axial.contour(1,shock.peak.one.row(3)),passage.position.corrected(1
,shock.peak.one.row(3)), scale.kdp(1,shock.peak.one.row(3)),'*r')
hold on
plot3(scale.axial.contour(2,shock.peak.two.row(1)),passage.position.corrected(2
,shock.peak.two.row(1)), scale.kdp(2,shock.peak.two.row(1)),'*r')
hold on
plot3(scale.axial.contour(3,shock.peak.three.row(1)),passage.position.corrected
(3,shock.peak.three.row(1)), scale.kdp(3,shock.peak.three.row(1)),'*r')
hold on
line([scale.axial.contour(1,shock.peak.one.row(3)),scale.axial.contour(2,shock.
peak.two.row(1)),scale.axial.contour(3,shock.peak.three.row(1))],...

[passage.position.corrected(1,shock.peak.one.row(3)),passage.position.corrected
(2,shock.peak.two.row(1)),passage.position.corrected(3,shock.peak.three.row(1))
],...
    [scale.kdp(1,shock.peak.one.row(3)), scale.kdp(2,shock.peak.two.row(1)),
scale.kdp(3,shock.peak.three.row(1))])
line([0 0.6409], [0 1.5])

angle = (passage.position.corrected(1,shock.peak.one.row(3))- ...

passage.position.corrected(3,shock.peak.three.row(1)))/(abs(scale.axial.contour
(1,shock.peak.one.row(3))))
%plot lines going back in through the blade passage
%interpolate along the blade
temp.pressure = zeros(kulite.length, length(passage.position.raw))

for i = 1: length(passage.position.raw); %101
    for j = 1:kulite.length; %11
[tempx tempi] = unique(passage.position.corrected(j,:)); %pressures
tempy = scale.kdp(j,tempi);
        temp.pressure(j,i) = interp1(tempx, tempy, passage.position.raw(i),
'cubic');%11x101
```

```
    end
end
temp.pressure; %11x101
temp.position =  ones(kulite.length,1) *passage.position.raw
figure(31)
plot3(plot.threed.posit.axial, temp.position, temp.pressure)
xlabel('Axial Position'), ylabel('Passage Position'), zlabel('Pressure (Pa)')


%pretend that we have 75 kulites
fakekulites = 75;
smooth.position = zeros(fakekulites, length(passage.position.raw)); %75x101
smooth.axial = zeros(fakekulites, length(passage.position.raw)); %75x101
smooth.pressure = zeros(fakekulites, length(passage.position.raw));%75x101

%fred_smooth.shock = ones(fakekulites,11) * shock.position.slope; %75x101
smooth.position = ones(fakekulites,1) * passage.position.raw; %75x101
smooth.axial = (linspace(
plot.threed.posit.axial(1,1),plot.threed.posit.axial(end,1),fakekulites))'*
ones(1,length(passage.position.raw))
%75x101
for ii = 1: length(passage.position.raw); %

tempyy = temp.pressure(:,ii); %11x1 after loop 11x101
tempxx = plot.threed.posit.axial(:,ii);%11x1 after loop 11x101

smooth.pressure(:,ii) =interp1(tempxx, tempyy, smooth.axial(:,ii), 'cubic');
ii
end
smooth.pressure; %75x101

figure(32)
plot3( smooth.axial, smooth.position, smooth.pressure)
xlabel('Axial Position'), ylabel('Passage Position'), zlabel('Pressure (Pa)')

%scale the plots
figure(33)
contourf( smooth.axial, smooth.position, smooth.pressure, 250, 'Linestyle',
'none')
xlabel('Axial Position'), ylabel('Passage Position'), zlabel('Pressure (Pa)')
%step one, shift the passage posit up to the kulite position on casing

%shift contour plot on to the blade

bladeshift = (linspace( blade.axes.y(1),blade.axes.y(end),fakekulites))' *
ones(1,length(passage.position.raw));%75x101
smooth.blade.position= smooth.position + bladeshift

figure(34)
contourf( smooth.axial, smooth.blade.position, smooth.pressure, 250,
'Linestyle', 'none')
xlabel('Axial Position'), ylabel('Passage Position'), axis equal

shock.passage.contour(1,:) = passage.position.corrected(1,:)-
(passage.position.corrected(1,shock.peak.one.row(3))-
passage.position.corrected(3,shock.peak.three.row(1)))
shock.passage.contour(2,:) = passage.position.corrected(2,:)-
(passage.position.corrected(2,shock.peak.two.row(1))-
passage.position.corrected(3,shock.peak.three.row(1)))
shock.passage.contour(3,:) = passage.position.corrected(3,:)
shock.passage.contour(4,:) = passage.position.corrected(4,:)
```

```
shock.passage.contour(5,:) = passage.position.corrected(5,:)
shock.passage.contour(6,:) = passage.position.corrected(6,:)
shock.passage.contour(7,:) = passage.position.corrected(7,:)
shock.passage.contour(8,:) = passage.position.corrected(8,:)
shock.passage.contour(9,:) = passage.position.corrected(9,:)
shock.passage.contour(10,:) = passage.position.corrected(10,:)


figure(35)
plot3(scale.axial.contour',shock.passage.contour' , scale.kdp')
hold on
axis equal,
xlabel('Axial Position'), ylabel('Passage Position'), zlabel('Pressure (Pa)')
hold on
line([scale.axial.contour(1,shock.peak.one.row(3)),scale.axial.contour(2,shock.
peak.two.row(1)),scale.axial.contour(3,shock.peak.three.row(1))],...

[shock.passage.contour(1,shock.peak.one.row(3)),shock.passage.contour(2,shock.p
eak.two.row(1)),shock.passage.contour(3,shock.peak.three.row(1))],...
    [scale.kdp(1,shock.peak.one.row(3)), scale.kdp(2,shock.peak.two.row(1)),
scale.kdp(3,shock.peak.three.row(1))])


%interp1 along the shock
for iii = 1: length(passage.position.raw); %
  for jjj = 1:kulite.length
[tempxxx tempiii] = unique(shock.passage.contour(jjj,:)); %pressures
tempyyy = scale.kdp(jjj,tempiii);
        shock.pressure(jjj,iii) = interp1(tempxxx, tempyyy,
passage.position.raw(iii), 'cubic'); %11x101

    end
end

final.shock.position = zeros(fakekulites, length(passage.position.raw));
%75x101
final.shock.axial = zeros(fakekulites, length(passage.position.raw)); %75x101
final.shock.pressure = zeros(fakekulites, length(passage.position.raw));%75x101

final.shock.position = ones(1,fakekulites)'* temp.position(1,:)   %75x101
final.shock.axial = (linspace(plot.threed.posit.axial(1,1),
plot.threed.posit.axial(kulite.length,1),fakekulites)'*
ones(1,length(passage.position.raw)))
%75x101,
for iiii = 1: length(passage.position.raw); %

tempyyyy =  shock.pressure(:,iiii)%75x1 after loop 75x101
tempxxxx = plot.threed.posit.axial(:,iiii);%11x1 after loop 11x101

final.shock.pressure(:,iiii) =interp1(tempxxxx, tempyyyy,
final.shock.axial(:,iiii), 'cubic');
iiii
end

figure(36)
plot3( final.shock.axial, final.shock.position, final.shock.pressure)
xlabel('Axial Position'), ylabel('Passage Position'), zlabel('Pressure (Pa)')

%
%interp1 along the blade passages
for iiia = 1: length(passage.position.raw); %
  for jjja = 1:kulite.length
```

```
[tempxxxa tempiiia] = unique(plot.passage.contour(jjja,:)); %pressures
tempyyya = scale.kdp(jjja,tempiiia);
        blade.pressure(jjja,iiia) = interp1(tempxxxa, tempyyya,
passage.position.raw(iiia), 'cubic'); %11x101
    iiia
    end
end

final.blade.position = zeros(fakekulites, length(passage.position.raw));
%75x101
final.blade.axial = zeros(fakekulites, length(passage.position.raw)); %75x101
final.blade.pressure = zeros(fakekulites, length(passage.position.raw));%75x101

final.blade.position = smooth.position   %75x101
final.blade.axial = (linspace(scale.axial.contour(1,1),
scale.axial.contour(kulite.length,1),fakekulites)'*
ones(1,length(passage.position.raw)))
%75x101,
for iiiia = 1: length(passage.position.raw); %

tempyyyya =  blade.pressure(:,iiiia)%75x1 after loop 75x101
tempxxxxa = scale.axial.contour(:,iiiia);%11x1 after loop 11x101

final.blade.pressure(:,iiiia) =interp1(tempxxxxa, tempyyyya,
final.blade.axial(:,iiiia), 'cubic');
iiiia
end

figure(37)
plot3( final.blade.axial, final.blade.position, final.blade.pressure)
xlabel('Passage Position'), ylabel('Pressure (Pa)')
%
figure(38)
contourf( final.blade.axial, final.blade.position, final.blade.pressure,250,
'Linestyle', 'none')
xlabel('Passage Position'), ylabel('Pressure (Pa)')

figure(39)
contourf( final.blade.axial, final.blade.position, final.blade.pressure,250,
'Linestyle', 'none')
xlabel('Passage Position'), ylabel('Pressure(Pa)'), axis equal



figure(40)
contourf( final.shock.axial, final.shock.position, final.shock.pressure, 250,
'Linestyle', 'none')
xlabel('Passage Position'), ylabel('Pressure (Pa)')

figure(41)
contourf( final.shock.axial, final.shock.position, final.shock.pressure, 250,
'Linestyle', 'none')
xlabel('Passage Position'), ylabel('Pressure (Pa)'), axis equal

%shift the shock back to its orginal angle, using ployfit and ployval to create
a  curved shock instead or a straight shock:

X = [scale.axial.contour(1,shock.peak.one.row(3))
scale.axial.contour(2,shock.peak.two.row(1))
scale.axial.contour(3,shock.peak.three.row(1))]
Y = [(passage.position.corrected(1,shock.peak.one.row(3))-
passage.position.corrected(3,shock.peak.three.row(1)))...
```

```
    (passage.position.corrected(2,shock.peak.two.row(1))-
passage.position.corrected(3,shock.peak.three.row(1)))...
    passage.position.corrected(3,shock.peak.three.row(1))-
passage.position.corrected(3,shock.peak.three.row(1))]

fzeros = zeros(75, 1)
P = polyfit (X,Y,2)
x = final.shock.axial(:,1)
f = polyval(P,x)

d = le(f, fzeros)
f(d) =   0

interm.position= f *ones(1, length(passage.position.raw))
final.plot.position=interm.position+final.shock.position
figure(42)
plot3( final.shock.axial, final.plot.position, final.shock.pressure)
xlabel('Axial Position'), ylabel('Passage Position'), zlabel('Pressure (Pa)')

figure(43)
contourf( final.shock.axial, final.plot.position, final.shock.pressure, 250,
'Linestyle', 'none')
xlabel('Passage Position'), ylabel('Pressure (Pa)')

%shifting the blade back up to its original angle
interm.blade.position = (linspace(scale.bladepassage.contour(1,1),
scale.bladepassage.contour(kulite.length,1),fakekulites)'*
ones(1,length(passage.position.raw)))
final.plot.blade.position = final.blade.position +  interm.blade.position
figure(44)
contourf( final.blade.axial, final.plot.blade.position ,
final.blade.pressure,250, 'Linestyle', 'none')
xlabel('Passage Position'), ylabel('Pressure (Pa)'), axis equal

%add the shock shift and the blade shift together
final.position =  zeros(fakekulites, length(passage.position.raw))
final.position(1:27,:) = final.plot.position(1:27,:)
final.position(28:75,:) = final.plot.blade.position(28:75,:)

final.pressure =  zeros(fakekulites, length(passage.position.raw))
final.pressure(1:27,:) = final.shock.pressure(1:27,:)
final.pressure(28:75,:) = final.blade.pressure(28:75,:)

final.axial = final.shock.axial

figure(45)
contourf( final.axial, final.position , final.pressure,250, 'Linestyle',
'none')
xlabel('Passage Position'), ylabel('Pressure (Pa)'), axis equal

%now you have to add passages
final.scale.position = [(final.position- 3.01) (final.position- 2.01)
(final.position- 1.01) final.position  (final.position+ 1.01) (final.position+
2.01)]
final.scale.axial = [final.axial final.axial final.axial final.axial
final.axial final.axial]
final.scale.pressure = [final.pressure final.pressure final.pressure
final.pressure final.pressure final.pressure]

figure(46)
contourf( final.scale.axial, final.scale.position, final.scale.pressure, 250,
'Linestyle', 'none')
```

```
xlabel('Passage Position'), ylabel('Pressure (Pa)'), axis equal, xlim([-.6409
1]),ylim([0 3])

figure(47)
contourf( final.scale.axial, final.scale.position, final.scale.pressure, 250,
'Linestyle', 'none')
xlabel('Passage Position'), ylabel('Pressure (Pa)'), axis equal, xlim([-.6409
1]),ylim([0 3])
h= [line([0 0.6409], [-0.5 1]), line([0 0.6409], [0.5 2]),line([0 0.6409], [1.5
3]),line([0 0.6409], [2.5 4])]
set(h,'LineWidth',3),set(h,'Color',[0 0 0])
```

Main Analyze 2 FFT

```
close all
clear all
load('kulite.mat')
load('pressure.mat')
load('blade.mat')
load('contourdata.mat')
load('loc.mat')
load('volts.mat')
tic
% Single Kulite Plots
%assign storage for each vector
count.lines= 17000;%lines
count.dots=100;%dots

passage.position.raw = zeros(count.lines, count.dots+1);
passage.position.raw = linspace(0,1,count.dots +1);

passage.position.data = zeros(count.lines, count.dots+1, kulite.length);
for iteration = 1:kulite.length
iteration

%fft.data.raw = zeros(count.lines, count.dots+1, iteration);

for cycles=1:count.lines; %number of cycles
% [iteration, cycles]
    range1 = find(kulite.position.correct > (cycles - 0.25) ,1,'first');
    range2 = find(kulite.position.correct < ((cycles+1)+ 0.25),1,'last');
%
%     h(cycles) = plot(kulite.position.correct(range(cycles):range(cycles+1))-
cycles, ...
%                 pressure.final(range(cycles):range(cycles+1),iteration));
        tempx = kulite.position.correct(range1:range2)-cycles;
        tempy= pressure.final(range1:range2,iteration);

           for points = 1: count.dots + 1
%            passage.position.data(cycles, points, iteration) =
interp1(get(h(cycles), 'Xdata'),...
%                get(h(cycles),'Ydata'), passage.position.raw(points));
passage.position.data(cycles, points, iteration) = interp1(tempx,...
            tempy, passage.position.raw(points));

           end
           %set(h(cycles), 'ZData',(cycles*(ones(size(get(h(cycles),
'XData'))))));
           %hold on

end

%fft.data.raw= passage.position.data
toc
end

save fft passage
```

FFT Test Case

```
close all
clear all
load fft
tic

NFFTN = 30; % Number of factors of zero padding to improve the peak magnitude
and location
Time = (1: size(passage.position.data,1))/20;%101x1
NFFsize = NFFTN*(2^nextpow2(length(Time)))/2+1;
%amplitude.raw= zeros(NFFsize,101);
% amplitude.raw= zeros(length(runnumber), length(runtime));
for  runnumber = 1:10% each kulite

% Some arbitrary data
%assign the variable in main analyze 2 that tells you where 101 comes from
    for runtime = 1:101 % there are a total of 101 samples within each kulite
y1   =  passage.position.data(:,runtime,runnumber);
y1   =  y1 - mean(y1);

%y1   = (1*sin(2*pi*1000*Time) + 0*sin(2*pi*1500*Time) + 0*sin(2*pi*1600*Time)
+ 0*sin(2*pi*1900*Time)); % [1000 1300 1600 1900] Hz signal with unity
amplitude
y2   = y1.*blackman(length(y1));             % Blackman window is applied but
just to show what is being done, not needed for dBCalc

%f    = Fs/2*linspace(0,1,NFFT/2+1); % Half of frequency spectrum

[Y2, Fs, f] = FFTCalc(Time, y1, NFFTN);
Fs; % Sampling frequency
%amplitude.raw(:, runtime) = Y2;
amplitude.raw = Y2;
freq = find(f > 0.25 & f < 1.5) ;%1x98304 finds the index not the actual number
0.1 < f &

amplitude.raw= amplitude.raw(freq,:) ;
f = f(freq);


runnumber
runtime

fft_results = ['tcrresults' num2str(runnumber) 'runtime' num2str(runtime)]

toc

save(fft_results,'amplitude','f','-v7.3')

    end


 end %delete this "end" when adding plots
------------------------
function [Y2, Fs, f] = FFTCalc(Time, y1, NFFTN)
% dBCalc
% Time:  Time stamp of samples
% y1:  Pressure level at each time stamp
%
% NFFT: Increases the number of times the FFT is padded with zeros to improve
% freq domain calculations
%
```

```
% A blackman window is used to condition the
% data before passing it to the frequency domain.

NFFT  = NFFTN*(2^nextpow2(length(Time))); % Next power of 2 from length of y
Fs    = (length(Time)-1)/(Time(end)-Time(1)); % Sampling frequency
f     = Fs/2*linspace(0,1,NFFT/2+1);         % Half of frequency

% Time domain calculation
y1   = y1 - mean(y1);
y2   = y1.*blackman(length(y1));     % Blackman window is applied
Y1   = fft(y1,NFFT)/length(Time);  % Fast Fourier transform, raw data with zero
padding
Y2   = fft(y2,NFFT)/length(Time);  % Fast Fourier transform, blackman corrected
data with zero padding
Y2   = Y2*max(abs(Y1))/max(abs(Y2)); % Power is corrected to take into account
the Blackman window

% Only first half of domain is needed so power needs to be doubled
Y1 = 2*Y1(1:NFFT/2+1);
Y2 = 2*Y2(1:NFFT/2+1);

end
```

FFT Analyze

```
clear all
close all
% amp.max = zeros(10,101);
% amp.maxfreq = zeros(10,101);
% amp.abs=zeros(93388,101);

tic
for runnumber = 1:10

%size93388x101

for runtime = 1:101
       fft_results = ['tcrresults' num2str(runnumber) 'runtime'
num2str(runtime)]
load(fft_results)
pause(2)
% freq = find( f < 2) ;%1x98304 finds the index not the actual number  0.1 < f
&
temp= amplitude.raw ;   %
[runnumber runtime]

[amp.max(:,runtime), amp.loc(:,runtime)] = max(abs(temp)); %size(1x101)
%(runnumber,runtime),(:,runtime)

amp.maxfreq= f(amp.loc);

amp.phase = imag(temp(amp.loc,:));%size1x101

toc
pause(1)
end
amp_results = ['runnumber' num2str(runnumber)]
save(amp_results,'amp','-v7.3')
toc
end
clear all
close all
%zeros(length(runnumber),length(amp.max))

newamp.max = zeros(10,101);
newamp.maxfreq = zeros(10,101);
newamp.phase = zeros(10,101);

for runnumber = 1:10
amp_results = ['runnumber' num2str(runnumber)]
load(amp_results)

newamp.max(runnumber,:) = amp.max

newamp.maxfreq(runnumber, :) = amp.maxfreq

newamp.phase(runnumber, :) = amp.phase

end
newamp;
%% For freq of Interest
clear all
close all

for runnumber = 1:10
```

```
freqint_results = ['DomFreqRunnumber' num2str(runnumber)]
load(freqint_results)

NewLowDomFreq.index(runnumber,:)  = LowDomFreq.index;
NewLowDomFreq.amplitude(runnumber,:) = LowDomFreq.amplitude;
NewLowDomFreq.check(runnumber,:)= LowDomFreq.check;
NewLowDomFreq.phase(runnumber,:)= LowDomFreq.phase;

NewRotorDomFreq.index(runnumber,:)= RotorDomFreq.index;
NewRotorDomFreq.amplitude(runnumber,:)= RotorDomFreq.amplitude ;
NewRotorDomFreq.check(runnumber,:) = RotorDomFreq.check ;
NewRotorDomFreq.phase(runnumber,:)= RotorDomFreq.phase;
end

save('NewDomFreq_results','NewLowDomFreq','NewRotorDomFreq','-v7.3')

%% used to find the maximums in each matrix
[y, i]=max(newamp.max,[],2);
 for ii=   1:10
freq.dominant(ii)= newamp.maxfreq(ii,i(ii))

 end

[y, x]=max(newamp.max,[],2);
  for ii=   1:10
freq.amplitude(ii) = newamp.max(ii,x(ii))
  end

[freq.max freq.index] = max(freq.amplitude)
freq.freq = freq.dominant(freq.index)
save('freq_interest', 'freq')

%% for freq contour plots
clear all
close all
load passage scale kdp
load('plot.mat')
load NewDomFreq_results
%NewLowDomFreq.phase = unwrap(angle(NewRotorDomFreq.amplitude),[],2)
NewLowDomFreq.phase = NewLowDomFreq.phase
plotamp = [NewLowDomFreq.phase NewLowDomFreq.phase NewLowDomFreq.phase...
    NewLowDomFreq.phase NewLowDomFreq.phase NewLowDomFreq.phase...
   NewLowDomFreq.phase NewLowDomFreq.phase NewLowDomFreq.phase...
   NewLowDomFreq.phase NewLowDomFreq.phase NewLowDomFreq.phase]
%plotamp = log(plotamp);

%plot lines going back in through the blade passage
%interpolate along the blade
temp.pressure = zeros(kulite.length, length(passage.position.raw))

for i = 1: length(passage.position.raw); %101
    for j = 1:kulite.length; %11
[tempx tempi] = unique(passage.position.corrected(j,:)); %pressures
tempy = plotamp(j,tempi);
        temp.pressure(j,i) = interp1(tempx, tempy, passage.position.raw(i),
'cubic');%11x101
    %temp.pressure(j,i) = nearest(tempy);%11x101
    end
end
temp.pressure; %11x101
temp.position =  ones(kulite.length,1) *passage.position.raw
figure(31)
```

```
plot3(plot.threed.posit.axial, temp.position, temp.pressure)
xlabel('passage contour'), ylabel('kdp')


%pretend that we have 75 kulites
fakekulites = 75;
smooth.position = zeros(fakekulites, length(passage.position.raw)); %75x101
smooth.axial = zeros(fakekulites, length(passage.position.raw)); %75x101
smooth.pressure = zeros(fakekulites, length(passage.position.raw));%75x101

%75x101
smooth.position = ones(fakekulites,1) * passage.position.raw; %75x101
smooth.axial = (linspace(
plot.threed.posit.axial(1,1),plot.threed.posit.axial(end,1),fakekulites))'*
ones(1,length(passage.position.raw))
%75x101
for ii = 1: length(passage.position.raw); %

tempyy = temp.pressure(:,ii); %11x1 after loop 11x101
tempxx = plot.threed.posit.axial(:,ii);%11x1 after loop 11x101

smooth.pressure(:,ii) =interp1(tempxx, tempyy, smooth.axial(:,ii), 'cubic');
ii
end
smooth.pressure; %75x101

figure(31)
plot3( smooth.axial, smooth.position, smooth.pressure)
xlabel('passage contour'), ylabel('kdp')

%scale the plots
figure(32)
contourf( smooth.axial, smooth.position, smooth.pressure, 250, 'Linestyle',
'none')
xlabel('passage contour'), ylabel('kdp')
%step one, shift the passage posit up to the kulite position on casing

%shift contour plot on to the blade

bladeshift = (linspace( blade.axes.y(1),blade.axes.y(end),fakekulites))' *
ones(1,length(passage.position.raw));%75x101
smooth.blade.position= smooth.position + bladeshift

figure(33)
contourf( smooth.axial, smooth.blade.position, smooth.pressure, 250,
'Linestyle', 'none')
xlabel('passage contour'), ylabel('kdp'), axis equal
load fftshockbend
shock.passage.contour(1,:) = passage.position.corrected(1,:)- Y(1)
shock.passage.contour(2,:) = passage.position.corrected(2,:)- Y(2)
shock.passage.contour(3,:) = passage.position.corrected(3,:)
shock.passage.contour(4,:) = passage.position.corrected(4,:)
shock.passage.contour(5,:) = passage.position.corrected(5,:)
shock.passage.contour(6,:) = passage.position.corrected(6,:)
shock.passage.contour(7,:) = passage.position.corrected(7,:)
shock.passage.contour(8,:) = passage.position.corrected(8,:)
shock.passage.contour(9,:) = passage.position.corrected(9,:)
shock.passage.contour(10,:) = passage.position.corrected(10,:)


%interp1 along the shock
for iii = 1: length(passage.position.raw); %
```

```
  for jjj = 1:kulite.length
[tempxxx tempiii] = unique(shock.passage.contour(jjj,:)); %pressures
tempyyy = plotamp(jjj,tempiii);
        shock.pressure(jjj,iii) = interp1(tempxxx, tempyyy,
passage.position.raw(iii), 'cubic'); %11x101

    end
end

final.shock.position = zeros(fakekulites, length(passage.position.raw));
%75x101
final.shock.axial = zeros(fakekulites, length(passage.position.raw)); %75x101
final.shock.pressure = zeros(fakekulites, length(passage.position.raw));%75x101

final.shock.position = ones(1,fakekulites)'* temp.position(1,:)   %75x101
final.shock.axial = (linspace(plot.threed.posit.axial(1,1),
plot.threed.posit.axial(kulite.length,1),fakekulites)'*
ones(1,length(passage.position.raw)))
%75x101,
for iiii = 1: length(passage.position.raw); %

tempyyyy =  shock.pressure(:,iiii)%75x1 after loop 75x101
tempxxxx = plot.threed.posit.axial(:,iiii);%11x1 after loop 11x101

final.shock.pressure(:,iiii) =interp1(tempxxxx, tempyyyy,
final.shock.axial(:,iiii), 'cubic');
iiii
end

figure(35)
plot3( final.shock.axial, final.shock.position, final.shock.pressure)
xlabel('passage contour'), ylabel('kdp')

%
%interp1 along the blade passages
for iiia = 1: length(passage.position.raw); %
  for jjja = 1:kulite.length
[tempxxxa tempiiia] = unique(plot.passage.contour(jjja,:)); %pressures
tempyyya = plotamp(jjja,tempiiia);
        blade.pressure(jjja,iiia) = interp1(tempxxxa, tempyyya,
passage.position.raw(iiia), 'cubic'); %11x101
    iiia
    end
end

final.blade.position = zeros(fakekulites, length(passage.position.raw));
%75x101
final.blade.axial = zeros(fakekulites, length(passage.position.raw)); %75x101
final.blade.pressure = zeros(fakekulites, length(passage.position.raw));%75x101

final.blade.position = smooth.position   %75x101
final.blade.axial = (linspace(scale.axial.contour(1,1),
scale.axial.contour(kulite.length,1),fakekulites)'*
ones(1,length(passage.position.raw)))
%75x101,
for iiiia = 1: length(passage.position.raw); %

tempyyyya =  blade.pressure(:,iiiia)%75x1 after loop 75x101
tempxxxxa = scale.axial.contour(:,iiiia);%11x1 after loop 11x101

final.blade.pressure(:,iiiia) =interp1(tempxxxxa, tempyyyya,
final.blade.axial(:,iiiia), 'cubic');
```

```
iiiia
end

figure(24)
plot3( final.blade.axial, final.blade.position, final.blade.pressure)
xlabel('passage contour'), ylabel('kdp')
%
figure(23)
contourf( final.blade.axial, final.blade.position, final.blade.pressure,250,
'Linestyle', 'none')
xlabel('passage contour'), ylabel('kdp')

figure(22)
contourf( final.blade.axial, final.blade.position, final.blade.pressure,250,
'Linestyle', 'none')
xlabel('passage contour'), ylabel('kdp'), axis equal

figure(36)
contourf( final.shock.axial, final.shock.position, final.shock.pressure, 250,
'Linestyle', 'none')
xlabel('passage contour'), ylabel('kdp')

figure(37)
contourf( final.shock.axial, final.shock.position, final.shock.pressure, 250,
'Linestyle', 'none')
xlabel('passage contour'), ylabel('kdp'), axis equal

%shift the shock back to its orginal angle, using ployfit and ployval to create
a  curved shock instead or a straight shock:

fzeros = zeros(75, 1)
P = polyfit (X,Y,2)
x = final.shock.axial(:,1)
f = polyval(P,x)

d = le(f, fzeros)
f(d) =   0

interm.position= f *ones(1, length(passage.position.raw))
final.plot.position=interm.position+final.shock.position
figure(38)
plot3( final.shock.axial, final.plot.position, final.shock.pressure)
xlabel('Axial Position'), ylabel('Passage Position'), zlabel('Pressure')

figure(39)
contourf( final.shock.axial, final.plot.position, final.shock.pressure, 250,
'Linestyle', 'none')
xlabel('passage contour'), ylabel('kdp')

%shifting the blade back up to its orginal angle
interm.blade.position = (linspace(scale.bladepassage.contour(1,1),
scale.bladepassage.contour(kulite.length,1),fakekulites)'*
ones(1,length(passage.position.raw)));
final.plot.blade.position = final.blade.position +  interm.blade.position;
figure(40)
contourf( final.blade.axial, final.plot.blade.position ,
final.blade.pressure,250, 'Linestyle', 'none')
xlabel('passage contour'), ylabel('kdp'), axis equal

%add the shock shift and the blade shift together
final.position =  zeros(fakekulites, length(passage.position.raw));
final.position(1:27,:) = final.plot.position(1:27,:);
```

```
final.position(28:75,:) = final.plot.blade.position(28:75,:);

final.pressure =  zeros(fakekulites, length(passage.position.raw));
final.pressure(1:27,:) = final.shock.pressure(1:27,:);
final.pressure(28:75,:) = final.blade.pressure(28:75,:);

final.axial = final.shock.axial

figure(41)
contourf( final.axial, final.position , final.pressure,250, 'Linestyle',
'none')
xlabel('passage contour'), ylabel('kdp'), axis equal

%now you have to add passages
final.scale.position = [(final.position- 3.01) (final.position- 2.01)
(final.position- 1.01) final.position  (final.position+ 1.01) (final.position+
2.01)]
final.scale.axial = [final.axial final.axial final.axial final.axial
final.axial final.axial]
final.scale.pressure = [final.pressure final.pressure final.pressure
final.pressure final.pressure final.pressure]

figure(42)
contourf( final.scale.axial, final.scale.position, final.scale.pressure, 250,
'Linestyle', 'none')
xlabel('passage contour'), ylabel('kdp'), axis equal, xlim([-.6409 1]),ylim([0
3]),colorbar
h= [line([0 0.6409], [-0.5 1]), line([0 0.6409], [0.5 2]),line([0 0.6409], [1.5
3]),line([0 0.6409], [2.5 4])]
set(h,'LineWidth',3),set(h,'Color',[0 0 0])

figure(43)
[C,u]=contour( final.scale.axial, final.scale.position, final.scale.pressure)
xlabel('passage contour'), ylabel('kdp'), axis equal, xlim([-.6409 1]),ylim([0
3]),colorbar
h= [line([0 0.6409], [-0.5 1]), line([0 0.6409], [0.5 2]),line([0 0.6409], [1.5
3]),line([0 0.6409], [2.5 4])]
set(h,'LineWidth',3),set(h,'Color',[0 0 0])
text_handle = clabel(C,u)
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. FIGURES AND PLOTS

Contour figures for 70%, 80%, 85%, and 90% are shown below from peak efficiency, close to stall, and near-stall

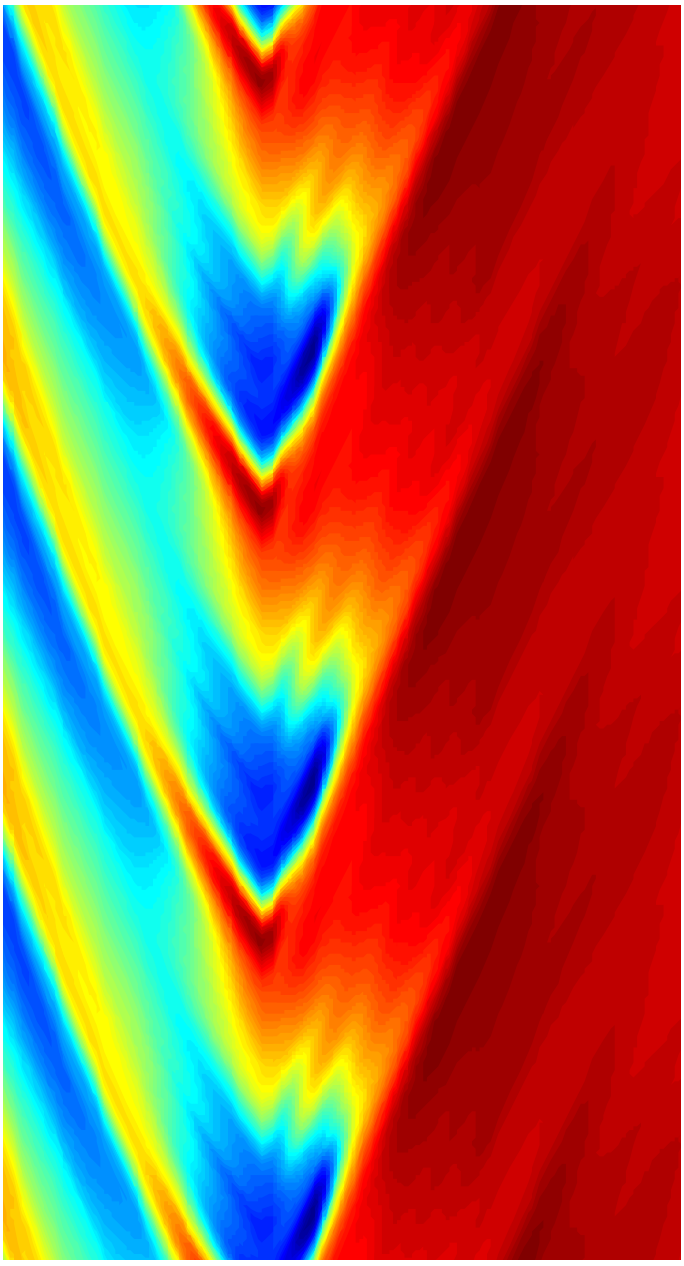


Figure 51. Peak Efficiency: 70%

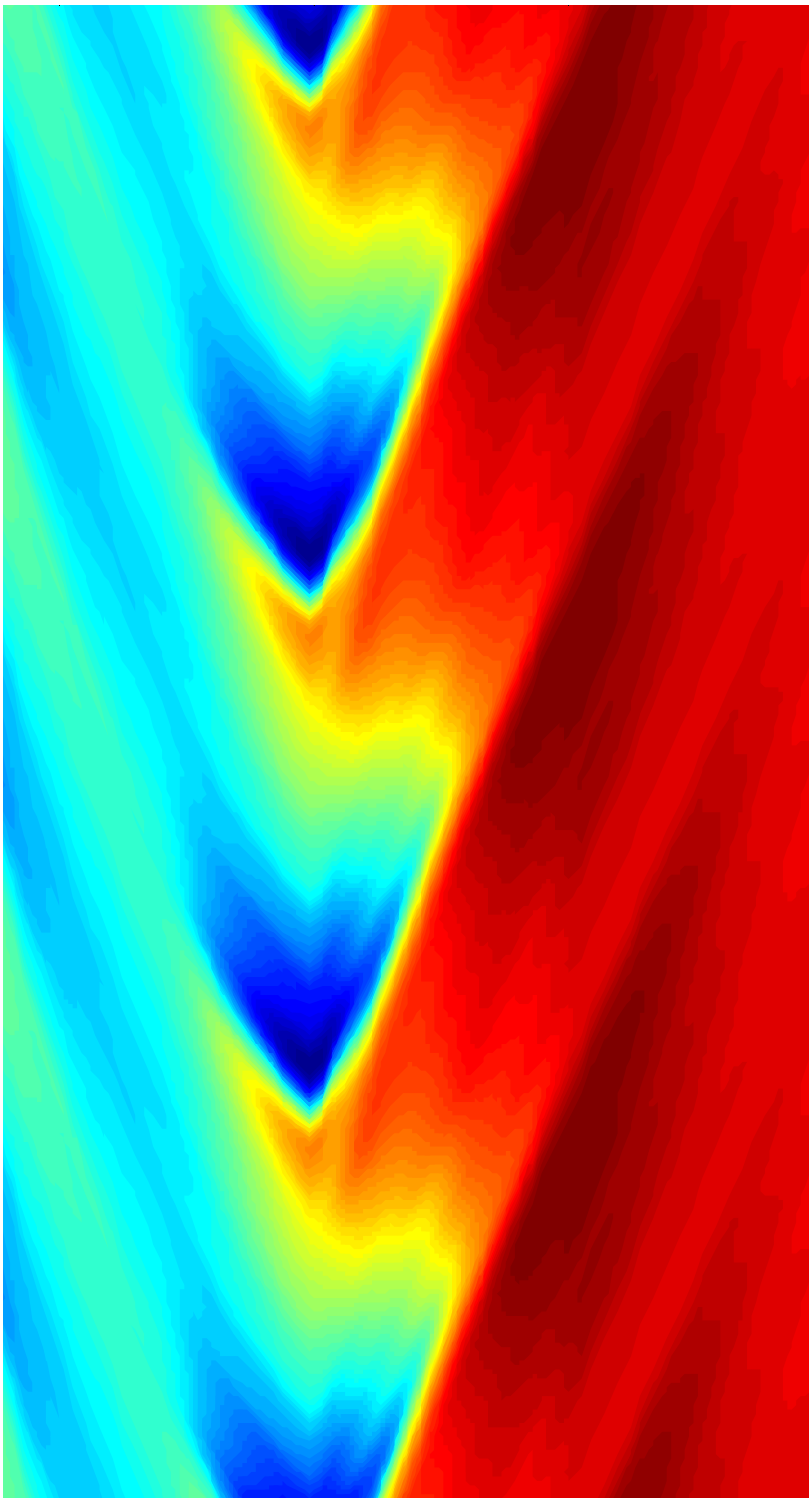


Figure 52. Close to Stall: 70%

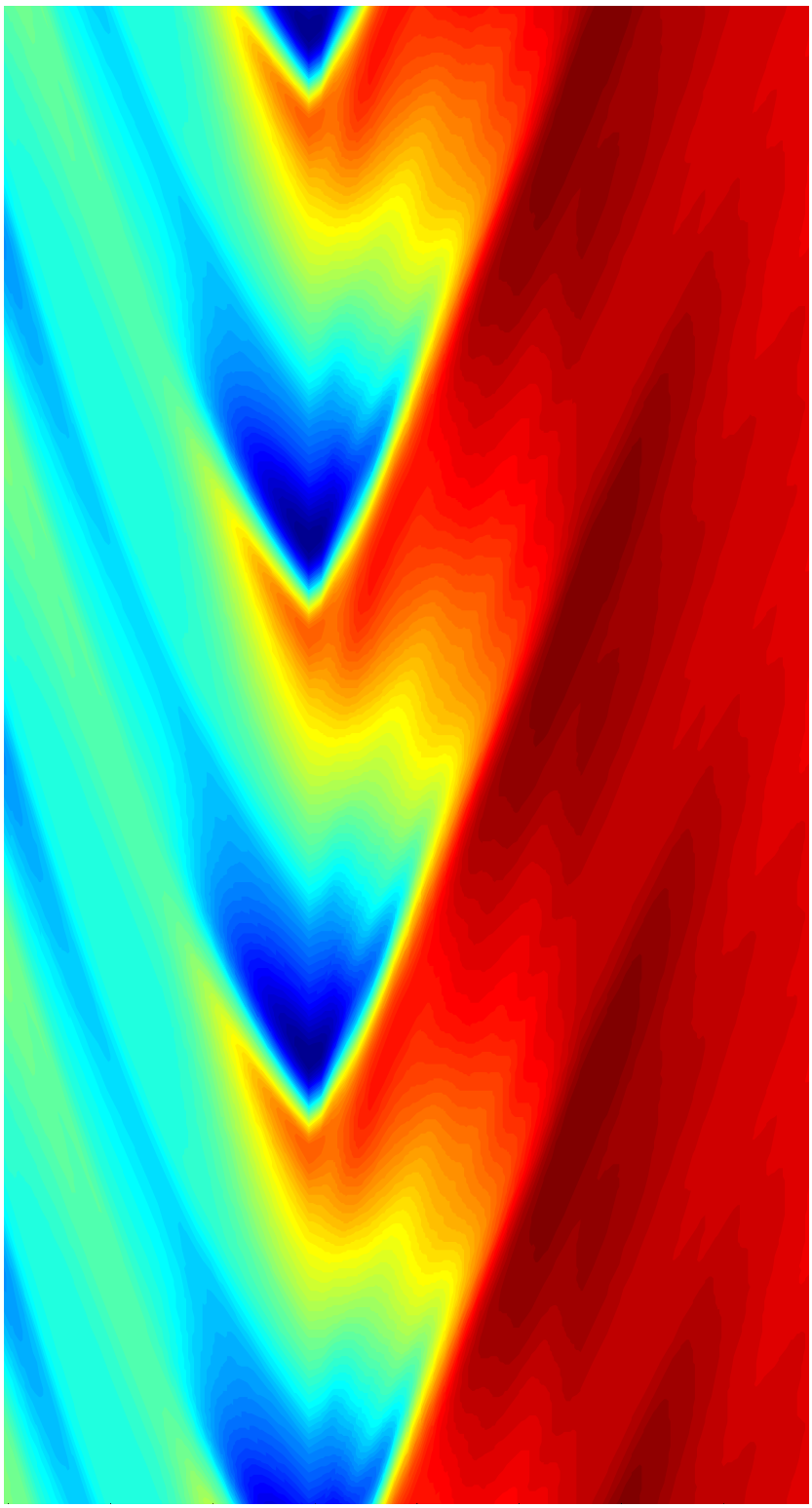


Figure 53. Near-Stall: 70%

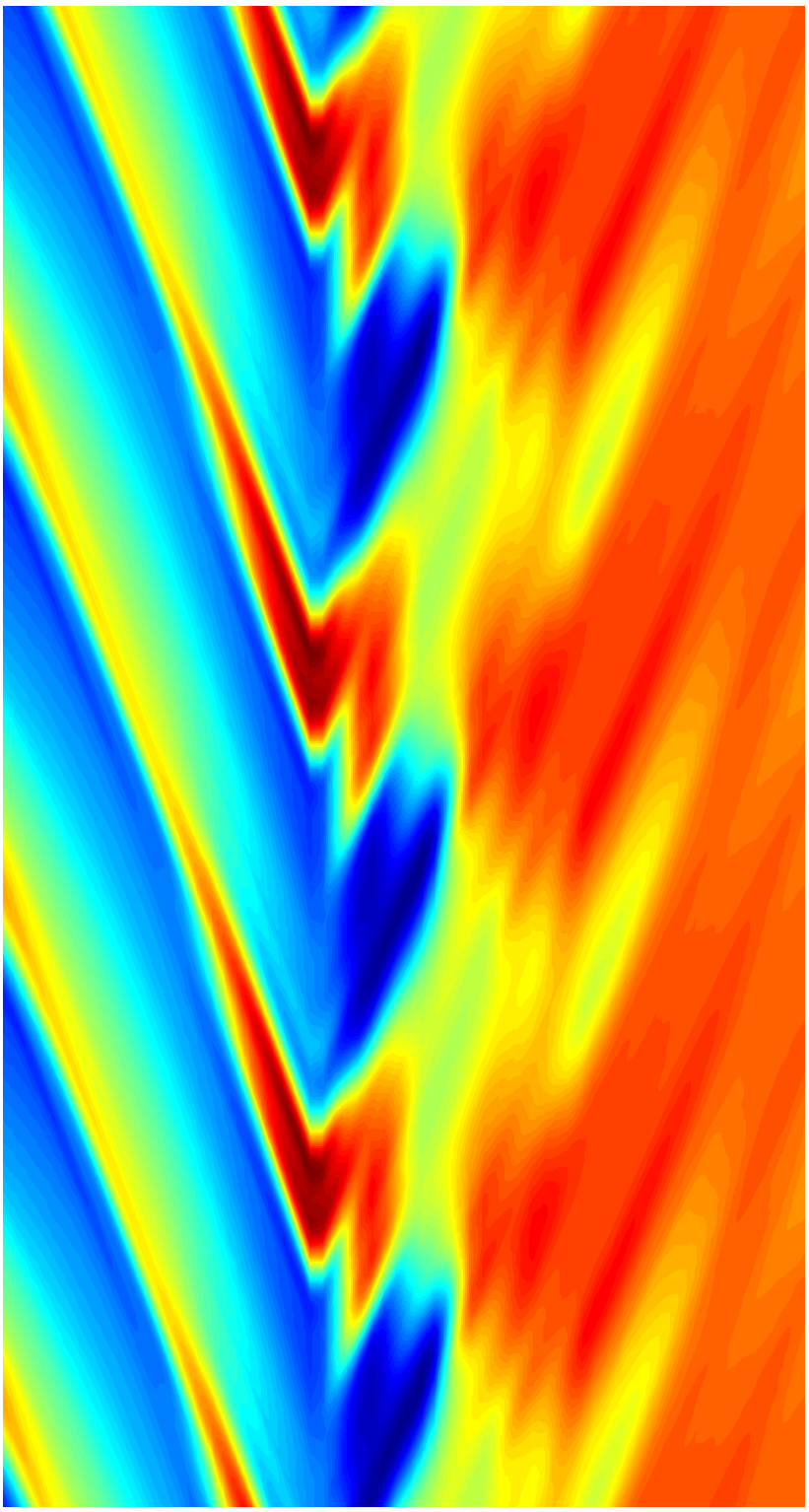


Figure 54. Peak Efficiency: 80%

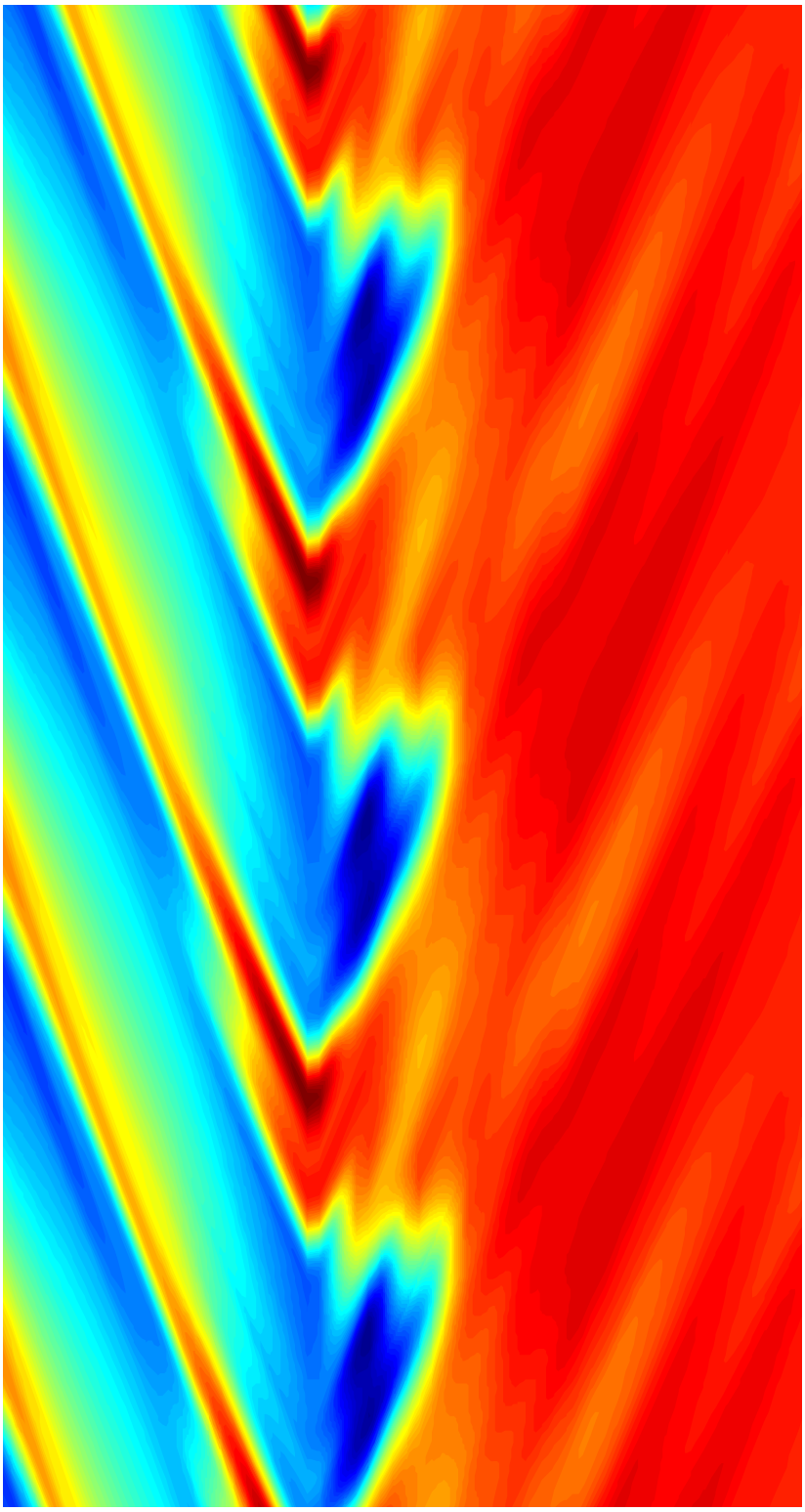


Figure 55. Close to Stall: 80%

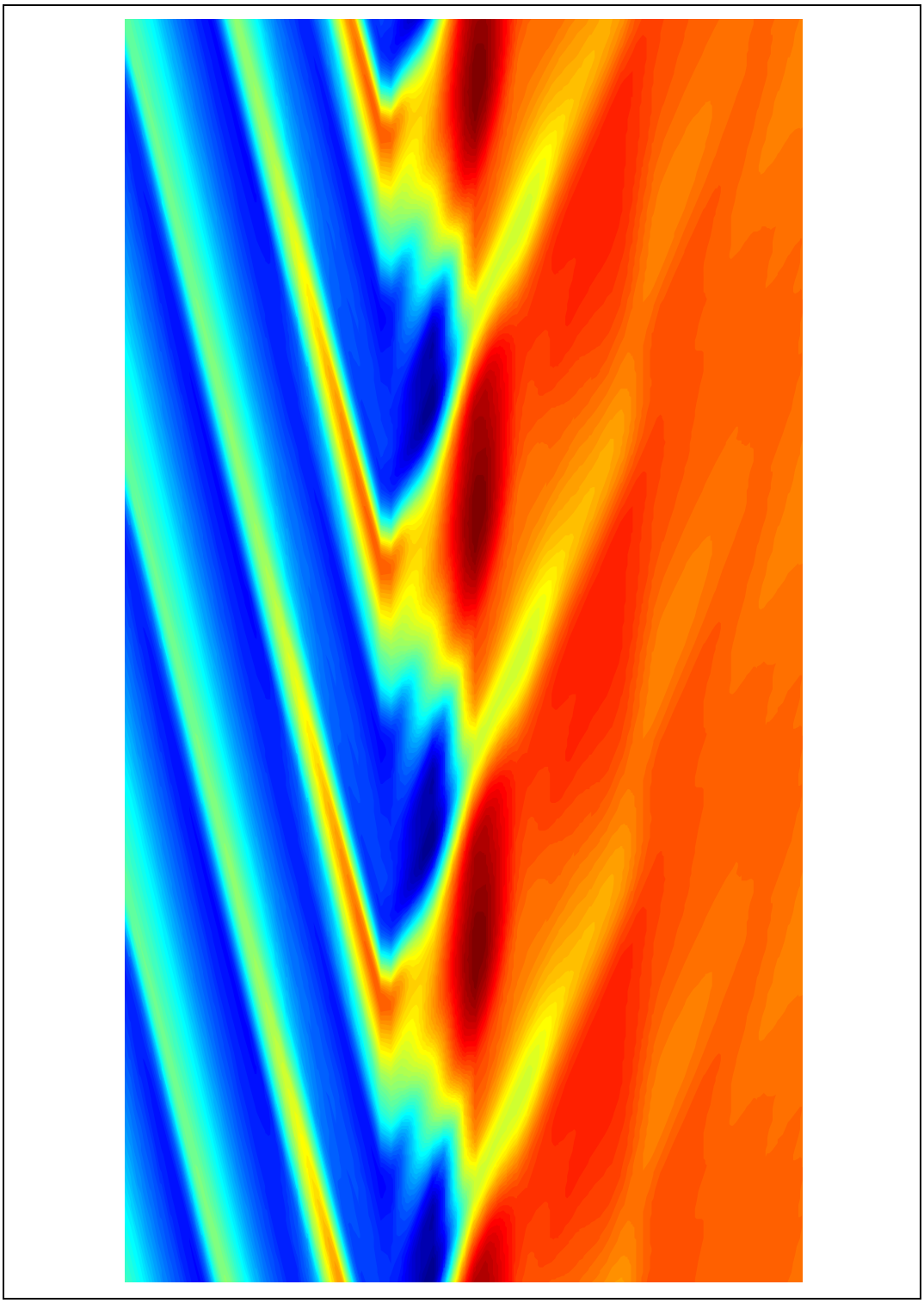


Figure 56. Near Stall: 80%

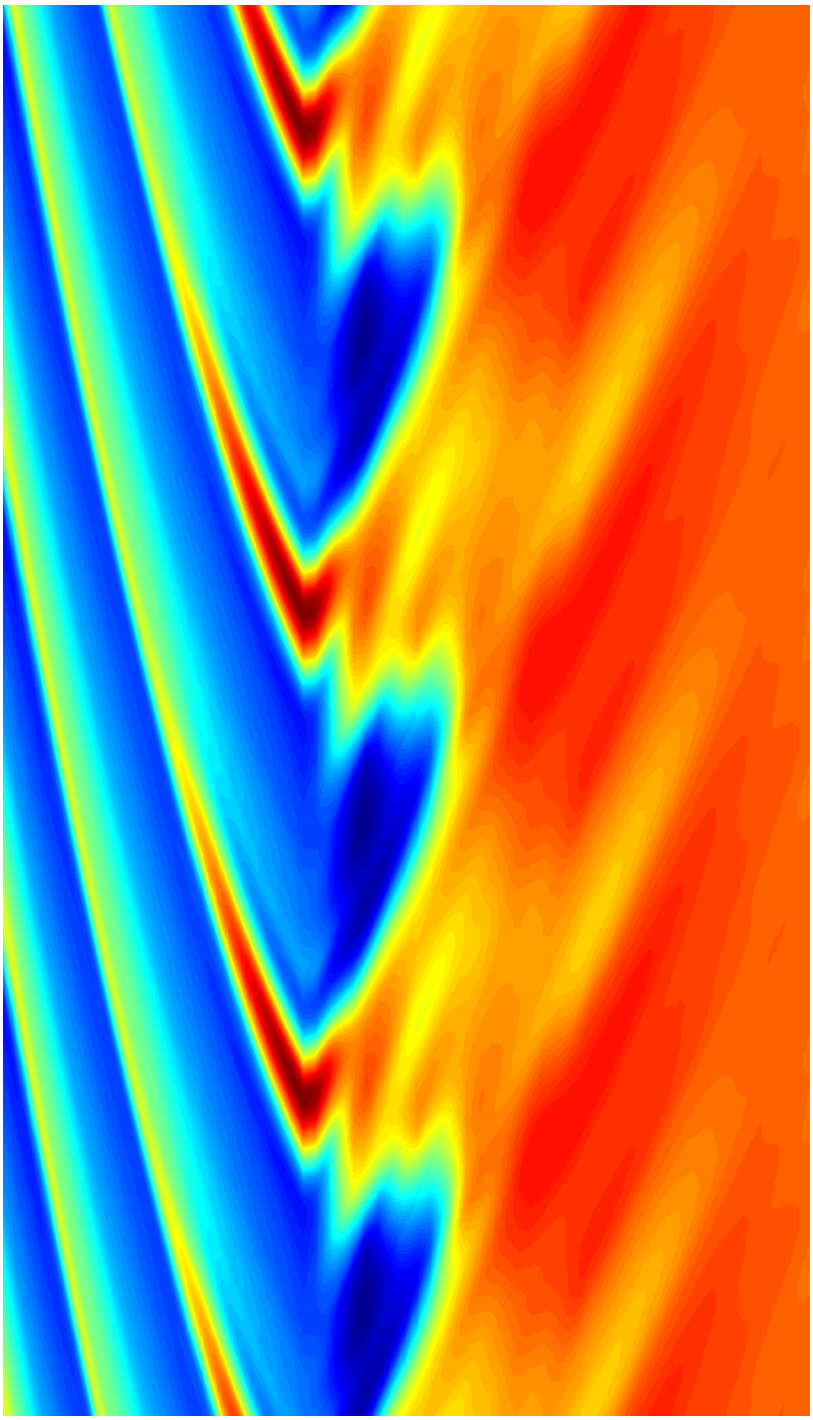


Figure 57. Peak Efficiency: 85%

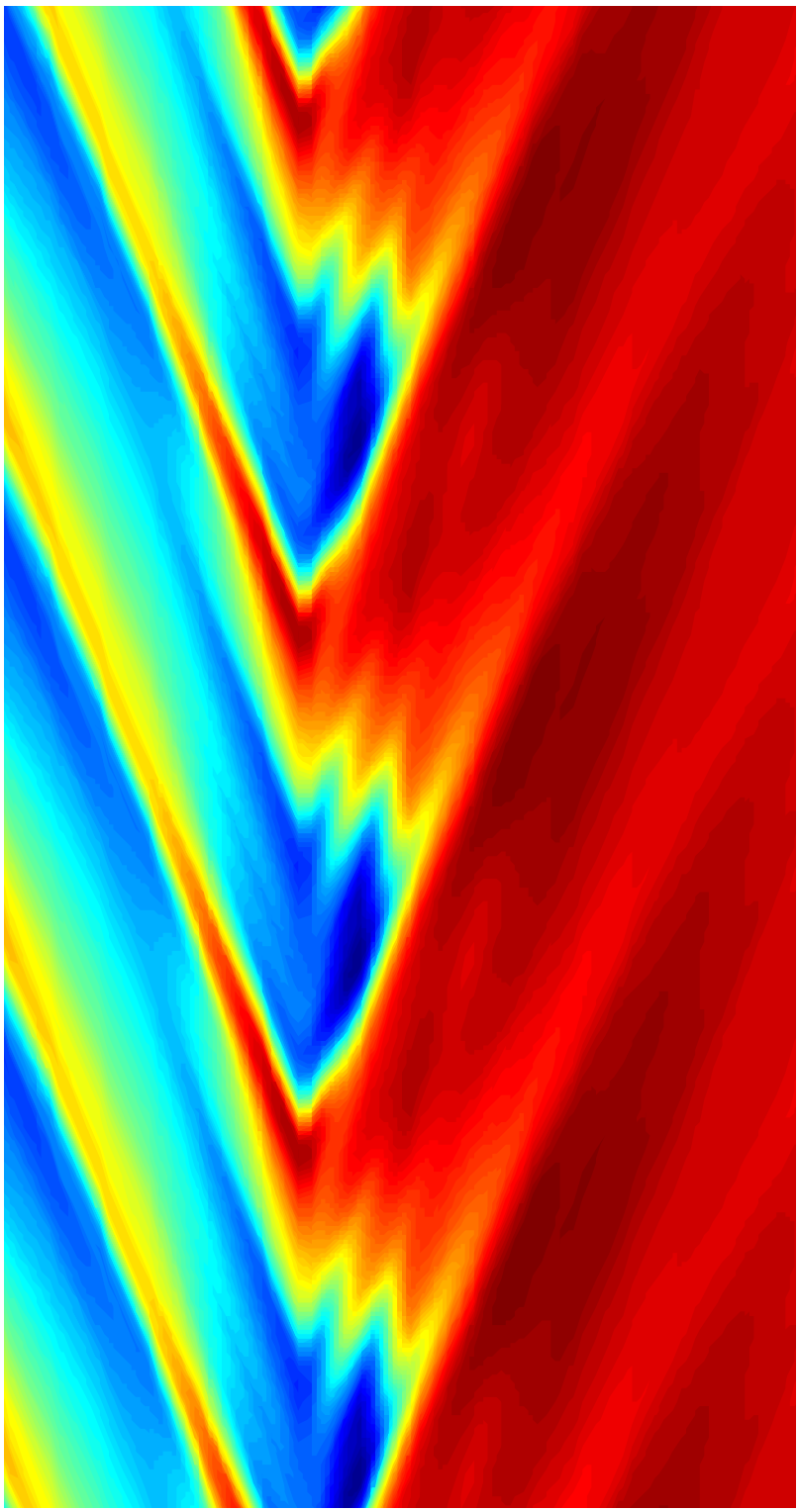


Figure 58. Close to Stall: 85%

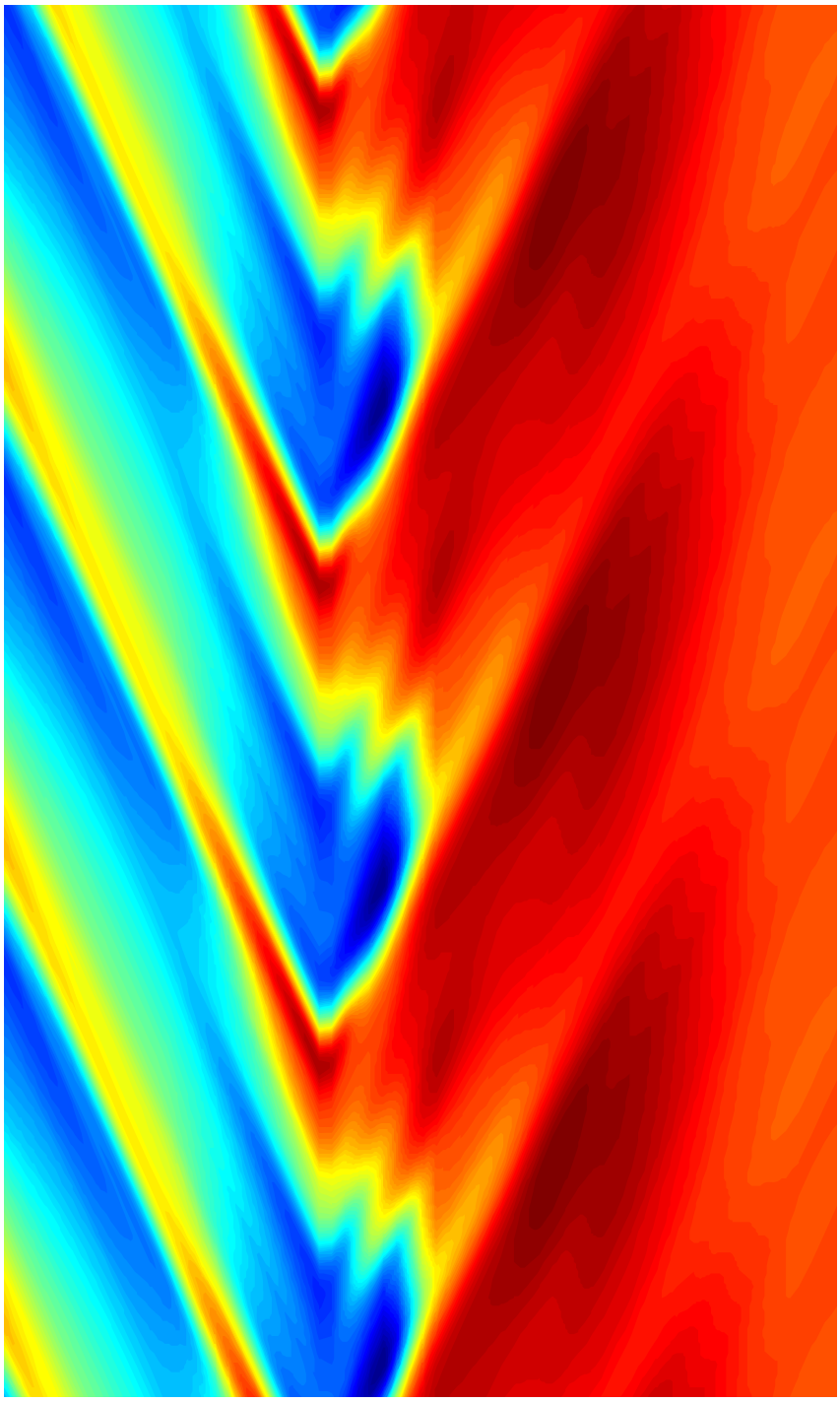


Figure 59. Near-Stall: 85%

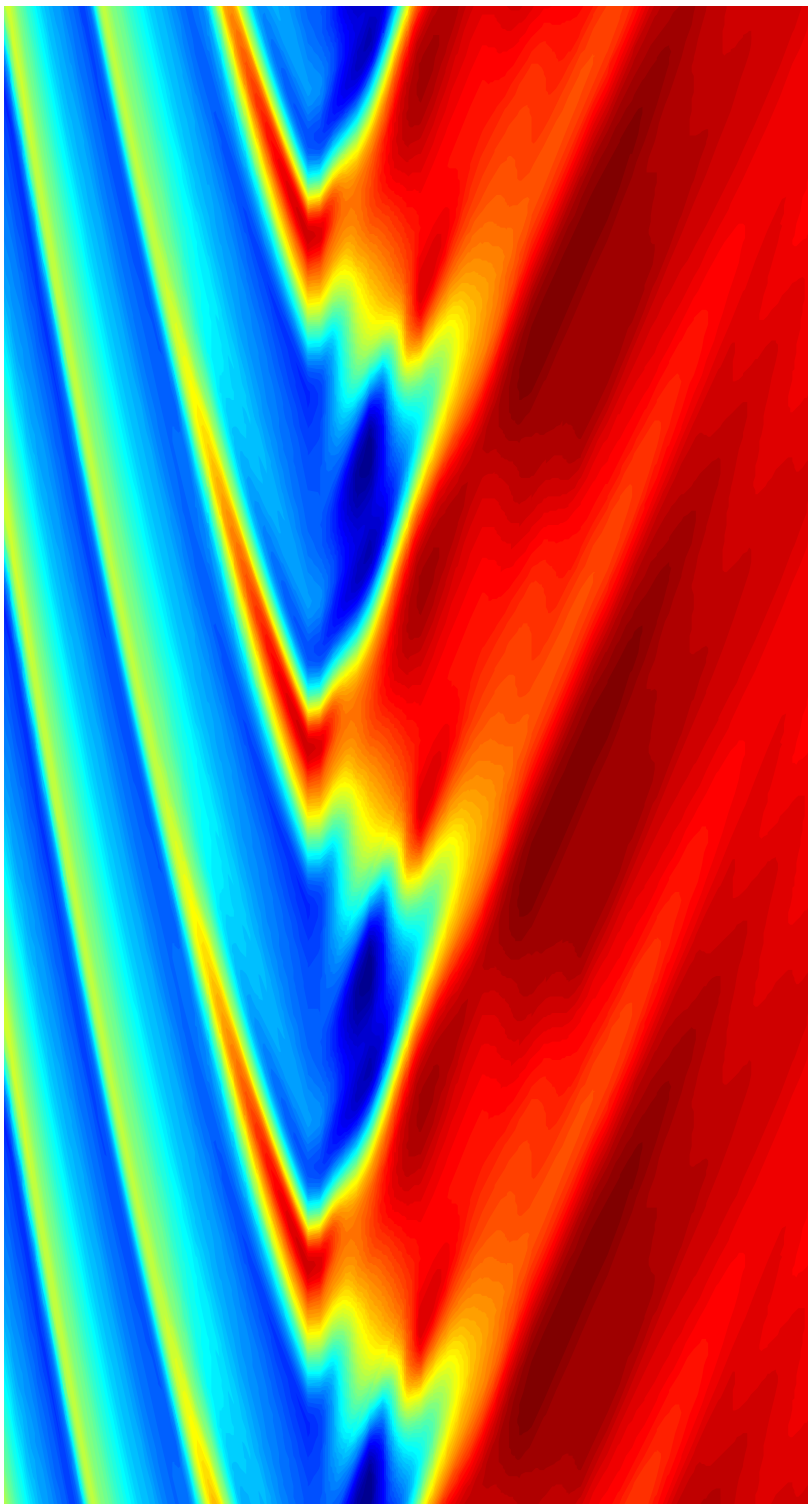


Figure 60. Peak Efficiency: 90%

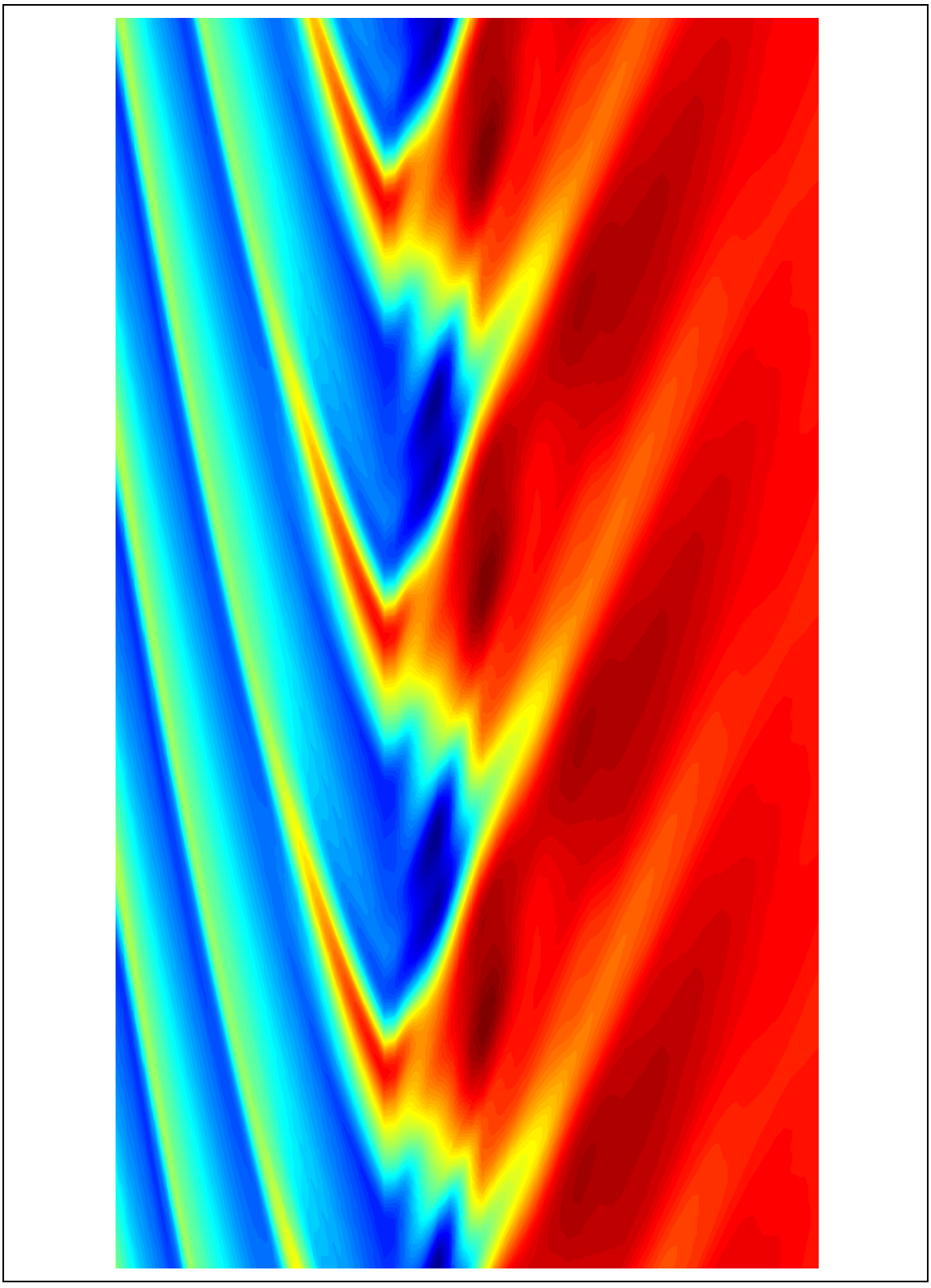


Figure 61. Close to Stall: 90%

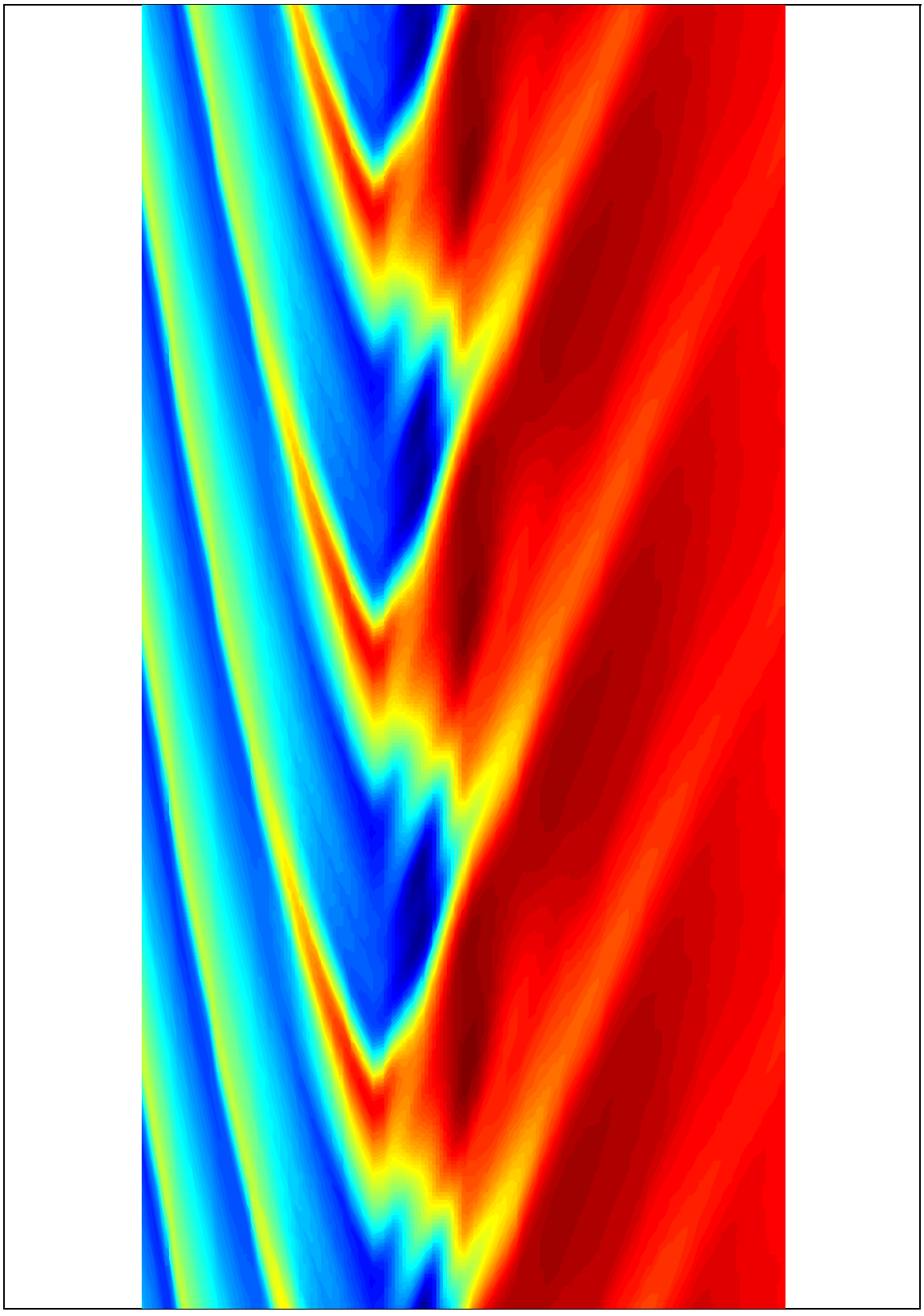


Figure 62. Near-Stall: 90%

APPENDIX C. EXPERIMENTAL HARDWARE SET-UP PROCEDURE

Before every experiment the Transonic Compressor's hardware and software are examined and tested to ensure proper function.

Hardware Test Set-up

1. Conduct a visual inspection of the TCR for foreign objects in or near the rotor casing.

2. Check the probes and ensure their numbering is correct [-2 -1 0 – 1 2 3 4 5 6 7 8] and that they are seeded and connected properly.

A tested method used to check if the pressure transducers are in place and not leaking is two-fold. First, measure the voltages for each pressure sensor. The expected voltage is about 1 volt. If the voltage on the DAC Express is not showing about 1 volt for each sensor, check the Kulite reference line. For leaks in the Kulite reference line a classical method works best to discover the leak: place the air line in a cup of water. If the water bubbles, there is a leak.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. EXPERIMENTAL SOFTWARE TEST SET-UP

The steady state data is acquired using: Agilent VEE Pro, HP75000 Data Acquisition System Main Frame B for the steady-state data, E8404A Data Acquisition System Main Frame C for the high-speed data is needed to calibrate backpressures.

To verify the VEE Pro program working a low efficiency near 0.2, mass flow rate near zero, and pressure ratio of 1 should appear when the compressor is not running. Temperature rates do not affect the probes because of their built in temperature compensation.

During the experiment the warning "FIFO Overflow" appeared on the DAC interface. This meant too many channels were being used. The impact Kulite probe had just been inserted. In order to quickly correct this error (since the compressor was still running) the PC was restarted and the mainframes were rebooted.

After the run, the data is exported from the DAC to a computer where the data processing will take place. The exporting is shown:

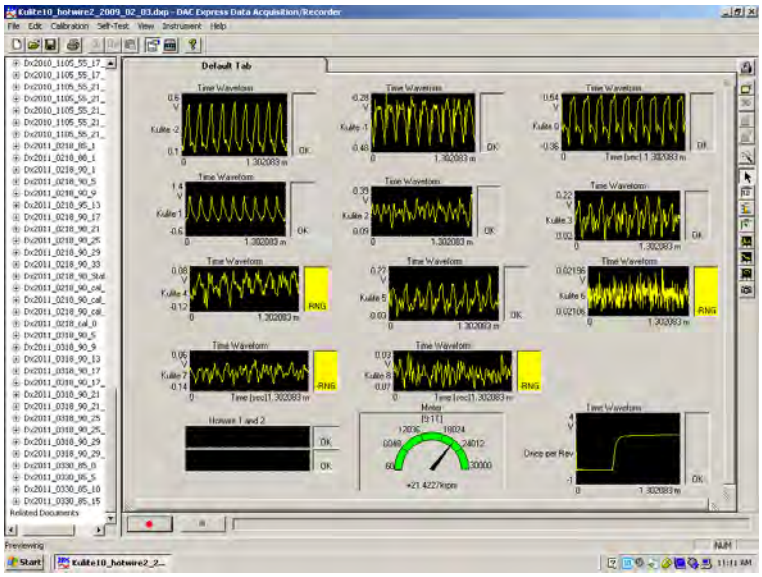


Figure 63. DAC Express Data Acquisition Interface Screen

Select the saved data file on the DAC and click on File- ‘Export Data’. The following window will appear: Ensure that the dynamic channels are selected, as well as all channel data, in *.csv format. Press- ‘Export Data’.

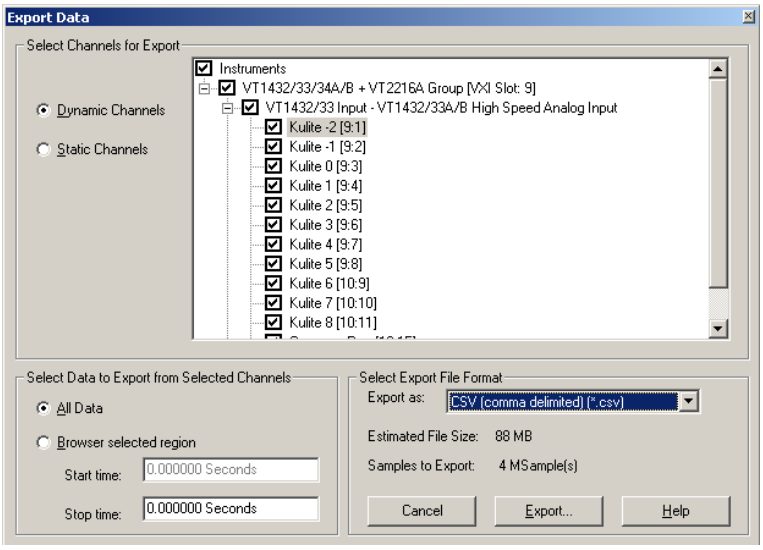


Figure 64. Exporting Data from DAC Express

Save data in desired folders (example: 90%NPSMF/Run 21/Dx2011_0318_90_21) and Press- ‘Save’.

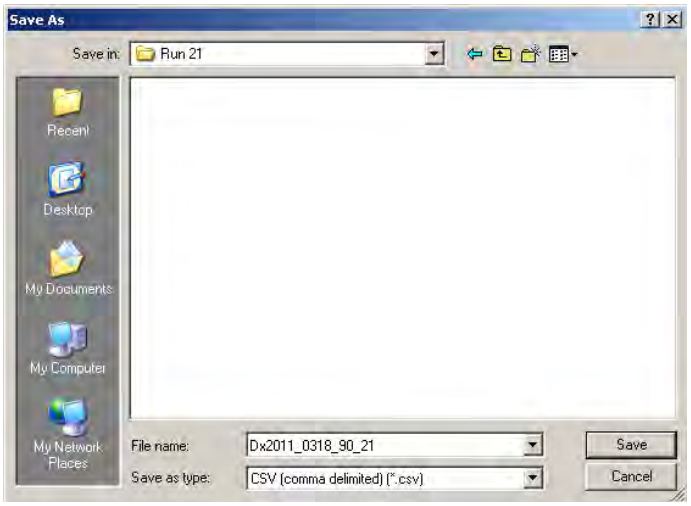


Figure 65. Exporting: Save As

Data will export and save to new desired location.

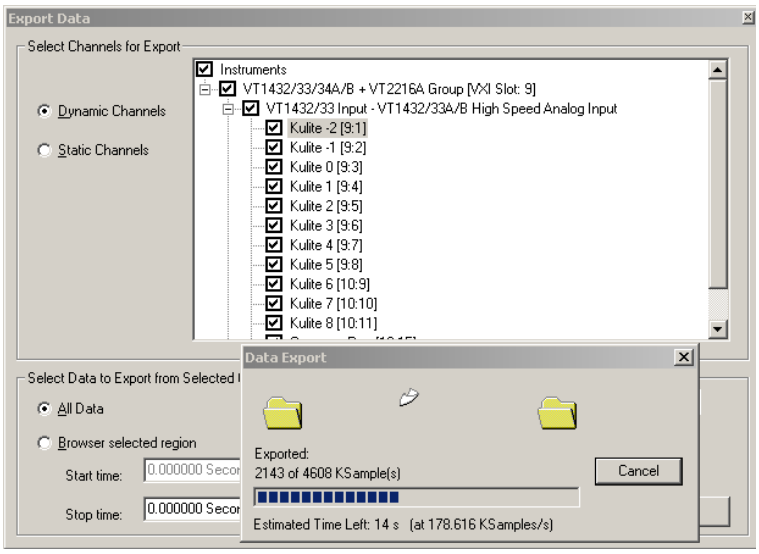


Figure 66. Exporting Data

Load Data by opening MATLAB and setting the path to desired folder (example: C:\Andie Londono\MATLAB_code\MATLAB).

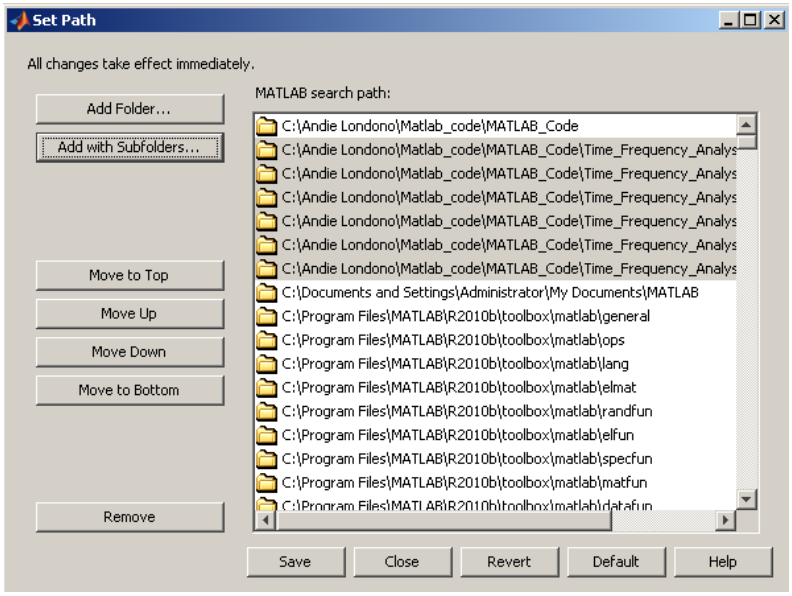


Figure 67. Set Path Part 1

Set another path for the data files under D:\Anthony Gannon\Kulite. Ensure the sub folders are added to both paths. This is done is there is more than one folder the files used are located.

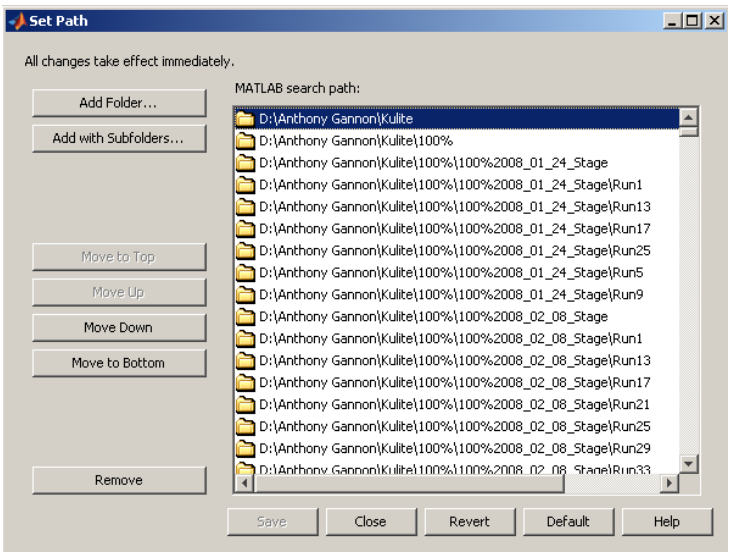


Figure 68. Set Path Part 2

APPENDIX E. SPEED PROFILES: 85%, 80%, AND 70%

85% Speed (25500 RPM)	
Run 1-4	Full Open Throttle (Not at full speed due to power limitation)
Run 5-8	Throttle Setting 5.0
Run 9-12	Throttle Setting 6.75
Run 13-16	Throttle Setting 6.8, near stall

Table 7. Speed Profile: 85%

At 85% (25500 rpm) the impact Kulite had stopped responding because the lead had come off. At full open throttle, the first few probes experiences a voltage increase, this was due to the shock moving forward.

80% Speed – Normal Shock conditions	
Run 1-4	Full Open Throttle (Not at full speed due to power limitation)
Run 5-8	Throttle Setting 5
Run 9-12	Throttle Setting 6.75, Kulite probe #7 dead for remaining runs

Table 8. Speed Profile: 80%

At 80% design speed open throttle Kulite probes 1 and 4 are unstable since this is where the shock was forming. After open throttle (Run 5) Kulite 4 is still extremely volatile. During peak efficiency Kulite 1 Kulites 3, 4, and 7 experienced significant variation. The near-stall run approached stall closer than previous experiments.

70% Speed (21krpm)	
Run 1-4	Full Open Throttle (Not at full speed due to power limitation)
Run 5-8	Throttle Setting 5
Run 9-12	Throttle Setting 6.5 (Whistling Sound from TCR)
Run 13-16	Throttle Setting 6.75
Run 17-20	Throttle Setting 6.8, very near stall

Table 9. Speed Profile: 70%

At 70% speed open throttle Kulite probes 4 and 5 may be saturated. At peak efficiency a whistling noise was heard from the transonic compressor. Variation in voltage and instability are noticed in Kulite probess 4, 5, and 7 when close to stall.

APPENDIX F. ENGINEERING DRAWING OF PROBE LOCATIONS

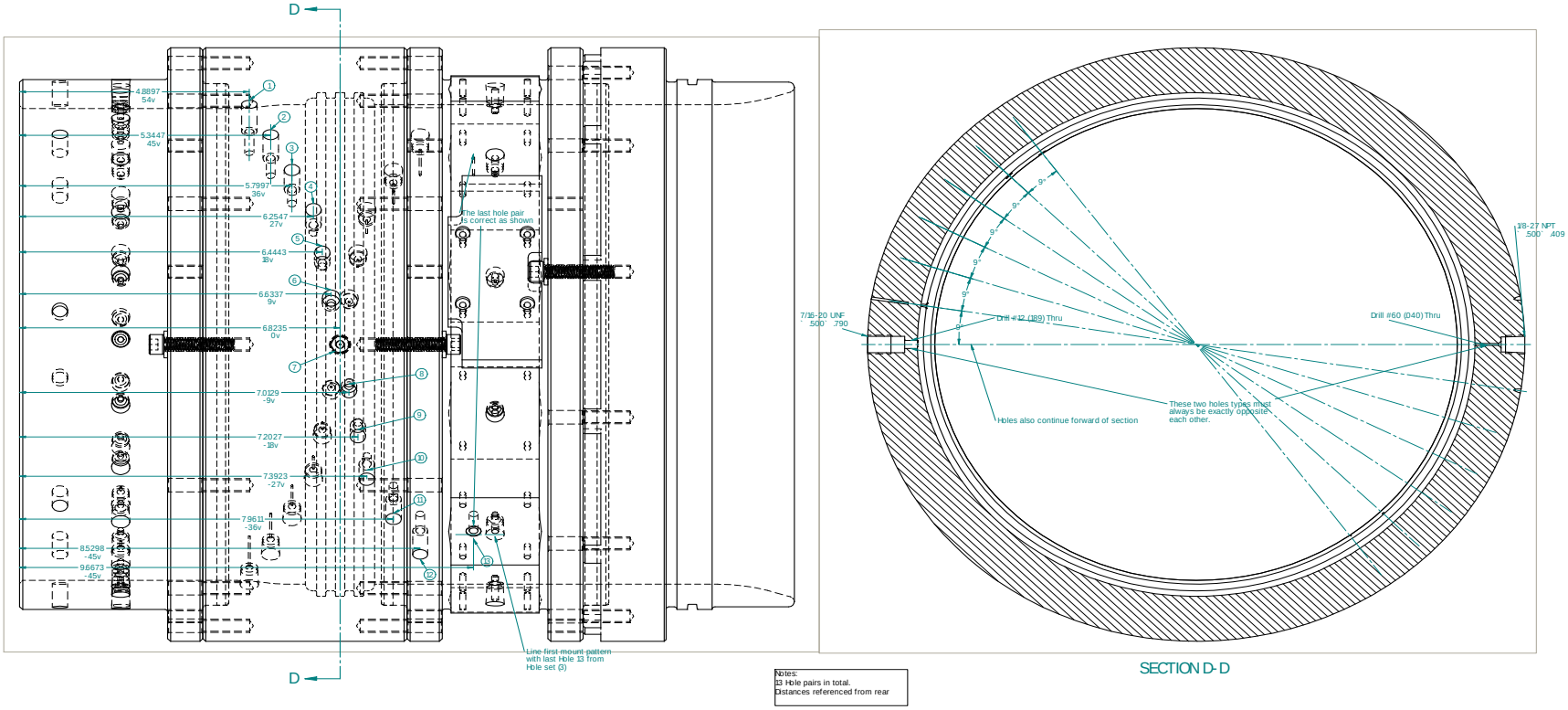


Figure 69. Engineering Drawing of Probe Location

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Head, Information Operations and Space Integration Branch
   PLI/PP&O/HQMC
   Washington, DC

4. Professor and Chairman Knox T. Milsaps
   Department of Mechanical and Aeronautical Engineering
   Naval Postgraduate School
   Monterey, California

5. Professor AntYhony J. Gannon
   Department of Mechanical and Aeronautical Engineering
   Naval Postgraduate School
   Monterey, California

6. Professor Garth V. Hobson
   Department of Mechanical and Aeronautical Engineering
   Naval Postgraduate School
   Monterey, California

7. LT Andrea Londono
   Monterey, California