Analysis of Vortex-Induced Counter Torque and Fin Pressure on a Finned Body of Revolution

Leyen S. Chang

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy’s National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.
Analysis of Vortex-Induced Counter Torque and Fin Pressure on a Finned Body of Revolution

Leyen S Chang
Department of Mechanical Engineering
Stanford University
Stanford, California 94305-3030

Aerosciences and Compressible Fluid Mechanics Division
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-0825

Abstract

Finned bodies of revolution firing lateral jets in flight may experience lower spin rates than predicted. This reduction in spin rate is a result of vortices generated by the interaction between the lateral jets and freestream air flowing past the body. The vortices change the pressure distribution on the fins, inducing a counter torque that opposes the desired spin. Wind tunnel data measuring roll torque and fin pressures were collected for a full-scale model at varying angle of attack, roll angle, airspeed, and jet strength. The current analysis builds upon previously written code that computes torque by integrating pressure over the fin surfaces at 0° angle of attack. The code was modified to investigate the behavior of counter torque at different angles of attack and roll angles as a function of $J$, the ratio of jet dynamic pressure to freestream dynamic pressure. Numerical error analysis was applied to all data to assist with interpretation of results. Results show that agreement between balance and fin pressure counter torque at 0° angle of attack was not as close as previously believed. Counter torque at 4° angle of attack was higher than at 0°, and agreement between balance and fin pressure counter torque was closer. Plots of differential fin pressure coefficient revealed a region of high pressure at the leading edge and an area of low pressure over the center and aft regions of the tapped surface. Large differences in the counter-torque coefficient were found between various freestream dynamic pressures, especially at Mach 0.95 and 1.1. Roll angle had significant effect only for cases at angle of attack, where it caused counter torque to change unpredictably.
Acknowledgments

My deepest appreciation goes to Walt Wolfe and Steve Beresh for working with me on this project and allowing me to learn so much. I would also like to thank Basil Hassan and Bonnie Wheeler for all their work in facilitating and supporting my employment at Sandia. Finally, I would like to thank the members of the Aerosciences and Compressible Fluid Mechanics Division (1515) for providing a friendly and supportive environment and for making my experience at Sandia both educational and productive.
Figures

Figure 1. Vortex-Fin Interaction on FBR (View from Fore to Aft) ........................................ 10
Figure 2. Finned Body of Revolution (View from Fore to Aft) ........................................ 11
Figure 3. Side View of FBR ......................................................................................... 11
Figure 4. Fin Shape and Pressure Taps (red) .................................................................. 11
Figure 5. Meshed Pressure Tap Region Generated with Delaunay Triangulation .......... 13
Figure 6. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio -
Configuration 2 ........................................................................................................ 19
Figure 7. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio -
Configuration 3 ........................................................................................................ 19
Figure 8. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio -
Configuration 4 ........................................................................................................ 20
Figure 9. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run
#0386 .................................................................................................................. 21
Figure 10. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run
#0396 .................................................................................................................. 21
Figure 11. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run
#0416 .................................................................................................................. 22
Figure 12. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run
#0451 .................................................................................................................. 23
Figure 13. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run
#0381 .................................................................................................................. 24
Figure 14. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run
#0391 .................................................................................................................. 24
Figure 15. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run
#0401 .................................................................................................................. 25
Figure 16. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Runs
#0377 and 0375 ........................................................................................................ 26
Figure 17. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Runs
#0387 and 0385 ........................................................................................................ 26
Figure 18. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Runs
#0397 and 0395 ........................................................................................................ 27
Figure 19. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Runs
#0407 and 0405 ........................................................................................................ 28
Figure 20. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Runs
#0472 and 0470 ........................................................................................................ 28
Figure 21. Differential Pressure Coefficient for Run #0376 at J=8 ................................. 29
Figure 22. Differential Pressure Coefficient for Run #0376 at J=16 ............................. 30
Figure 23. Differential Pressure Coefficient for Run #0376 at J=36 ............................. 30
Figure 24. Differential Pressure Coefficient for Run #0386 at J=16 ............................. 31
Figure 25. Differential Pressure Coefficient for Run #0396 at J=16 ............................. 31
Figure 26. Differential Pressure Coefficient for Run #0406 at J=16 ............................. 32
Figure 27. Differential Pressure Coefficient for Run #0471 at J=16 ............................. 32
Figure 28. Pressure Coefficient Variation along Slice A at Mach 0.8, J=8................. 33
Figure 29. Pressure Coefficient Variation along Slice A at Mach 0.95, J=8............... 34
Figure 30. Pressure Coefficient Variation along Slice A at Mach 1.1, J=8............... 34
Figure 31. Counter Torque Coefficients at Mach 0.8, 0° Angle of Attack, 0° Roll........ 35
Figure 32. Counter Torque Coefficients at Mach 0.95, 0° Angle of Attack, 0° Roll...... 36
Figure 33. Counter Torque Coefficients at Mach 1.1, 0° Angle of Attack, 0° Roll........ 36
Figure 34. Counter Torque Coefficients at Mach 0.8, 4° Angle of Attack, 0° Roll........ 37
Figure 35. Counter Torque Coefficients at Mach 0.95, 4° Angle of Attack, 0° Roll....... 38
Figure 36. Counter Torque Coefficients at Mach 1.1, 4° Angle of Attack, 0° Roll......... 38
Figure 37. Counter Torque Coefficients at Mach 0.8, 4° Angle of Attack, 45° Roll..... 39
Figure 38. Counter Torque Coefficients at Mach 0.95, 4° Angle of Attack, 45° Roll...... 39
Figure 39. Counter Torque Coefficients at Mach 1.1, 4° Angle of Attack, 45° Roll...... 40

Tables

Table 1. Uncertainty in Pressure-Derived and Balance-Measured Counter Torques....... 18
Table 2. Uncertainty in Pressure Coefficient........................................................... 29
Nomenclature

\( A_{\text{fin}} \)  fin area  \\
\( A_{\text{tr}} \)  tapped region area  \\
\( J \)  nozzle dynamic pressure ratio  \\
\( C_{CT} \)  counter torque coefficient  \\
\( C_P \)  coefficient of pressure  \\
\( CT \)  counter torque due to vortex-fin interaction  \\
\( F \)  force vector  \\
\( \text{FBR} \)  finned body of revolution  \\
\( P \)  pressure measured by transducer  \\
\( P_{\infty} \)  freestream pressure  \\
\( Q_{\text{nozzle}} \)  dynamic pressure on nozzle centerline at nozzle exit  \\
\( Q_{\infty} \)  freestream dynamic pressure  \\
\( r \)  radius vector  \\
\( T \)  total torque on finned body of revolution  \\
\( T_F \)  torque from fins with jet off  \\
\( T_J \)  torque from jets with tunnel off  \\
\( T_{\text{fin}} \)  torque on fin  \\
\( T_{\text{tr}} \)  torque on tapped region  \\
\( U_{CT} \)  uncertainty in counter torque  \\
\( U_{CP} \)  uncertainty in coefficient of pressure  \\
\( U_P \)  uncertainty in pressure transducer  \\
\( U_{Q_{\infty}} \)  uncertainty in freestream dynamic pressure  \\
\( U_r \)  uncertainty in parameter \( r \)  \\
\( \text{VFI} \)  vortex-fin interaction  \\
\( \alpha \)  angle of attack  \\
\( \theta_i \)  sensitivity coefficient  \\
\( \Delta P \)  difference between odd and even side pressure
1 Introduction

Finned bodies of revolution (FBR) firing lateral jets in flight may experience lower spin rates than expected due to an interaction of the jets with freestream air. This interaction creates vortices at the nozzle exits that are swept downstream and affect the pressure distribution over the fin surfaces. Known as Vortex-Fin Interaction (VFI), this phenomenon creates a counter torque opposing the spin of the body. The mechanism of VFI is illustrated in Figure 1, reproduced from Reference 1. A counter-rotating vortex pair is formed at each nozzle and is believed to be the primary cause of interaction with the downstream fins. The objective of the current work is to characterize the behavior of counter torque at various flight conditions by deriving pressure-induced counter torque and comparing these results to balance torque. A thorough investigation of pressure-induced torque may provide a better understanding of the physics of VFI that cannot be obtained from the balance data. Previous MATLAB code applied integration to pressure data for a 0° angle of attack trial to obtain counter torque as a function of $J$, the ratio of jet-to-freestream dynamic pressure. The parameter $J$ is the dominant variable for correlating counter torque with various run conditions. It is defined as:

$$J \equiv \frac{Q_{nozzle}}{Q_\infty}$$

The variables $Q_{nozzle}$ and $Q_\infty$ denote dynamic pressures on the nozzle centerline intersecting the vehicle surface and the freestream air, respectively.

The original code is expanded to accommodate various angles of attack, roll angles, fin positions, and jet nozzle configurations. Behavior of counter-torque coefficient is also investigated in order to identify trends associated with test conditions. In all cases, comparisons are made between results obtained from fin pressure and from the aerodynamic balance. An error analysis is conducted to complement this comparison and assist in validation of results. In addition, visualization of the pressure distribution across the fin is
accomplished with contour plots of pressure coefficient variations. This approach facilitates a better understanding of the physical effects of VFI on fin pressure distribution, and may assist in the interpretation of the findings of the counter-torque plots. These tasks are implemented using numerical integration and interpolation techniques in MATLAB.

2 Experimentation

Wind tunnel testing of a full-scale FBR model was conducted in the Arnold Engineering Development Center (AEDC) Tunnel 16-T and the NASA Ames 11-ft Unitary Tunnel. Tests were conducted at Mach 0.8, 0.95, and 1.1 and freestream dynamic pressures of 220, 440, and 880 psf. Other variables include fin cant angle, angle of attack, roll angle, carriage lugs, varying nozzle size, repositioned fins, and jet strength. Schematics of the FBR are shown in Figures 2-4, reproduced from Reference 2. An aerodynamic balance is placed within the model in order to measure forces and moments upon the body.
The FBR has four fins spaced 90° apart, and fins A and B were instrumented with pressure taps. The taps were placed in the pattern seen in Figure 1, but slightly staggered on
opposing sides due to the fin’s small thickness. Taps were numbered 1 to 48 with 24 on each side; odd numbers on one side, even numbers on the other. During the run, \( J \) was gradually increased while continuously recording pressure and balance data. These results are directly input into the MATLAB code for analysis after headers and titles are stripped. Tare runs were also conducted with tunnel on, jets off to determine torque resulting solely from vehicle geometry and jets on, tunnel off to determine torque due to jet thrust.

3 Counter Torque Computation

The FBR experiences torque from the jets, the fins, and the VFI counter torque. The counter torque \( (CT) \) is defined as follows:

\[
CT = -(T - T_J - T_F)
\]  

\( T \) is the total torque acting on the body, \( T_J \) is the torque from the jets with no freestream flow, and \( T_F \) is the torque from the fins in the absence of the jets. The torque from the jets without a freestream flow does not affect the fin pressures and the torque from the fins without the jets can be calculated using the tare runs. In order to determine counter torque acting upon the body, the 24 pressure readings on each of the 4 instrumented surfaces must be integrated and the torque from the tare runs subtracted. Using Delaunay Triangulation, a mesh is formed between the 24 taps, generating roughly 35 triangles as seen in Figure 5. A mean pressure is then determined for each triangle using the average of the pressures at its 3 vertices.

By computing triangle areas and taking the cross products of 2 legs of each triangle, an array of area vectors is generated. An array of force vectors \( (F_i) \) is created by multiplying the area vectors by the corresponding mean pressures. Since the forces are assumed to act at the centroid of each triangle, the moment arm is defined as the distance from the coordinate system origin to the centroid. By computing the moment arm corresponding to each triangle \( (r_i) \), the total torque on a single surface \( (T_{surf}) \) is given by:

\[
T_{surf} = \sum_{i=1}^{N} r_i \times F_i
\]
The same basic procedure can be applied to tests under various conditions with the FBR’s four fins. Assuming symmetry in the model at 0° angle of attack, the total torque upon the FBR is computed by multiplying the torque on the two instrumented fins by 2. This counter torque can be plotted and compared with the torque measured by the aerodynamic balance. The same basic procedure can be applied to tests under various conditions with some modifications to the MATLAB code.

\[ T_{\text{fin}} = T_{\text{tr}} \frac{A_{\text{fin}}}{A_{\text{tr}}} \]  

(4)

\( T_{\text{fin}}, T_{\text{tr}}, A_{\text{fin}}, \text{ and } A_{\text{tr}} \) denote the torques and areas of the fin and tapped region respectively. Implicit in the use of Eq. (4) is the assumption that the tapped region adequately captures the nature of the pressure in the untapped region. The validity of the assumption is discussed in more detail in the following sections.

When the torque upon a single fin surface is obtained, the torque from the tare run can be subtracted out to obtain the counter torque. Repeating the process for all 4 instrumented surfaces and summing the torques produces the total torque acting upon two of the FBR’s four fins. Assuming symmetry in the model at 0° angle of attack, the total torque upon the FBR is computed by multiplying the torque on the two instrumented fins by 2. This counter torque can be plotted and compared with the torque measured by the aerodynamic balance. The same basic procedure can be applied to tests under various conditions with some modifications to the MATLAB code.

4 MATLAB Implementation

The original MATLAB code, \textit{VFI\_torque.m},\textsuperscript{4} read data for a run at 0° angle of attack and plotted the resulting counter torque together with torque from the aerodynamic balance as a function of \( J \). Plots of torque from the individual tapped surfaces and fins were also
generated. The code was modified to include the effect of torque from the fins with jets off, which had previously been neglected. The modified code was then used to produce plots for all trials at 0° angle of attack, including those at a 45° roll angle. Major modifications were made to the code to accept angles of attack other than 0°, though the basic integration scheme remained the same. Error analysis was conducted numerically using the original code (see below), while new code was written to create pressure contour and counter-torque coefficient plots. A brief description of each code is given, and the complete codes are included in Appendix A.

4.1 VFI_torque.m

As it existed at the start of the current effort, the code initially asks for user input of the directory and filename of the trial to be plotted. The complete data set is loaded, and all fin pressures and nozzle dynamic pressure ratios (J) extracted. Subroutines were then called for all 4 tapped surfaces: FAodd.m, FAeven.m, FBodd.m, FBeven.m. FAodd.m corresponds to “Fin A, odd side,” and the appropriate data are fed into the subroutine. The subroutine vert_and_area.m is then called to generate a triangular mesh from the x, y, and z coordinates using the MATLAB Delaunay function. The vertex coordinates of each triangle are read into an array of x, y, and z vertex coordinates and sent to the subroutine tri3Darea.m, which returns the area magnitude. vert_and_area.m then computes the tapped region area, an array of triangle centroid coordinates, and an array of area vectors (calculated from the cross product of two legs of each triangle). The subfunction triangle_pressures.m computes the mean pressure for each triangle from the pressures at its 3 vertices and outputs the results in an array. The subfunction moment.m then accepts the triangle mean pressures, area vectors, and centroid coordinates and computes torque over the tapped region. Using the known areas of the tapped region and fin, this torque is scaled to the entire fin surface as shown in Eq. (4). The process is repeated for the entire range of J, resulting in a vector of torques. Each of the 4 main subroutines follows this process, resulting in torques for all 4 fin surfaces. After plotting these torques, they are summed and multiplied by 2, resulting in the total torque on the FBR. This result is then plotted against the balance measured torque as a function of J for comparison. A more detailed explanation of the code can be found in Reference 4.

4.2 Fin Torque Removal

The fin pressures resulting from the tare runs are in the same format as the data files, and are read into VFI_torques_tare.m (for 4° angle of attack cases) and VFI_torques_0tare.m (for 0° angle of attack cases). The output from the code is a set of 4 tare torques (one for each tapped surface) for each combination of conditions: Mach 0.8, 0.95, and 1.1; Q 220, 440, and 880 psf; 0° and 4° angle of attack; and 0° and 45° roll angle. The sets of torques at all Mach numbers and freestream dynamic pressures for a given roll angle and angle of attack were compiled together in a text file. These text files are used in VFI_torques_0_AOA.m and VFI_torques_AOA.m to remove the tare torque from the torque on each tapped surface. The subroutines FAodd.m, FAeven.m, FBodd.m, FBeven.m were modified to identify the tare torque corresponding to the current set of conditions, subtract the tare torque from the fin surface torque, and output the results as before to the main
program. This procedure eliminates the torque from the fins that was neglected in the original code. The results from the modified program show significant differences from those generated with the original code.

4.3 Angle of Attack

The program VFI_torques_AOA.m is a modification of VFI_torque.m that facilitates an investigation of counter torque behavior at an angle of attack of ±4°. Because of the symmetric vehicle geometry and the use of only two instrumented fins, a data set at both + and − 4° angle of attack must be used to completely describe the torque upon the FBR. VFI_torques_AOA accepts two data files, one at positive and one at negative angles of attack. The program now works with 8 distinct torque surfaces (2 for each of the 4 fins), calling subroutines FA_odd_pos.m and FA_odd_neg.m instead of FA_odd.m and so forth for the other 7 surfaces. These subfunctions return 8 torque vectors to the main program, where the torques resulting from positive and negative angles of attack are summed separately. This gives a torque vector as a function of J for each half of the FBR at angle of attack. Since the two runs correspond to slightly different J values, their torques cannot be summed directly. A uniform array of J values is generated based upon the range of the two J sets, and both torque vectors are interpolated onto this array. Summing the interpolated torques produces the total counter torque upon the FBR at angle of attack as a function of J. The two balance-measured torques are also extracted from the dataset and interpolated onto the uniform J vector.

4.4 Error Analysis

Error analysis of the experimental data and MATLAB results was conducted to aid in the comparison of pressure-derived torques to experimentally measured torques. The program VFI_torquesuncertainty.m was used to solve for errors numerically. The uncertainty equation that was used is given in its basic form as follows:

\[ U_{r}^{2} = \sum_{i=1}^{J} \theta_{i}^{2}U_{X_{i}}^{2} \]  

(5)

\( U_{r} \) is the uncertainty in the relevant parameter (in this case the counter torque CT). \( U_{X_{i}} \) denotes the uncertainty in each independently measured variable. \( \theta_{i} \) is the sensitivity coefficient, defined as

\[ \theta_{i} = \frac{\partial r}{\partial X_{i}} \]  

(6)

The only source of uncertainty is the fin pressure — thus \( \theta_{i} \) is given by:

\[ \theta_{i} = \frac{\partial CT}{\partial P_{i}} \]  

(7)
$U_{x_i}$ represents the uncertainty within each pressure transducer. This is assumed to be constant at 0.25% of full scale, or 0.0375 psi. The only other variable is the sensitivity coefficient — a measure of the sensitivity of counter torque with respect to pressure change. A coefficient must be computed for each of the 48 taps on a fin. Symmetry can be used to obtain the uncertainty for the other fins. While these sensitivity coefficients can be determined analytically, the algebra is quite complex and requires keeping track of each triangle’s area, unit normal, and moment arm vector. This procedure is thus very error-prone. To reduce the possibility of introducing errors in the process, it was decided to compute the coefficients numerically using existing MATLAB scripts.

The MATLAB code directly computes the sensitivity coefficients for fin A and uses symmetry to compute the total uncertainty. Initially, the value of tap 1 was set to one and the rest to zero before calling the functions `FA_odd.m` and `FA_even.m`. The torque output from the function is the sensitivity coefficient of the tap. The value of tap 2 was then set to one and all others to zero. This process was repeated for all 48 taps and the resulting torques were recorded in a vector. With the sensitivity coefficients on one fin, the uncertainty of the counter torque for all fins, $U_{CT}$, can be calculated from the following:

$$U_{CT} = \sqrt{a \sum_{i=1}^{48} (\theta * U_p)_i^2}$$

(8)

The uncertainty in the pressure transducers, $U_p$, is constant at 0.0375 psi as stated previously. The coefficient $a$ is a correction factor used to scale the uncertainty of a fin to the entire FBR. For tests at angle of attack, data are obtained from 4 instrumented fins and hence $a$ takes the value of 4. At 0° angle of attack, data from only 2 instrumented fins are used. In this case, $a$ equals 8 — a factor of 4 corresponding to the 4 fins, while a factor of 2 appears since the error is doubled by obtaining torque from only 2 instrumented fins. Thus, uncertainty is less for trials at angle of attack by a factor of $\sqrt{2}$.

The error in the torque measured by the aerodynamic balance was computed from the following equation:

$$U_r^2 = B_r^2 + P_r^2$$

(9)

Both the bias, $B_r$, and precision, $P_r$, of the aerodynamic balance are known quantities provided by the wind tunnel facility. Using these values, the uncertainty in the balance, $U_r$, was determined.

### 4.5 Pressure Coefficient Distribution

Plots of the pressure coefficient variation across the tapped region were created using `finpress_plot.m`. Pressure coefficient $C_P$, is defined as:

$$C_P = (P - P_\infty) / \rho_\infty$$

(10)
$P$ is the pressure from the pressure transducer and the freestream pressure $P_\infty$ is determined from the jet-off tare. The program takes a directory and run number as inputs and generates a shaded pressure coefficient distribution across the tapped region. Five slices are also extracted from the plot along the leading edge and chord-wise across the tapped region and the variation of pressure coefficient is plotted against distance along the slice. Plots were only generated for Fin B since no significant variations were seen in Fin A. As seen in Figure 2, Fin B is almost directly downstream of the nozzle and experiences a much larger effect from the induced vortices than Fin A.

A uniform grid was first generated over the tapped region using the MATLAB “meshgrid” command. Because a differential pressure coefficient was desired (i.e. the difference between pressure coefficients on the odd and even sides), the pressures and coordinates of both the odd and even taps were extracted from the data. The odd and even pressures were then interpolated onto the mesh using the “griddata” command. A separate error analysis was also conducted upon the pressure coefficient data using equations 5 and 7. The resulting equation for uncertainty in the coefficient of pressure, $U_{CP}$, is as follows:

$$U_{CP} = \sqrt{\left(\frac{1}{Q_\infty} U_P\right)^2 + \left(\frac{-\Delta P}{Q_\infty^2} U_{Q_\infty}\right)^2}$$

(11)

The uncertainty in pressure, $U_P$, again corresponds to uncertainty in the pressure transducers (0.0375 psi). The term $U_{Q_\infty}$ refers to uncertainty in the tunnel dynamic pressure and was obtained directly from AEDC’s uncertainty analysis. Since differential pressure coefficient is computed, $\Delta P$ corresponds to the difference in pressure coefficient between the even and odd side of the fin. Uncertainty is computed for all points on the grid, although only the maximum is reported since variation between points is minimal.

### 4.6 Counter Torque Coefficient

The counter torque coefficient is a dimensionless parameter that enables the comparison of the magnitude of counter torque under different conditions. It is defined as:

$$C_{CT} = \frac{CT}{Q_\infty AD}$$

(12)

The counter torque is represented by $CT$, while $A$ is the model maximum cross-sectional area ($\pi D^2 / 4$). The reference length $D$ is the maximum body diameter (13.3 inches). Plots with error bars were generated for all run conditions using $VFI\_CTcoeff.m$ and $VFI\_CT\_AOA.m$. Again using equations 5 and 7, uncertainty in the counter torque coefficient is given by:

$$U_{CCT} = \sqrt{\left(\frac{1}{Q_\infty AD} U_{CT}\right)^2 + \left(\frac{CT}{Q_\infty^2 AD} U_{Q_\infty}\right)^2}$$

(13)
5 Results

Previous results for 0° angle of attack runs were obtained while neglecting the torque from the tare runs. As mentioned previously, the code was corrected and plots were generated for models of configurations 2, 3, and 4 at all run conditions. Error bands were also included on the plots, represented by dashed lines above and below the data. The uncertainties in the counter torques determined from the error analysis are displayed in Table 1.

<table>
<thead>
<tr>
<th>Counter Torque</th>
<th>Uncertainty (± in-lbf)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure-derived at 0° angle of attack</td>
<td>39.33</td>
</tr>
<tr>
<td>Pressure-derived at 4° angle of attack</td>
<td>27.81</td>
</tr>
<tr>
<td>Balance-measured</td>
<td>13.28</td>
</tr>
</tbody>
</table>

Total counter torques for configuration 2, 3, and 4 at 0° angle of attack can be seen in Figures 6-8. Over a similar range of nozzle dynamic pressure ratio, $J$, the counter torques for the three configurations are essentially identical within the experimental uncertainty. This trend is seen at nearly all run conditions. Configurations 3 and 4 have different lateral jet nozzles from configuration 2 -- the baseline model upon which the rest of this investigation will focus. The “RMS difference” reported on the plots is the root mean square of the difference between the balance and fin pressure counter torques. Before $J = 10$ there is close agreement between the balance and fin pressure counter torques. After this point, the agreement between the two torques is not as close, although they are near or within the error limits.
Figure 6. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio - Configuration 2

Figure 7. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio - Configuration 3
Figure 8. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio - Configuration 4

The remaining figures in this section show data for configuration 2. Results were first produced for the 0° angle of attack runs. Figures 9 and 10 show counter torques at Q = 220 psf for Mach 0.95 and 1.1.
Figure 9. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run #0386.

Figure 10. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run #0396
The fin pressure counter torque determined by the modified VFI_torque.m does not match plots created using the original code. After subtracting the tare torques, the fin pressure counter torque is consistently less than the balance measured torque. Agreement between the two plotted torques decreases as Mach number increases, which is reflected by an increase in RMS difference. Increase in Mach number also corresponds to an increase in the magnitude of the counter torque.

As the freestream dynamic pressure is increased from 220 to 880 psf, the magnitude of the counter torque increases for all cases. However, the agreement between balance and fin pressure torques remains close at Mach 0.8, but generally slightly in excess of the measurement uncertainty. The agreement between the two torques becomes poor at Mach 0.95 and 1.1 as seen in Figures 11 and 12.

Figure 11. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run #0416
Figure 12. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run #0451.

Plots of counter torque on the FBR at a roll angle of 45° were also produced using VFI_torque.m. The magnitude and shape of the counter torque plots at a 45° roll angle are similar to those at 0° roll as seen in Figures 13-15. This trend is seen at all Mach numbers and freestream dynamic pressures. It is thus verified that roll angle has little effect upon the behavior of counter torque at 0° angle of attack.
Figure 13. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run #0381.

Figure 14. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Run #0391.
The next investigation involved trials at an angle of attack of 4°. As mentioned previously, data were combined from 2 individual trials to determine counter torque on the FBR at angle of attack. Figures 16-18 display the counter torque for trials at Q = 220 psf and are representative of the data at other freestream dynamic pressures. The agreement between fin pressure and balance counter torque is clearly better at angle of attack though differences still often exceed the uncertainty. The magnitude and behavior of the counter torque at Mach 0.8 is similar for 0° and 4° angle of attack. However, the magnitude of counter torque increases significantly at angle of attack for Mach 0.95 and 1.1.
Figure 16. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Runs #0377 and 0375.

Figure 17. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Runs #0387 and 0385.
At a given Mach number for cases at angle of attack, the counter torque curve maintains the same general shape but increases in magnitude as the freestream dynamic pressure increases. The agreement between the fin pressure and balance counter torque decreases slightly as Mach number and freestream dynamic pressure are increased, but is nowhere near the marked decrease in agreement seen in zero angle of attack cases. These trends can be seen by comparing Figures 18-20. For a given Mach number and free stream dynamic pressure, the agreement between the balance and fin pressure counter torques is always better at angle of attack.
Figure 19. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Runs #0407 and 0405.

Figure 20. Fin and Balance Counter Torque vs. Nozzle Dynamic Pressure Ratio for Runs #0472 and 0470.
Fin differential pressure coefficient distributions for runs at zero angle of attack are seen in Figures 21-23. Since most of the counter torque acts upon Fin B, only the pressure distribution across this fin is plotted. For a given freestream dynamic pressure and Mach number, plots were generated at $J = 8$, 16, and 36. The plots show the differential pressure coefficient variation (odd side $C_P$ minus even side $C_P$) over the tapped surface. All the plots are characterized by a high pressure region at the leading edge and a large low pressure region over the middle of the tapped surface. The change in pressure coefficient with Mach number can be seen in Figures 22, 24, and 25. An expansion of the large pressure differential is seen at the leading edge as well as an increase in the overall magnitude of the pressure coefficient as Mach number increases. No discernible trend can be seen as freestream dynamic pressure increases in Figures 22, 26, and 27. Similar behavior is seen at all three values of $J$. Uncertainties in the pressure coefficient are dependent only on freestream dynamic pressure and are displayed in Table 2.

### Table 2. Uncertainty in Pressure Coefficient

<table>
<thead>
<tr>
<th>$Q$ (psf)</th>
<th>$C_P$ Uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>220</td>
<td>0.0245</td>
</tr>
<tr>
<td>440</td>
<td>0.0123</td>
</tr>
<tr>
<td>880</td>
<td>0.0061</td>
</tr>
</tbody>
</table>

![Figure 21. Differential Pressure Coefficient for Run #0376 at J=8.](image)
Figure 22. Differential Pressure Coefficient for Run #0376 at J=16.

Figure 23. Differential Pressure Coefficient for Run #0376 at J=36.
Figure 24. Differential Pressure Coefficient for Run #0386 at J=16.

Figure 25. Differential Pressure Coefficient for Run #0396 at J=16.
Figure 26. Differential Pressure Coefficient for Run #0406 at J=16.

Figure 27. Differential Pressure Coefficient for Run #0471 at J=16.
In order to quantify the variation of pressure coefficient, slices were taken longitudinally across the tapped region (Slice A) and along the leading edge (Slices B-E). According to Figures 28-30, the coefficient of pressure behaves similarly along the longitudinal slice at Mach 0.8 and 0.95 although the magnitude is slightly greater in the latter. At Mach 1.1, a small positive pressure differential arises toward the center of the tapped region. The greatest pressure coefficient magnitude occurs at Mach 1.1, particularly in the peak at the leading edge. There is little change in the pressure coefficient distribution as freestream dynamic pressure varies from 220 to 880 psf. The most significant change in the pressure coefficient distribution occurs between Mach 0.95 and Mach 1.1.

The high pressure area at the leading edge clearly extends beyond the boundaries of the tapped region. This calls into question the validity of the assumption that the ratio of the pressure in the tapped region to the pressure over the fin surface is proportional to the ratio of their respective areas. The existence of complexities beyond the tapped area decreases the accuracy of this assumption. The poor agreement of the balance and pressure-derived torques at $0^\circ$ angle of attack as previously mentioned may be a result of this problem.

![Figure 28. Pressure Coefficient Variation along Slice A at Mach 0.8, J=8.](image-url)
Figure 29. Pressure Coefficient Variation along Slice A at Mach 0.95, J=8.

Figure 30. Pressure Coefficient Variation along Slice A at Mach 1.1, J=8.
Plots of counter torque coefficient as a function of $J$ were generated for all run conditions. The counter torque curves at $Q = 220$, $440$, and $880$ psf were plotted together for comparison for each Mach number in Figures 31-33. In all cases at zero angle of attack, discrepancies between the balance and fin pressure counter torque coefficients became significant at approximately $J > 10$. However at Mach 0.95 and 1.1 larger differences between the coefficients appear, particularly at $Q = 440$ psf. This behavior reflects similar results obtained in Reference 1. According to Figures 32 and 33, the fin pressure torque follows the general pattern of the balance torque, but is offset downward by a nearly constant factor.

Figure 31. Counter Torque Coefficients at Mach 0.8, 0° Angle of Attack, 0° Roll.
Figure 32. Counter Torque Coefficients at Mach 0.95, 0° Angle of Attack, 0° Roll.

Figure 33. Counter Torque Coefficients at Mach 1.1, 0° Angle of Attack, 0° Roll.
More consistent results were obtained for cases at an angle of attack of 4° and 0° roll angle. The disagreement seen in 0° angle of attack cases is no longer as prominent in Figures 34-36. The behavior of the counter torque coefficient at Q = 440 psf is again the most anomalous, though differences are not as large as those at 0° angle of attack.

Figure 34. Counter Torque Coefficients at Mach 0.8, 4° Angle of Attack, 0° Roll.
Figure 35. Counter Torque Coefficients at Mach 0.95, 4° Angle of Attack, 0° Roll.

Figure 36. Counter Torque Coefficients at Mach 1.1, 4° Angle of Attack, 0° Roll.

At 0° angle of attack, the counter torque coefficients at 0° and 45° roll angle are nearly identical. Counter torque coefficient behaves similarly at 45° roll and 4° angle of attack as seen in Figures 37-39. The close agreement between the counter-torque coefficients in Figures 34-36 is lost when the FBR is at a roll angle. Again, the most conspicuous discrepancies in the data occur at Q = 440 psf.
Figure 37. Counter Torque Coefficients at Mach 0.8, 4° Angle of Attack, 45° Roll.

Figure 38. Counter Torque Coefficients at Mach 0.95, 4° Angle of Attack, 45° Roll.
Figure 39. Counter Torque Coefficients at Mach 1.1, 4° Angle of Attack, 45° Roll.

6 Summary and Conclusions

The main objective of the project was to define the behavior of counter torque at various flowfield conditions as derived from fin pressure data. Modification of the original code from Reference 2 revealed that agreement between the fin pressure and balance counter torques was poorer than previously believed because of the jet-off tares. The investigation of pressure coefficient distributions over the tapped region at zero angle of attack implied that the complexity of the pressure distribution beyond the instrumented surface cannot be neglected. MATLAB code was successfully produced to generate counter torque plots for runs at angle of attack. In this case, there was good agreement between the fin pressure and balance results. Roll angle was found to be a significant factor only at angle of attack. In addition, different nozzle configurations had little effect on counter torque. An investigation of the counter torque coefficient verified the aberrant behavior of counter torque at Mach 0.95 and 1.1 as previously observed.

The MATLAB code successfully produced counter torque based on fin pressure. However, there is a large inconsistency between the accuracy of results at 0° and 4° angle of attack. Closer agreement between the balance and fin pressure counter torques was consistently obtained at 4° angle of attack. An investigation of the pressure coefficient distribution across the fins at 4° angle of attack may help to identify the problem. A better
method of obtaining the fin counter torque from the torque acting upon the tapped region may need to be developed at zero angle of attack. Also, correlation of the fin pressure distributions to the Particle Image Velocimetry results in Reference 4 may provide useful information.

References


Appendix A: MATLAB M-Files

VFI_torques_0_AOA.m

% plots counter torque at 0 angle of attack
clear all
clc

% global variable declaration
global dir run J FA_p FB_p tap_coords fin_area
global M data tare_pressures

% conversions
psf2psi = 0.0069444; % convert from psf to psi

% user input
dir = input('Enter directory name (ex. C2_Q440_M8): ', 's'); % the directory with DAT files
run = input('Enter the run number (ex. 0426): ', 's'); % the first DAT file

% file input
data = load(['dir '/'RUN0' run '.dat']); % loads entire DAT file
tap_coords = load(['fin_taps.dat']); % loads entire taps file that contains coordinates for the taps

% determining minimum and maximum J values allowed for data
a = find(find(data(:,427)>4.6, 1);
b = find(find(data(:,427)>2.3, 1);
c = find(find(data(:,427)>1.1, 1);;

i = find(find(data(:,427)>73.7, 1);;
j = find(find(data(:,427)>36.8, 1);;
k = find(find(data(:,427)>18.4, 1);;

% finds point at which J begins to decrease and discards subsequent data
if isempty(i)==1 i(1) = min(find(diff(data(100:size(data,1),427))<0)); end
if isempty(j)==1 j(1) = min(find(diff(data(100:size(data,1),427))<0)); end
if isempty(k)==1 k(1) = min(find(diff(data(100:size(data,1),427))<0)); end

if 215<=(round(data(1,7)/10))*10<=225 % sets minimum J value for Q220 data
minsize = a(1); elseif 435<=(round (data(1,7)/10))*10<=445%sets minimum J value for Q440 data
    minsize = b(1); else minsize = c(1);%sets minimum J value for Q880 data end

if 215<=(round (data(1,7)/10))*10<=225%sets maximum J value for Q220 data
    maxsize = i(1); elseif 435<=(round (data(1,7)/10))*10<=445%sets maximum J value for Q440 data
    maxsize = j(1); else maxsize = k(1);%sets maximum J value for Q880 data end

%calculating area of whole fin
fin_dim = load('fin_shape.dat');%loads coordinates of fin edge points
fin_x = fin_dim(:,1);%maintains edge x-coordinates
fin_y = fin_dim(:,2);%maintains edge y-coordinates
fin_area = polyarea(fin_x,fin_y);%calculates area of whole fin

%data filter
J = data(:,427);%maintains Nozzle Dynamic Pressure Ratio from DAT file, in column 427
FA_p = data(:,[87:134]) * psf2psi;%maintains 48 columns of pressure data for Fin A
FB_p = data(:,[135:182]) * psf2psi;%maintains 48 columns of pressure data for Fin B
T_roll = abs(data(:,29));%maintains Roll Torques from Aerodynamic Balance

%run identification
AOA = int2str (round (data(1,13)));%reads angle of attack from DAT file
M= num2str((round (data(1,4)*100))/100);%reads Mach number from DAT file
Q= dir(5:7);%reads Q from DAT file
rollangle= num2str(round(data(1,14)));%reads roll angle from DAT file

%loading tare pressures
Tare_pressures = load ([dir(1:2) '_0AOA_' rollangle 'taretare.txt']);

%function calls
T_FAodd = FAodd;%calls function FAodd stores fin pressure, roll torques on Fin A, Odd
T_FBodd = FBodd;%calls function FBodd stores fin pressure, roll torques on Fin B, Odd
T_FAeven = FAeven;%calls function FAeven stores fin pressure, roll torques on Fin A, Even
T_FBeven = FBeven;%calls function FBeven stores fin pressure, roll torques on Fin B, Even

%calculations
T_pressure = 2*(T_FAodd + T_FAeven + T_FBodd + T_FBeven);%determines total torque from fin pressures, factor of 2 is for the other 2 symmetric fins
T_FA = T_FAodd + T_FAeven;%determines total torque on Fin A
T_FB = T_FBodd + T_FBeven;%determines total torque on Fin B

%error analysis
T_diff= T_roll-T_pressure;%determines difference between balance and fin pressure torques
RMSdiff= num2str((mean(T_diff.^2))^0.5);%determines RMS difference between balance and fin pressure torques

%plots both balance-measured torque and fin pressure torque on same axis
%and saves it in the same directory with DAT file
hold on
plot(J,T_pressure, 'r', 'LineWidth', 2);
plot(J,T_roll, 'b', 'LineWidth', 2);
plot(J,(T_pressure+39.3264), 'r--', 'LineWidth', 1);%plots error in pressure torques
plot(J,(T_pressure-39.3264 ), 'r--', 'LineWidth', 1);%plots error in pressure torques
plot(J,(T_roll+13.2827), 'b-.', 'LineWidth', 1);%plots error in balance torques
plot(J,(T_roll-13.2827), 'b-.', 'LineWidth', 1);%plots error in balance torques

grid on
legend('T_f_i_n', 'T_b_a_l_a_n_c_e','Location', 'Best');
title(['Run #' run ': Total Counter Torque']);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque [in-lb_f]');
saveas(gcf,[dir '/RUN0' run 'total'],'jpg');
hold off
close

%plots total roll torque on Fin A versus J and saves it in the directory with the DAT files
plot(J, T_FA, 'LineWidth', 1);
grid on
title(['Run #' run ': Total Counter Torque on Fin A']);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque [in-lb_f]');
saveas(gcf,[dir '/RUN0' run 'finAtotal'],'jpg');
close

%plots total roll torque on Fin B versus J and saves it in the
directory with
%the DAT files
plot(J, T_FB, 'LineWidth', 1);
grid on
title(['Run #' run ': Total Counter Torque on Fin B']);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque [in-lb_f]');
saveas(gcf,[dir '/RUN0' run 'finBtotal'],'jpg');
close

FAodd.m

function T_FAodd = FAodd

%global variable declaration
global dir run J FA_p tap_coords fin_area tare_pressures

%data filter
p = FA_p(:,[1:2:47]);%maintains pressure data for Fin B, odd
pressure taps
x = tap_coords(:,1);%maintains x coordinates for Fin A, odd pressure
taps
y = tap_coords(:,2);%maintains y coordinates for Fin A, odd pressure
taps
z = tap_coords(:,3);%maintains z coordinates for Fin A, odd pressure
taps

%reading appropriate tare pressure from file
if str2num(dir(10:length(dir)))==8 && str2num(dir(5:7))==220
    tare_pressure=tare_pressures(1);
elseif str2num(dir(10:length(dir)))==95 && str2num(dir(5:7))==220
    tare_pressure=tare_pressures(5);
elseif str2num(dir(10:length(dir)))==11 && str2num(dir(5:7))==220
    tare_pressure=tare_pressures(9);
elseif str2num(dir(10:length(dir)))==8 && str2num(dir(5:7))==440
    tare_pressure=tare_pressures(13);
elseif str2num(dir(10:length(dir)))==95 && str2num(dir(5:7))==440
    tare_pressure=tare_pressures(17);
elseif str2num(dir(10:length(dir)))==11 && str2num(dir(5:7))==440
    tare_pressure=tare_pressures(21);
elseif str2num(dir(10:length(dir)))==8 && str2num(dir(5:7))==880
    tare_pressure=tare_pressures(25);
elseif str2num(dir(10:length(dir)))==95 && str2num(dir(5:7))==880
    tare_pressure=tare_pressures(29);
elseif str2num(dir(10:length(dir)))==11 && str2num(dir(5:7))==880
tare_pressure=tare_pressures(34);
else disp('Error identifying tare pressure')
end

%function calls
[vertices, area_vectors, r, a_surf] = vert_and_area(x, y, z);%calls function to determine triangle vertices,
...area vectors, triangle centroids, and area on Fin A, Odd within tapped region
p_triangle = triangle_pressures(p, vertices);%calls function to determine mean pressures for each triangle
T_tot = moment(p_triangle, area_vectors, r);%calls function to determine torque for the entire surface, Fin A:Odd

%calculations
T_tot_x = -1.*T_tot(:,1);

%plots roll torque for Fin A, Odd versus J and saves it in directory with
%DAT file
plot(J, T_tot_x, 'LineWidth', 1)
grid on
ylabel('Counter Torque [in-lb_f]')
xlabel('Nozzle Dynamic Pressure Ratio, J');
title(['Run #' run ': Fin A, Odd Taps']);
saveas(gcf,[dir '/RUN0' run 'finAodd'],'jpg');
close

%scales torque to approximate the torque over the entire Fin A, Odd
T_FAodd = T_tot_x*(fin_area/a_surf)-tare_pressure;

VFI_torques_AOA.m

clear all
clc
format long

%global variable declaration
global dir run_pos run_neg J_pos J_neg FA_p_pos FA_p_neg FB_p_pos
FB_p_neg tap_coords fin_area
global AOA_pos AOA_neg M data_pos data_neg tare_pressures

%conversions
psf2psi = 0.0069444;%convert from psf to psi

%user input
dir = input('Enter directory name (ex. C2_Q440_M8): ', 's');%the directory with DAT files
run_pos = input('Enter the run number for test with positive angle of attack (ex. 0426): ','s');%the first DAT file
run_neg = input('Enter the run number for test with negative angle of attack (ex. 0426): ','s');%the second DAT file

data_pos = load([dir '/RUN0' run_pos '.dat']);%loads positive angle of attack DAT file
data_neg = load([dir '/RUN0' run_neg '.dat']);%loads negative angle of attack DAT file
tap_coords = load(['fin_taps.dat']);%loads entire taps file that contains coordinates for the taps

%constructing range of J values allowed for data
a=find (4.6<data_pos(:,427)<73.7);%J limits on Q220 data, positive AOA
b=find (2.3<data_pos(:,427)<36.8);%J limits on Q440 data, positive AOA
c=find (1.1<data_pos(:,427)<18.4);%J limits on Q880 data, positive AOA
d=find (4.6<data_neg(:,427)<73.7);%J limits on Q220 data, negative AOA
e=find (2.3<data_neg(:,427)<36.8);%J limits on Q440 data, negative AOA
f=find (1.1<data_neg(:,427)<18.4);%J limits on Q880 data, negative AOA

%sets range of J values for Q220 data, positive AOA
if str2num(dir(5:7))==220 && round (data_pos(1,13))==4
    if isempty(find(diff(data_pos(:,427))<0))==0%filtering out bad data (decreasing J values)
        min(find(diff(data_pos(1:min(size(data_neg,1),size(data_pos,1)),427))<0))<20%filters bad data at beginning
        s=find(diff(data_pos(1:min(size(data_neg,1),size(data_pos,1)),427))<0);
        s=1+max(s(s<20));
        a_start=s;
    else a_start=1;
end

if isempty(find(diff(data_pos(100:min(size(data_neg,1),size(data_pos,1)),427))<0))==0%filters bad data at end
    a_end=min(find(diff(data_pos(100:min(size(data_neg,1),size(data_pos,1)),427))<0));
    else a_end=size(data_pos,1);
end

datasize_pos=a(a_start:a_end);
else
datasize_pos = a;
end

% sets range of J values for Q440 data, positive AOA
elseif str2num(dir(5:7))==440 && round(data_pos(1,13))==4
    if isempty(find(diff(data_pos(:,427))<0))==0% filtering out bad
data (decreasing J values)
        if
        min(find(diff(data_pos(1:min(size(data_neg,1),size(data_pos,1)),427)
        )<0))<20% filters bad data at beginning

        s=find(diff(data_pos(1:min(size(data_neg,1),size(data_pos,1)),427))<
        0);
            s=1+max(s(s<20));
            b_start=s;
            else b_start=1;
        end
        if
        isempty(find(diff(data_pos(100:min(size(data_neg,1),size(data_pos,1)
        ) ,427))<0))==0% filters bad data at end
        b_end=
        min(find(diff(data_pos(100:min(size(data_neg,1),size(data_pos,1)),427)
        )<0));
            else b_end=size(data_pos,1);
            end

        datasize_pos=b(b_start:b_end);
        else
        datasize_pos = b;
    end

% sets range of J values for Q880 data, positive AOA
elseif str2num(dir(5:7))==880 && round(data_pos(1,13))==4
    if isempty(find(diff(data_pos(:,427))<0))==0% filtering out bad
data (decreasing J values)
        if
        min(find(diff(data_pos(1:min(size(data_neg,1),size(data_pos,1)),427)
        )<0))<20% filters bad data at beginning

        s=find(diff(data_pos(1:min(size(data_neg,1),size(data_pos,1)),427))<
        0);
            s=1+max(s(s<20));
            c_start=s;
            else c_start=1;
        end
        if
        isempty(find(diff(data_pos(100:min(size(data_neg,1),size(data_pos,1)
        ) ,427))<0))==0% filters bad data at end


c_end = min(find(diff(data_pos(100:min(size(data_neg,1),size(data_pos,1)),42 7))<0));
    else c_end = size(data_pos,1);
    end

datasize_pos = c(c_start:c_end);
    else
    datasize_pos = c;
    end
else disp ('Error in data limits')
end

% sets range of J values for Q220 data, negative AOA
if str2num(dir(5:7))==220 && round (data_neg(1,13))==-4
    if isempty(find(diff(data_neg(:,427))<0))==0% filtering out bad data (decreasing J values)
        if min(find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427)<0))<20% filters bad data at beginning
            t = find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427))< 0);
            t = 1+max(t(t<20));
            d_start = t;
            else d_start = 1;
            end
        if isempty(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)),427))<0))==0% filters bad data at end
            d_end = min(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)),427))<0));
            else d_end = size(data_neg,1);
            end
        datasize_neg = d(d_start:d_end);
        else datasize_neg = d;
        end
    if str2num(dir(5:7))==440 && round (data_neg(1,13))==-4% sets range of J values for Q440 data, negative AOA
        if isempty(find(diff(data_neg(:,427))<0))==0% filtering out bad data (decreasing J values)
            if min(find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427)<0))<20% filters bad data at beginning
            t = find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427))< 0);
            t = 1+max(t(t<20));
            d_start = t;
            else d_start = 1;
            end
        if isempty(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)),427))<0))==0% filters bad data at end
            d_end = min(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)),427))<0));
            else d_end = size(data_neg,1);
            end
        datasize_neg = d(d_start:d_end);
        else datasize_neg = d;
        end
    endif str2num(dir(5:7))==440 && round (data_neg(1,13))==-4
endif str2num(dir(5:7))==220 && round (data_neg(1,13))==-4
t=find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427))<0);
    t=1+max(t(t<20));
    e_start=t;
    else e_start=1;
    end

    if isempty(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)),427))<0))==0%filters bad data at end
        e_end=min(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)),427))<0));
        else e_end=size(data_neg,1);
    end

    datasize_neg=e(e_start:e_end);
    else
        datasize_neg = e;
    end

%sets range of J values for Q880 data, negative AOA
elseif str2num(dir(5:7))==880 && round(data_neg(1,13))==-4
    if isempty(find(diff(data_neg(:,427))<0))==0%filtering out bad data (decreasing J values)
        if min(find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427))<0))<20%filters bad data at beginning
            t=find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427))<0);
                t=1+max(t(t<20));
                f_start=t;
                else f_start=1;
            end

            if isempty(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)),427))<0))==0%filters bad data at end
                f_end=min(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)),427))<0));
                else f_end=size(data_neg,1);
            end

                datasize_neg=f(f_start:f_end);
                else
                    datasize_neg = f;
                end

    else disp ('Error in data limits')
end

% calculating area of whole fin
fin_dim = load('fin_shape.dat'); % loads coordinates of fin edge points
fin_x = fin_dim(:,1); % maintains edge x-coordinates
fin_y = fin_dim(:,2); % maintains edge y-coordinates
fin_area = polyarea(fin_x,fin_y); % calculates area of whole fin

% data filter
J_pos = data_pos(datasize_pos,427); % maintains Nozzle Dynamic Pressure Ratio from positive AOA DAT file, in column 427
J_neg = data_neg(datasize_neg,427); % maintains Nozzle Dynamic Pressure Ratio from negative AOA DAT file, in column 427
FA_p_pos = data_pos(datasize_pos,[87:134]) * psf2psi; % maintains 48 columns of pressure data for positive AOA Fin A
FA_p_neg = data_neg(datasize_neg,[87:134]) * psf2psi; % maintains 48 columns of pressure data for negative AOA Fin A
FB_p_pos = data_pos(datasize_pos,[135:182]) * psf2psi; % maintains 48 columns of pressure data for positive AOA Fin B
FB_p_neg = data_neg(datasize_neg,[135:182]) * psf2psi; % maintains 48 columns of pressure data for negative AOA Fin B
T_roll_pos = abs(data_pos(datasize_pos,29)); % maintains Roll Torques from positive AOA Aerodynamic Balance
T_roll_neg = abs(data_neg(datasize_neg,29)); % maintains Roll Torques from negative AOA Aerodynamic Balance

% run identification
AOA_pos = int2str (round (data_pos(1,13))); % reads angle of attack from DAT file
AOA_neg = int2str (round (data_neg(1,13))); % reads angle of attack from DAT file
M= num2str((round (data_pos(1,4)*100))/100); % reads Mach number from DAT file
Q= dir(5:7); % reads Q from DAT file
rollangle= num2str(round(data_pos(1,14))); % reads roll angle from DAT file

% loading tare pressures
tare_pressures = load ([dir(1:2) 'AOA' rollangle 'rolltare.txt']);

% function calls
T_FAodd_pos = FAodd_pos; % calls function FAodd_pos stores fin pressure, roll torques on Fin A, Odd, positive AOA
T_FAodd_neg = FAodd_neg; % calls function FAodd_neg stores fin pressure, roll torques on Fin A, Odd, negative AOA
T_FBodd_pos = FBodd_pos; % calls function FBodd_pos stores fin pressure, roll torques on Fin B, Odd, positive AOA
T_FBodd_neg = FBodd_neg; % calls function FBodd_neg stores fin pressure, roll torques on Fin B, Odd, negative AOA
T_FAeven_pos = FAden_pos; % calls function FAden_pos stores fin pressure, roll torques on Fin A, Even, positive AOA
T_FAEven_neg = FAeven_neg;%calls function FAeven_neg stores fin pressure, roll torques on Fin A, Even, negative AOA
T_FBeven_pos = FBeven_pos;%calls function FBeven_pos stores fin pressure, roll torques on Fin B, Even, positive AOA
T_FBeven_neg = FBeven_neg;%calls function FBeven_neg stores fin pressure, roll torques on Fin B, Even, negative AOA

%calculations
T_FA_pos = T_FAodd_pos + T_FAeven_pos;%determines total torque on Fin A, positive AOA
T_FA_neg = T_FAodd_neg + T_FAeven_neg;%determines total torque on Fin A, negative AOA
T_FB_pos = T_FBodd_pos + T_FBeven_pos;%determines total torque on Fin B, positive AOA
T_FB_neg = T_FBodd_neg + T_FBeven_neg;%determines total torque on Fin B, negative AOA

%interpolates J values for total fin pressure from the two sets of J data
U = [(max (J_pos(1), J_neg(1))): (range (J_pos))/350: (min (J_pos(min(length(J_pos), length(J_neg)))))];
%creates uniform matrix with approximately 350 J values from the limits of the two sets of data
T_pos = T_FA_pos + T_FB_pos;%determines total torque on the positive AOA side
T_neg = T_FA_neg + T_FB_neg;%determines total torque on the negative AOA side

TU_pos = interp1(J_pos, T_pos, U);%interpolates torque from positive AOA side onto uniform matrix
TU_neg = interp1(J_neg, T_neg, U);%interpolates torque from negative AOA side onto uniform matrix
T_pressure = TU_pos + TU_neg;%determines total roll torque

%interpolates torque from aerodynamic balance from the two sets of data
T_rollUpos = interp1(J_pos, T_roll_pos, U);%interpolates AOA balance torque data from positive AOA onto uniform J matrix
T_rollUneg = interp1(J_neg, T_roll_neg, U);%interpolates AOA balance torque data from negative AOA onto uniform J matrix
T_roll = (T_rollUpos + T_rollUneg)/2;%averages the 2 balance torques at each J value

%error analysis
T_diff = T_roll - T_pressure;%determines difference between balance and fin pressure torques
RMSdiff = num2str((mean(T_diff.^2))^0.5)%determines RMS difference between balance and fin pressure torques
format short
mean_Taero_pos = mean(T_roll_pos)%determines mean of balance torque data, positive AOA
mean_Taero_neg = mean(T_roll_neg) % determines mean of balance torque data, negative AOA
mean_Tpressure = mean(T_pressure) % determines mean of pressure torque data
stdev_Taero_pos = std(T_roll_pos) % determines standard deviation of balance torque data, positive AOA
stdev_Taero_neg = std(T_roll_neg) % determines standard deviation of balance torque data, negative AOA
stdev_Tpressure = std(T_pressure) % determines standard deviation of pressure torque data

% plots positive and negative AOA balance-measured torque vs. J and saves it
% in the directory with the DAT files
hold on
plot(J_pos, T_roll_pos, 'r', 'LineWidth', 1);
plot(J_neg, T_roll_neg, 'b', 'LineWidth', 1);
grid on
legend('T_b_a_l_a_n_c_e_p_o_s', 'T_b_a_l_a_n_c_e_n_e_g', 'Location', 'Best');
title(['Run #' run_pos ' and #' run_neg ': Balance Measured Torques']);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque [in-lb_f]');
saveas(gcf, [dir '/RUN0' run_pos run_neg 'aero'], 'jpg');
hold off
close

% plots both balance-measured torque and fin pressure torque on same axis
% and saves it in the same directory with DAT file
hold on
plot(U, T_pressure, 'r', 'LineWidth', 2);
plot(U, T_roll, 'b', 'LineWidth', 2);
plot(U, (T_pressure+27.808), 'r--', 'LineWidth', 1); % plots error in pressure torques
plot(U, (T_pressure-27.808), 'r--', 'LineWidth', 1); % plots error in pressure torques
plot(U, (T_roll+13.2827), 'b-.', 'LineWidth', 1); % plots error in balance torques
plot(U, (T_roll-13.2827), 'b-.', 'LineWidth', 1); % plots error in balance torques

text(0.025, 0.975, ['M= ' M},'sc');
text(0.025, 0.945, ['Q= ' Q],'sc');
text(0.025, 0.915, ['\alpha = ' AOA_pos],'sc');
text(0.025, 0.885, ['Roll angle= ' rollangle],'sc');
text(0.025, 0.855, ['RMS difference= ' RMSdiff],'sc');
grid on
legend('T_f_i_n', 'T_b_a_l_a_n_c_e', 'Location', 'Best');
title(['Run #' run_pos ' and #' run_neg ': Total Counter Torque']);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque [in-lb_f]');
saveas(gcf,[dir '/RUN0' run_pos run_neg 'total'],'jpg');
hold off
close

%plots total roll torque on Fin A, positive AOA versus J and saves
it in the directory with
%the DAT files
plot(J_pos, T_FA_pos, 'LineWidth', 1);
grid on
title([{'Run #' run_pos ' M ' M ', AOA=' AOA_pos ' degrees: Total
Counter Torque on Fin A'}]);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque [in-lb_f]');
saveas(gcf,[dir '/RUN0' run_pos 'finAtotal'],'jpg');
close

%plots total roll torque on Fin A versus J, negative AOA and saves
it in the directory with
%the DAT files
plot(J_neg, T_FA_neg, 'LineWidth', 1);
grid on
title([{'Run #' run_neg ' M ' M ', AOA=' AOA_neg ' degrees: Total
Counter Torque on Fin A'}]);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque [in-lb_f]');
saveas(gcf,[dir '/RUN0' run_neg 'finAtotal'],'jpg');
close

%plots total roll torque on Fin B, positive AOA versus J and saves
it in the directory with
%the DAT files
plot(J_pos, T_FB_pos, 'LineWidth', 1);
grid on
title([{'Run #' run_pos ' M ' M ', AOA=' AOA_pos ' degrees: Total
Counter Torque on Fin B'}]);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque [in-lb_f]');
saveas(gcf,[dir '/RUN0' run_pos 'finBtotal'],'jpg');
close

%plots total roll torque on Fin B, negative AOA versus J and saves
it in the directory with
%the DAT files
plot(J_neg, T_FB_neg, 'LineWidth', 1);
grid on
title([{'Run #' run_neg ' M ' M ', AOA=' AOA_neg ' degrees: Total
Counter Torque on Fin B'}]);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque [in-lb_f]');
saveas(gcf,[dir '/RUN0' run_neg 'finBtotal'],'jpg');
close
%plots total roll torque on negative AOA side versus J and saves it in the directory with %the DAT files
plot(J_neg, T_neg, 'LineWidth', 1);
grid on
title(['Run #' run_neg ' M ' M ', AOA=' AOA_neg ' degrees: Total Counter Torque']);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque [in-lb f]');
saveas(gcf,[dir '/RUN0' run_neg 'finABtotal'],'jpg');
close

%plots total roll torque on positive AOA side versus J and saves it in the directory with %the DAT files
plot(J_pos, T_pos, 'LineWidth', 1);
grid on
title(['Run #' run_pos ' M ' M ', AOA=' AOA_pos ' degrees: Total Counter Torque']);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque [in-lb f]');
saveas(gcf,[dir '/RUN0' run_pos 'finABtotal'],'jpg');
close

FAodd_pos.m

function T_FAodd_pos = FAodd_pos

%global variable declaration
global dir run_pos J_pos FA_p_pos tap_coords fin_area AOA_pos AOA_neg M data_pos tare_pressures

data filter
p = FA_p_pos(:,[1:2:47]);%maintains pressure data for Fin A, odd pressure taps, positive AOA
x = tap_coords(:,1);%maintains x coordinates for Fin A, odd pressure taps
y = tap_coords(:,2);%maintains y coordinates for Fin A, odd pressure taps
z = tap_coords(:,3);%maintains z coordinates for Fin A, odd pressure taps

%reading appropriate tare pressure from file
if str2num(dir(10:length(dir)))==8 & str2num(dir(5:7))==220
tare_pressure=tare_pressures(5);
elseif str2num(dir(10:length(dir)))==95 & str2num(dir(5:7))==220
tare_pressure=tare_pressures(13);
elseif str2num(dir(10:length(dir)))==11 & str2num(dir(5:7))==220
tare_pressure=tare_pressures(21);
elseif str2num(dir(10:length(dir)))==8 && str2num(dir(5:7))==440
tare_pressure=tare_pressures(29);
elseif str2num(dir(10:length(dir)))==95 && str2num(dir(5:7))==440
    tare_pressure=tare_pressures(37);
elseif str2num(dir(10:length(dir)))==11 && str2num(dir(5:7))==440
    tare_pressure=tare_pressures(45);
elseif str2num(dir(10:length(dir)))==8 && str2num(dir(5:7))==880
    tare_pressure=tare_pressures(53);
elseif str2num(dir(10:length(dir)))==95 && str2num(dir(5:7))==880
    tare_pressure=tare_pressures(61);
elseif str2num(dir(10:length(dir)))==11 && str2num(dir(5:7))==880
    tare_pressure=tare_pressures(69);
else disp('Error identifying tare pressure')
end

% function calls
[vertices, area_vectors, r, a_surf] = vert_and_area(x, y, z); % calls
function to determine triangle vertices,
... area vectors, triangle centroids, and area on Fin A, Odd within
tapped region
p_triangle = triangle_pressures(p, vertices); % calls function to
    determine mean pressures for each triangle
T_tot = moment(p_triangle, area_vectors, r); % calls function to
determine torque for the entire surface, Fin A: Odd

% calculations
T_tot_x = -1.*T_tot(:,1);

% plots roll torque for Fin A, Odd, positive AOA versus J and saves
it in directory with
% DAT file
plot (J_pos, T_tot_x, 'LineWidth', 1)
grid on
ylabel('Counter Torque [in-lb_f]')
xlabel('Nozzle Dynamic Pressure Ratio, J');
title(['Run #' run_pos ' M ' M ', AOA=' AOA_pos ' degrees: Fin A, 
    Odd Taps']);
saveas(gcf, [dir '/RUN0' run_pos 'finAodd'], 'jpg');
close

% scales torque to approximate the torque over the entire Fin A, Odd
T_FAodd_pos = T_tot_x*(fin_area/a_surf) - tare_pressure;

VFI_torques_AOAtare.m

clear all
clc

% global variable declaration
global dir run FA_p_pos FB_p_pos tap_coords fin_area FA_p_neg FB_p_neg

% conversions
psf2psi = 0.0069444; % convert from psf to psi

% user input
dir = input('Enter directory name (ex. C2_Q440_M8): ','s'); % the directory with DAT files
run = input('Enter the run number (ex. 0426): ','s'); % the first DAT file

% file input
data = load(['dir '/RUN0' run '.dat']); % loads entire DAT file
tap_coords = load(['fin_taps.dat']); % loads entire taps file that contains coordinates for the taps

% calculating area of whole fin
fin_dim = load('fin_shape.dat'); % loads coordinates of fin edge points
fin_x = fin_dim(:,1); % maintains edge x-coordinates
fin_y = fin_dim(:,2); % maintains edge y-coordinates
fin_area = polyarea(fin_x,fin_y); % calculates area of whole fin

% data filter
FA_p_neg = data(8,[87:134]) * psf2psi; % maintains 48 columns of pressure data for Fin A, negative AOA
FB_p_neg = data(8,[135:182]) * psf2psi; % maintains 48 columns of pressure data for Fin B, negative AOA
FA_p_pos = data(6,[87:134]) * psf2psi; % maintains 48 columns of pressure data for Fin A, positive AOA
FB_p_pos = data(6,[135:182]) * psf2psi; % maintains 48 columns of pressure data for Fin B, positive AOA

% function calls
T_FAodd_neg = FAodd_tare_neg % calls function, stores fin pressure, roll torques on Fin A, Odd, negative AOA
T_FBodd_neg = FBodd_tare_neg % calls function, stores fin pressure, roll torques on Fin B, Odd, negative AOA
T_FAeven_neg = FAEven_tare_neg % calls function, stores fin pressure, roll torques on Fin A, Even, negative AOA
T_FBeven_neg = FBeven_tare_neg % calls function, stores fin pressure, roll torques on Fin B, Even, negative AOA
T_FAodd_pos = FAodd_tare_pos % calls function, stores fin pressure, roll torques on Fin A, Odd, positive AOA
T_FBodd_pos = FBodd_tare_pos % calls function, stores fin pressure, roll torques on Fin B, Odd, positive AOA
T_FAeven_pos = FAEven_tare_pos % calls function, stores fin pressure, roll torques on Fin A, Even, positive AOA
T_FBeven_pos = FBeven_tare_pos % calls function, stores fin pressure, roll torques on Fin B, Even, positive AOA
T_FA_pos = T_FAodd_pos + T_FAeven_pos\%determines total torque on Fin A, positive AOA
T_FA_neg = T_FAodd_neg + T_FAeven_neg\%determines total torque on Fin A, negative AOA
T_FB_pos = T_FBodd_pos + T_FBeven_pos\%determines total torque on Fin B, positive AOA
T_FB_neg = T_FBodd_neg + T_FBeven_neg\%determines total torque on Fin B, negative AOA
T_FAB_pos = T_FA_pos + T_FB_pos\%determines total torque on Fin A, positive AOA
T_FAB_neg = T_FA_neg + T_FB_neg\%determines total torque on Fin B, negative AOA

T_sum = T_FAB_pos + T_FAB_neg;\%determines sum of positive and negative AOA torques
T_ratio = T_sum / T_FAB_pos;\%determines ratio of summed torque to a torque on a single side

VFI_torques_0tare.m

clear all
clc

%global variable declaration
global dir run J FA_p FB_p tap_coords fin_area

% conversions
psf2psi = 0.0069444; \%convert from psf to psi

% user input
dir = input('Enter directory name (ex. C2_Q440_M8): ','s'); \%the directory with DAT files
run = input('Enter the run number (ex. 0426): ','s'); \%the first DAT file

% file input
data = load([dir '/RUN0' run '.dat']); \%loads entire DAT file
tap_coords = load(['fin_taps.dat']); \%loads entire taps file that contains coordinates for the taps

% calculating area of whole fin
fin_dim = load('fin_shape.dat'); \%loads coordinates of fin edge points
fin_x = fin_dim(:,1); \%maintains edge x-coordinates
fin_y = fin_dim(:,2); \%maintains edge y-coordinates
fin_area = polyarea(fin_x,fin_y); \%calculates area of whole fin

% data filter
J = data(:, 427); % maintains Nozzle Dynamic Pressure Ratio from DAT file, in column 427
FA_p = data(2, [87:134]) * psf2psi; % maintains 48 columns of pressure data for Fin A
FB_p = data(2, [135:182]) * psf2psi; % maintains 48 columns of pressure data for Fin B

% function calls
T_FAodd = FAodd0 % calls function FAodd0 stores fin pressure, roll torques on Fin A, Odd
T_FBodd = FBodd0 % calls function FBodd0 stores fin pressure, roll torques on Fin B, Odd
T_FAeven = FAeven0 % calls function FAeven0 stores fin pressure, roll torques on Fin A, Even
T_FBeven = FBeven0 % calls function FBeven0 stores fin pressure, roll torques on Fin B, Even

FAeven0.m

function T_FAeven = FAeven

% global variable declaration
global dir run J FA_p tap_coords fin_area

% data filter
p = FA_p(:, [2:2:48]); % maintains pressure data for Fin A, even pressure taps
x = tap_coords(:, 4); % maintains x coordinates for Fin A, even pressure taps
y = tap_coords(:, 5); % maintains y coordinates for Fin A, even pressure taps
z = tap_coords(:, 6); % maintains z coordinates for Fin A, even pressure taps

% function calls
[vertices, area_vectors, r, a_surf] = vert_and_area(x, y, z); % calls function to determine triangle vertices, ... area vectors, triangle centroids, and area on Fin A, Even within tapped region
p_triangle = triangle_pressures(p, vertices); % calls function to determine mean pressures for each triangle
T_tot = moment(p_triangle, area_vectors, r); % calls function to determine torque for the entire surface, Fin A:Even
% calculations
T_tot_x = T_tot(:, 1);

% scales torque to approximate the torque over the entire Fin A, Even
T_FAeven = T_tot_x * (fin_area/a_surf);
% plots counter torque coefficient for a Mach number at various Q
% clear all
clc
mkdir('CTplots')

% global variable declaration
global dir J FA_p FB_p tap_coords fin_area
global M data tare_pressures

% conversions
psf2psi = 0.0069444; % convert from psf to psi

% user input
config = input('Enter configuration (ex. 2, 3, 4): ','s'); % the configuration
M = input('Enter Mach number (0.8, 0.95, 1.1): ','s'); % Mach number
rollangle = input('Enter roll angle (0, 45): ','s'); % roll angle
if str2num(config)~=2 & str2num(config)~=3 & str2num(config)~=4
    disp('Error in configuration input'); end
if str2num(M)~=0.8 & str2num(M)~=0.95 & str2num(M)~=1.1
    disp('Error in Mach number input'); end
if str2num(rollangle)~=0 & str2num(rollangle)~=45
    disp('Error in roll angle input'); end

% identifying first input file for given configuration, Mach number, and
... roll angle
if str2num(rollangle) == 0
    if str2num(config)==2 & str2num(M)==0.8
        file1='C2_Q220_M8/RUN00376.dat'; end
        if str2num(config)==2 & str2num(M)==0.95
            file1='C2_Q220_M95/RUN00386.dat'; end
            if str2num(config)==2 & str2num(M)==1.1
                file1='C2_Q220_M11/RUN00396.dat'; end

    else
        if str2num(config)==3 & str2num(M)==0.8
            file1='C3_Q220_M8/RUN00401.dat'; end
            if str2num(config)==3 & str2num(M)==0.95
                file1='C3_Q220_M95/RUN00411.dat'; end
                if str2num(config)==3 & str2num(M)==1.1
                    file1='C3_Q220_M11/RUN00421.dat'; end

            else
                if str2num(config)==4 & str2num(M)==0.8
                    file1='C4_Q220_M8/RUN00495.dat'; end
                    if str2num(config)==4 & str2num(M)==0.95
                        file1='C4_Q220_M95/RUN00505.dat'; end
                        if str2num(config)==4 & str2num(M)==1.1
                            file1='C4_Q220_M11/RUN00515.dat'; end
                            end

end

60
if str2num(rollangle) == 45
    if str2num(config) == 2 & str2num(M) == 0.8
        file1 = 'C2_Q220_M8/RUN00381.dat'; end
    else if str2num(config) == 2 & str2num(M) == 0.95
        file1 = 'C2_Q220_M95/RUN00391.dat'; end
    else if str2num(config) == 2 & str2num(M) == 1.1
        file1 = 'C2_Q220_M11/RUN00401.dat'; end
    end
if str2num(config) == 3 & str2num(M) == 0.8
    file1 = 'C3_Q220_M8/RUN00516.dat'; end
if str2num(config) == 3 & str2num(M) == 0.95
    file1 = 'C3_Q220_M95/RUN00526.dat'; end
if str2num(config) == 3 & str2num(M) == 1.1
    file1 = 'C3_Q220_M11/RUN00536.dat'; end
end
if str2num(config) == 4 & str2num(M) == 0.8
    file1 = 'C4_Q220_M8/RUN00600.dat'; end
if str2num(config) == 4 & str2num(M) == 0.95
    file1 = 'C4_Q220_M95/RUN00610.dat'; end
if str2num(config) == 4 & str2num(M) == 1.1
    file1 = 'C4_Q220_M11/RUN00620.dat'; end
end

if str2num(config) == 2
    n = 3;
else
    n = 2;
end

% determining counter torque coefficient for varying Q
for o = 1:n
    % file input
    data = load([file1]); % loads appropriate DAT file
    tap_coords = load(['fin_taps.dat']); % loads tap coordinates
    if str2num(M) == 0.8 dir = file1(1:10); end % identifies directory
    if str2num(M) == 0.95 dir = file1(1:11); end % identifies directory
    % determining minimum and maximum J values allowed for data
    if abs(data(1,7)-220<5) a = find (data(:,427)>4.6, 1); end
    if abs(data(1,7)-440<5) b = find (data(:,427)>2.3, 1); end
    if abs(data(1,7)-880<5) c = find (data(:,427)>1.1, 1); end
    if abs(data(1,7)-220<5) i = find (data(:,427)>73.7, 1); end
    if abs(data(1,7)-440<5) j = find (data(:,427)>36.8, 1); end
    if abs(data(1,7)-880<5) k = find (data(:,427)>18.4, 1); end
    % finds point at which J begins to decrease and discards subsequent data
    if isempty(i) == 1 i(1) =
        min(find(diff(data(100:size(data,1),427))<0)); end
    }
if isempty(j)==1 j(1) = 
min(find(diff(data(100:size(data,1),427))<0)); end 
if isempty(k)==1 k(1) = 
min(find(diff(data(100:size(data,1),427))<0)); end 

if (round ((data(1,7)/10)))*10==220%minimum J value for Q220 
data 
minsize = a(1);
elseif (round ((data(1,7)/10)))*10==440%minimum J value for Q440 
data 
minsize = b(1);
else minsize = c(1);%minimum J value for Q880 data
end 

if (round ((data(1,7)/10)))*10==220%maximum J value for Q220 
data 
maxsize = i(1);
elseif (round ((data(1,7)/10)))*10==440%maximum J value for Q440 
data 
maxsize = j(1);
else maxsize = k(1);%maximum J value for Q880 data
end

%calculating area of whole fin
fin_dim = load('fin_shape.dat');%loads coordinates of fin edge points 
fin_x = fin_dim(:,1);%maintains edge x-coordinates
fin_y = fin_dim(:,2);%maintains edge y-coordinates
fin_area = polyarea(fin_x,fin_y);%calculates area of whole fin

%data filter
J(1:size(data,1),o) = data(:,427);%maintains Nozzle Dynamic Pressure Ratio from DAT file, in column 427
FA_p = data(:,[87:134]) * psf2psi;%maintains 48 columns of pressure data for Fin A
FB_p = data(:,[135:182]) * psf2psi;%maintains 48 columns of pressure data for Fin B
T_roll = abs(data(:,29));%maintains Roll Torques from Aerodynamic Balance

%AoA = int2str (round (data(1,13)));%reads angle of attack from DAT file
rollangle= num2str(round(data(1,14)));%reads roll angle from DAT file
Qpsi=data(1,12);%reads free stream dynamic pressure from DAT file

%loading tare pressures
  tare_pressures = load (['C' config '_0AOA_' rollangle 'rolltare.txt'])
%function calls
T_FAodd = FAoddCT;%calls function FAodd stores fin pressure, roll torques on Fin A, Odd
T_FBodd = FBoddCT;%calls function FBodd stores fin pressure, roll torques on Fin B, Odd
T_FAeven = FAevenCT;%calls function FAeven stores fin pressure, roll torques on Fin A, Even
T_FBeven = FBevenCT;%calls function FBeven stores fin pressure, roll torques on Fin B, Even

%calculations
T_pressure = 2*(T_FAodd + T_FAeven + T_FBodd + T_FBeven);%determines total torque from fin pressures, ...
... factor of 2 is for the other 2 symmetric fins
CCT_fin(1:size(data,1),o) =
(T_pressure)/(Qpsi*pi/4*(13.3)^3);%calculates coefficient from pressure data
CCT_roll(1:size(data,1),o) =
(T_roll)/(Qpsi*pi/4*(13.3)^3);%calculates coefficient from balance data

%error analysis
%Q uncertainties from AEDC
if 215<=(round (data(1,7)/10))*10<=225 && str2num(M)==0.8
Uq=1.2786; end
if 215<=(round (data(1,7)/10))*10<=225 && str2num(M)==0.95
Uq=1.0391; end
if 215<=(round (data(1,7)/10))*10<=225 && str2num(M)==1.1
Uq=0.7639; end
if 435<=(round (data(1,7)/10))*10<=445 && str2num(M)==0.8
Uq=1.3004; end
if 435<=(round (data(1,7)/10))*10<=445 && str2num(M)==0.95
Uq=1.0541; end
if 435<=(round (data(1,7)/10))*10<=445 && str2num(M)==1.1
Uq=0.7708; end
if 875<=(round (data(1,7)/10))*10<=885 && str2num(M)==0.8
Uq=1.4525; end
if 875<=(round (data(1,7)/10))*10<=885 && str2num(M)==0.95
Uq=1.1365; end
if 875<=(round (data(1,7)/10))*10<=885 && str2num(M)==1.1
Uq=0.8102; end

%computing uncertainty in pressure and balance derived counter torque coefficient
for a=1:length(T_pressure)
    U_CCT_fin(a,o)= sqrt((1/(Qpsi*pi/4*13.3^3)*39.3264)^2 + (T_pressure(a)/(Qpsi*pi/4*13.3^3)*Uq/144)^2);
end

%setting up next run
if str2num(M)==0.8
    if o==1
        p=num2str(str2num(file1(17:19))+50);
        q=num2str(str2num(file1(5:7))*2);
    elseif o==2
        p=num2str(str2num(file1(17:19))+25);
        q=num2str(str2num(file1(5:7))*2);
    end
    file1(17:19)=p;
    file1(5:7)=q;
end

if str2num(M)==0.95
    if o==1
        p=num2str(str2num(file1(18:20))+30);
        q=num2str(str2num(file1(5:7))*2);
    elseif o==2
        p=num2str(str2num(file1(18:20))+45);
        q=num2str(str2num(file1(5:7))*2);
    end
    file1(18:20)=p;
    file1(5:7)=q;
end

if str2num(M)==1.1
    if o==1
        p=num2str(str2num(file1(18:20))+10);
        q=num2str(str2num(file1(5:7))*2);
    elseif o==2
        p=num2str(str2num(file1(18:20))+65);
        q=num2str(str2num(file1(5:7))*2);
    end
    file1(18:20)=p;
    file1(5:7)=q;
end
end

% plots both coefficients from pressure and balance data and error bars
hold on
% determining final data point for each set of data
r=find(CCT_roll(:,1)==0,1)-1;
if isempty(r) r=size(CCT_roll,1); end
s=find(CCT_roll(:,2)==0,1)-1;
if isempty(s) s=size(CCT_roll,1); end
if str2num(config)==2
    t=find(CCT_roll(:,3)==0,1)-1;
    if isempty(t) t=size(CCT_roll,1); end
end

if str2num(config)==2
    Jlimit=J(t,3);
else Jlimit=J(s,2);
end

%increment for error bar
b=1:40:r;
c=1:40:s;
if str2num(config)==2
d=1:40:t;
end

%plotting coefficients and error bars
plot(J(1:r,1),CCT_fin(1:r,1), 'r', 'LineWidth', 1);%Q220 pressure
plot(J(1:r,1),CCT_roll(1:r,1), 'b', 'LineWidth', 1);%Q220 balance
plot(J(1:s,2),CCT_fin(1:s,2), 'r--', 'LineWidth', 1);%Q440 pressure
plot(J(1:s,2),CCT_roll(1:s,2), 'b--', 'LineWidth', 1);%Q440 balance

if str2num(config)==2%Q880
    plot(J(1:t,3),CCT_fin(1:t,3), 'r:', 'LineWidth', 1);
    plot(J(1:t,3),CCT_roll(1:t,3), 'b:', 'LineWidth', 1);
    errorbar(J(d,3),CCT_fin(d,3),U_CCT_fin(d,3),U_CCT_fin(d,3),'r.');
    errorbar(J(d,3),CCT_roll(d,3),U_CCT_roll(d,3),U_CCT_roll(d,3),'b.');
end

Q220dev=mean(abs(CCT_fin(1:find(abs(J(1:r,1)-Jlimit)<1,1),1)-
    CCT_roll(1:find(abs(J(1:r,1)-Jlimit)<1,1),1)));
Q440dev=mean(abs(CCT_fin(1:find(abs(J(1:s,2)-Jlimit)<1,1),2)-
    CCT_roll(1:find(abs(J(1:s,2)-Jlimit)<1,1),2)));
if str2num(config)==2 Q880dev=mean(abs(CCT_fin(1:t,3)-
    CCT_roll(1:t,3))); end

%Q220 & Q440 error bars
errorbar(J(b,1),CCT_fin(b,1),U_CCT_fin(b,1),U_CCT_fin(b,1),'r.');
errorbar(J(b,1),CCT_roll(b,1),U_CCT_roll(b,1),U_CCT_roll(b,1),'b.');
errorbar(J(c,2),CCT_fin(c,2),U_CCT_fin(c,2),U_CCT_fin(c,2),'r.');
errorbar(J(c,2),CCT_roll(c,2),U_CCT_roll(c,2),U_CCT_roll(c,2),'b.');

if str2num(config)~=2
    legend (['Q220 Pressure C_C_T'], ['Q220 Balance C_C_T'], ['Q440
    Pressure C_C_T'], ['Q440 Balance C_C_T'], 'Location', 'Best')
else legend (['Q220 Pressure C_C_T'], ['Q220 Balance C_C_T'], ['Q440
    Pressure C_C_T'], ['Q440 Balance C_C_T'], ['Q880 Pressure C_C_T'],
    ['Q880 Balance C_C_T'],'Location', 'Best')
end

text(0.025, 0.985,['\alpha = ' AOA ''],'sc');%displays angle of
attack
text(0.025, 0.955,['Roll angle= ' rollangle ''],'sc');%displays roll
angle
grid on
title(['Counter Torque Coefficient at Mach' ' M ', Configuration' config ']);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque Coefficient (C_C_T)');
saveas(gcf,['CTplots/Mach' ' M ', Config' config ' rollangle 'roll.jpg'], 'jpg');
hold off
close

% plotting deviation between pressure and balance measured coefficients
hold on
plot(J(1:r,1),abs(CCT_fin(1:r,1)-CCT_roll(1:r,1)), 'r', 'LineWidth', 1);
plot(J(1:s,2),abs(CCT_fin(1:s,2)-CCT_roll(1:s,2)), 'b', 'LineWidth', 1);
if str2num(config)==2 plot(J(1:t,3),abs(CCT_fin(1:t,3)-CCT_roll(1:t,3)), 'k', 'LineWidth', 1);
end
if str2num(config)~=2
    legend ('Q220 deviation', 'Q440 deviation', 'Location', 'Best');
elseif str2num(config)==2
    legend ('Q220 deviation', 'Q440 deviation', 'Q880 deviation', 'Location', 'Best');
end

text(0.025, 0.985, ['\alpha = AOA ', 'sc']);
text(0.025, 0.955, ['Roll angle= rollangle ', 'sc']);
text(0.025, 0.925, ['Q220 deviation= ', num2str(Q220dev), ' '], 'sc');
text(0.025, 0.895, ['Q440 deviation= ', num2str(Q440dev), ' '], 'sc');
if str2num(config)==2 text(0.025, 0.865, ['Q880 deviation= ', num2str(Q880dev), ' '], 'sc');
end

grid on
title(['Deviation Between Pressure and Balance C_C_T at Mach' ' M ', Configuration' config ']);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque Coefficient (C_C_T) Deviation');
saveas(gcf,['CTplots/Mach' ' M ', Config' config '_' rollangle 'roll_deviation.jpg'], 'jpg');
hold off
close
function T_FAodd = FAoddCT

%global variable declaration
global dir run J FA_p tap_coords fin_area tare_pressures

%data filter
p = FA_p(:,[1:2:47]);%maintains pressure data for Fin B, odd pressure taps
x = tap_coords(:,1);%maintains x coordinates for Fin A, odd pressure taps
y = tap_coords(:,2);%maintains y coordinates for Fin A, odd pressure taps
z = tap_coords(:,3);%maintains z coordinates for Fin A, odd pressure taps

%reading appropriate tare pressure from file
if str2num(dir(10:length(dir)))==8 && str2num(dir(5:7))==220
tare_pressure=tare_pressures(1);
elseif str2num(dir(10:length(dir)))==95 && str2num(dir(5:7))==220
tare_pressure=tare_pressures(5);
elseif str2num(dir(10:length(dir)))==11 && str2num(dir(5:7))==220
tare_pressure=tare_pressures(9);
elseif str2num(dir(10:length(dir)))==8 && str2num(dir(5:7))==440
tare_pressure=tare_pressures(13);
elseif str2num(dir(10:length(dir)))==95 && str2num(dir(5:7))==440
tare_pressure=tare_pressures(17);
elseif str2num(dir(10:length(dir)))==11 && str2num(dir(5:7))==440
tare_pressure=tare_pressures(21);
elseif str2num(dir(10:length(dir)))==8 && str2num(dir(5:7))==880
tare_pressure=tare_pressures(25);
elseif str2num(dir(10:length(dir)))==95 && str2num(dir(5:7))==880
tare_pressure=tare_pressures(29);
elseif str2num(dir(10:length(dir)))==11 && str2num(dir(5:7))==880
tare_pressure=tare_pressures(34);
else disp('Error identifying tare pressure')
end

%function calls
[vertices, area_vectors, r, a_surf] = vert_and_area(x, y, z);%calls function to determine triangle vertices,
...area vectors, triangle centroids, and area on Fin A, Odd within tapped region
p_triangle = triangle_pressures(p, vertices);%calls function to determine mean pressures for each triangle
T_tot = moment(p_triangle, area_vectors, r);%calls function to determine torque for the entire surface, Fin A:Odd

%calculations
\[ T_{\text{tot}x} = -1 \cdot T_{\text{tot}}(:,1); \]

scales torque to approximate the torque over the entire Fin A, Odd
\[ T_{\text{FAodd}} = T_{\text{tot}x} \cdot (\text{fin\_area}/a_{\text{surf}}) - tare_{\text{pressure}}; \]

**VFI_CTcoeff_AOA.m**

%plots counter-torque coefficient at angle of attack
clear all
clc
format long

%global variable declaration
global dir run_pos run_neg J_pos J_neg FA_p_pos FA_p_neg FB_p_pos FB_p_neg tap_coords fin_area
global AOA_pos AOA_neg M data_pos data_neg tare_pressures

%conversions
psf2psi = 0.0069444;%convert from psf to psi

%user input
config = input('Enter configuration (ex. 2, 3, 4): ','s');%the configuration
M = input('Enter Mach number (0.8, 0.95, 1.1): ','s');%Mach number
rollangle = input('Enter roll angle (0, 45): ','s');%roll angle
if str2num (config)~=2 & str2num (config)~=3 & str2num (config)~=4
    disp('Error in configuration input'); end
if str2num (M)~=0.8 & str2num (M)~=0.95 & str2num (M)~=1.1
    disp('Error in Mach number input'); end

%identifying first input file for given configuration, Mach number, and
...roll angle
if str2num(rollangle) == 0
    if str2num(config)==2 & str2num(M)==0.8
        file1='C2_Q220_M8/RUN00375.dat'; end
    if str2num(config)==2 & str2num(M)==0.95
        file1='C2_Q220_M95/RUN00385.dat'; end
    if str2num(config)==2 & str2num(M)==1.1
        file1='C2_Q220_M11/RUN00395.dat'; end
    if str2num(config)==3 & str2num(M)==0.8
        file1='C3_Q220_M8/RUN00510.dat'; end
    if str2num(config)==3 & str2num(M)==0.95
        file1='C3_Q220_M95/RUN00520.dat'; end
    if str2num(config)==3 & str2num(M)==1.1
        file1='C3_Q220_M11/RUN00530.dat'; end
    if str2num(config)==4 & str2num(M)==0.8
        file1='C4_Q220_M8/RUN00594.dat'; end
    else
        disp('Error in roll angle input'); end
end
if str2num(config) == 4 & str2num(M) == 0.95
    file1 = 'C4_Q220_M95/RUN00604.dat'; end
if str2num(config) == 4 & str2num(M) == 1.1
    file1 = 'C4_Q220_M11/RUN00614.dat'; end
end

if str2num(rollangle) == 45
    if str2num(config) == 2 & str2num(M) == 0.8
        file1 = 'C2_Q220_M8/RUN00382.dat'; end
    if str2num(config) == 2 & str2num(M) == 0.95
        file1 = 'C2_Q220_M95/RUN00392.dat'; end
    if str2num(config) == 2 & str2num(M) == 1.1
        file1 = 'C2_Q220_M11/RUN00402.dat'; end
    end
    if str2num(config) == 3 & str2num(M) == 0.8
        file1 = 'C3_Q220_M8/RUN00517.dat'; end
    if str2num(config) == 3 & str2num(M) == 0.95
        file1 = 'C3_Q220_M95/RUN00527.dat'; end
    if str2num(config) == 3 & str2num(M) == 1.1
        file1 = 'C3_Q220_M11/RUN00537.dat'; end
    end
    if str2num(config) == 4 & str2num(M) == 0.8
        file1 = 'C4_Q220_M8/RUN00601.dat'; end
    if str2num(config) == 4 & str2num(M) == 0.95
        file1 = 'C4_Q220_M95/RUN00611.dat'; end
    if str2num(config) == 4 & str2num(M) == 1.1
        file1 = 'C4_Q220_M11/RUN00621.dat'; end
end

if str2num(config) == 2
    n = 3;
else
    n = 2;
end

%determining counter torque coefficient for varying Q
for o = 1:n
    file2 = file1;
    if str2num(rollangle) == 0
        file2(end-4) = num2str(str2num(file1(end-4)) + 2);
    end
    if str2num(rollangle) == 45
        file2(end-4) = num2str(str2num(file1(end-4)) - 2);
    end
end

%file input
data_neg = load(fullfile(file1)); %loads negative AOA DAT file
data_pos = load(fullfile(file2)); %loads positive AOA DAT file
tap_coords = load(fullfile('fin_taps.dat')); %loads tap coordinates
if str2num(M) == 0.8
    dir = fullfile(1:10);
else
    dir = fullfile(1:11);
end

%constructing range of J values allowed for data
a = find((4.6 < data_pos(:, 427) < 73.7)); % J limits on Q220 data, positive AOA
b = find (2.3 < data_pos(:, 427) < 36.8); \% J limits on Q440 data, positive AOA

c = find (1.1 < data_pos(:, 427) < 18.4); \% J limits on Q880 data, positive AOA

d = find (4.6 < data_neg(:, 427) < 73.7); \% J limits on Q220 data, negative AOA

e = find (2.3 < data_neg(:, 427) < 36.8); \% J limits on Q440 data, negative AOA

f = find (1.1 < data_neg(:, 427) < 18.4); \% J limits on Q880 data, negative AOA

\% sets range of J values for Q220 data, positive AOA
if str2num(dir(5:7)) == 220 && round(data_pos(1, 13)) == 4
    if isempty(find(diff(data_pos(:, 427)) < 0)) == 0 \% filtering out bad data (decreasing J values)
        if min(find(diff(data_pos(1: min(size(data_neg, 1), size(data_pos, 1)), 427)) < 0)) < 20 \% filters bad data at beginning
            s = find(diff(data_pos(1: min(size(data_neg, 1), size(data_pos, 1)), 427)) < 0);
                s = 1 + max(s(s < 20));
                a_start = s;
            else
                a_start = 1;
            end
        if isempty(find(diff(data_pos(100: min(size(data_neg, 1), size(data_pos, 1)), 427)) < 0)) == 0 \% filters bad data at end
            a_end = min(find(diff(data_pos(100: min(size(data_neg, 1), size(data_pos, 1)), 427)) < 0) + 100);
        else
            a_end = size(data_pos, 1);
        end
        datasize_pos = a(a_start:a_end);
    else
        datasize_pos = a;
    end
\end

\% sets range of J values for Q440 data, positive AOA
elseif str2num(dir(5:7)) == 440 && round(data_pos(1, 13)) == 4
    if isempty(find(diff(data_pos(:, 427)) < 0)) == 0 \% filtering out bad data (decreasing J values)
        if min(find(diff(data_pos(1: min(size(data_neg, 1), size(data_pos, 1)), 427)) < 0)) < 20 \% filters bad data at beginning
            s = find(diff(data_pos(1: min(size(data_neg, 1), size(data_pos, 1)), 427)) < 0);
                s = 1 + max(s(s < 20));
                b_start = s;
        else
            b_start = 1;
        end
    else
        b_start = size(data_pos, 1);
    end

else b_start=1; 
end

if isempty(find(diff(data_pos(100:min(size(data_neg,1),size(data_pos,1)),427))<0))==0%filters bad data at end 
b_end= min(find(diff(data_pos(100:min(size(data_neg,1),size(data_pos,1)),427))<0)+100); 
else b_end=size(data_pos,1); 
end 

datasize_pos=b(b_start:b_end);
else 
    datasize_pos = b;
end

%sets range of J values for Q880 data, positive AOA 
elseif str2num(dir(5:7))==880 && round (data_pos(1,13))==4 
    if isempty(find(diff(data_pos(:,427))<0))==0%filtering out bad data (decreasing J values) 
        if min(find(diff(data_pos(1:min(size(data_neg,1),size(data_pos,1)),427))<0))<20%filters bad data at beginning 
            s=find(diff(data_pos(1:min(size(data_neg,1),size(data_pos,1)),427))<0); 
            s=1+max(s(s<20)); 
            c_start=s; 
        else c_start=1; 
        end 
    end 
    if isempty(find(diff(data_pos(100:min(size(data_neg,1),size(data_pos,1)),427))<0))==0%filters bad data at end 
        c_end= min(find(diff(data_pos(100:min(size(data_neg,1),size(data_pos,1)),427))<0)+100); 
        else c_end=size(data_pos,1); 
        end 
        datasize_pos=c(c_start:c_end);
    else 
        datasize_pos = c;
end 
else disp ('Error in data limits') 
end

%sets range of J values for Q220 data, negative AOA 
if str2num(dir(5:7))==220 && round (data_neg(1,13))==-4
if isempty(find(diff(data_neg(:,427))<0))==0%filtering out
bad data (decreasing J values)
if
min(find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427)
)<0))<20%filters bad data at beginning

t=find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427))<
0);
    t=1+max(t(t<20));
    d_start=t;
else d_start=1;
end

if
isempty(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)
)),427))<0))==0%filters bad data at end
    d_end=
min(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)),427))<0)+100);
else d_end=size(data_neg,1);
end

datasize_neg=d(d_start:d_end);
else
    datasize_neg = d;
end

%sets range of J values for Q440 data, negative AOA
elseif str2num(dir(5:7))==440 && round (data_neg(1,13))==-4
    if isempty(find(diff(data_neg(:,427))<0))==0%filtering out
bad data (decreasing J values)
if
min(find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427)
)<0))<20%filters bad data at beginning

t=find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427))<
0);
    t=1+max(t(t<20));
    e_start=t;
else e_start=1;
end

if
isempty(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)
)),427))<0))==0%filters bad data at end
    e_end=
min(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)),427))<0)+100);
else e_end=size(data_neg,1);
end

datasize_neg=e(e_start:e_end);
else
    datasize_neg = e;
end

% sets range of J values for Q880 data, negative AOA
elseif str2num(dir(5:7))==880 && round (data_neg(1,13))==-4
    if isempty(find(diff(data_neg(:,427))<0))==0%filtering out bad data (decreasing J values)
        if min(find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427)<0))<20%filters bad data at beginning
            t=find(diff(data_neg(1:min(size(data_neg,1),size(data_pos,1)),427))<0);
            t=1+max(t(t<20));
            f_start=t;
            else f_start=1;
        end
        if isempty(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)),427))<0))==0%filters bad data at end
            f_end=min(find(diff(data_neg(100:min(size(data_neg,1),size(data_pos,1)),427))<0)+100);
            else f_end=size(data_neg,1);
        end
        datasize_neg=f(f_start:f_end);
    else disp ('Error in data limits')
end

% calculating area of whole fin
fin_dim = load('fin_shape.dat');%loads coordinates of fin edge points
fin_x = fin_dim(:,1);%maintains edge x-coordinates
fin_y = fin_dim(:,2);%maintains edge y-coordinates
fin_area = polyarea(fin_x,fin_y);%calculates area of whole fin

% data filter
J_pos = data_pos(datasize_pos,427);%maintains Nozzle Dynamic Pressure Ratio from positive AOA DAT file, in column 427
J_neg = data_neg(datasize_neg,427);%maintains Nozzle Dynamic Pressure Ratio from negative AOA DAT file, in column 427
FA_p_pos = data_pos(datasize_pos,[87:134]) * psf2psi;%maintains 48 columns of pressure data for positive AOA Fin A
FA_p_neg = data_neg(datasize_neg,[87:134]) * psf2psi;%maintains 48 columns of pressure data for negative AOA Fin A
FB_p_pos = data_pos(datasize_pos,[135:182]) * psf2psi;%maintains 48 columns of pressure data for positive AOA Fin B
FB_p_neg = data_neg(datasize_neg,[135:182]) * psf2psi;%maintains 48 columns of pressure data for negative AOA Fin B
T_roll_pos = abs(data_pos(datasize_pos,29));%maintains Roll Torques from positive AOA Aerodynamic Balance
T_roll_neg = abs(data_neg(datasize_neg,29));%maintains Roll Torques from negative AOA Aerodynamic Balance
Qpsi_pos=data_pos(datasize_pos,12);
Qpsi_neg=data_neg(datasize_neg,12);

%run identification
AOA = int2str (round (data_pos(1,13)));%reads angle of attack from DAT file
M= num2str((round (data_pos(1,4)*100))/100);%reads Mach number from DAT file
Q= dir(5:7);%reads Q from DAT file

tare_pressures = load ([dir(1:2) 'AOA' rollangle 'rolltare.txt']);

T_FAodd_pos = FAodd_posCT;%calls function FAodd_pos stores fin pressure, roll torques on Fin A, Odd, positive AOA
T_FAodd_neg = FAodd_negCT;%calls function FAodd_neg stores fin pressure, roll torques on Fin A, Odd, negative AOA
T_FBodd_pos = FBodd_posCT;%calls function FBodd_pos stores fin pressure, roll torques on Fin B, Odd, positive AOA
T_FBodd_neg = FBodd_negCT;%calls function FBodd_neg stores fin pressure, roll torques on Fin B, Odd, negative AOA
T_FAeven_pos = FAeven_posCT;%calls function FAeven_pos stores fin pressure, roll torques on Fin A, Even, positive AOA
T_FAeven_neg = FAeven_negCT;%calls function FAeven_neg stores fin pressure, roll torques on Fin A, Even, negative AOA
T_FBeven_pos = FBeven_posCT;%calls function FBeven_pos stores fin pressure, roll torques on Fin B, Even, positive AOA
T_FBeven_neg = FBeven_negCT;%calls function FBeven_neg stores fin pressure, roll torques on Fin B, Even, negative AOA

T_FA_pos= T_FAodd_pos + T_FAeven_pos;%determines total torque on Fin A, positive AOA
T_FA_neg= T_FAodd_neg + T_FAeven_neg;%determines total torque on Fin A, negative AOA
T_FB_pos= T_FBodd_pos + T_FBeven_pos;%determines total torque on Fin B, positive AOA
T_FB_neg= T_FBodd_neg + T_FBeven_neg;%determines total torque on Fin B, negative AOA

%interpolates J values for total fin pressure from the two sets of J data
U_1=[(max (J_pos(1), J_neg(1))):(range (J_pos))/350:(min (J_pos(min(length(J_pos),length(J_neg))), J_neg(min(length(J_pos),length(J_neg)))))';

% creates uniform vector with approximately 350 J values from the limits of the two sets of data
U(1:length(U_1),o) = U_1;% assigns vector a column in array U
T_pos= T_FA_pos + T_FB_pos;% determines total torque on the positive AOA side
T_neg= T_FA_neg + T_FB_neg;% determines total torque on the negative AOA side
TU_pos= interp1(J_pos,T_pos,U(:,o));% interpolates torque from positive AOA side onto uniform array
TU_neg= interp1(J_neg,T_neg,U(:,o));% interpolates torque from negative AOA side onto uniform array
Qpsi_posU= interp1(J_pos,Qpsi_pos,U(:,o));% interpolates Q from positive AOA side onto uniform array
Qpsi_negU= interp1(J_neg,Qpsi_neg,U(:,o));% interpolates Q from negative AOA side onto uniform array

% interpolates torque from aerodynamic balance from the two sets of data
T_rollUpos= interp1(J_pos,T_roll_pos,U(:,o));% interpolates balance torque data from positive AOA onto uniform J matrix
T_rollUneg= interp1(J_neg,T_roll_neg,U(:,o));% interpolates balance torque data from negative AOA onto uniform J matrix

CCT_fin(1:size(U,1),o) =
(TU_pos)./(Qpsi_posU)/(pi/4*(13.3)^3)+(TU_neg)./(Qpsi_negU)/(pi/4*(13.3)^3);% calculates coefficient from pressure data
CCT_roll(1:size(U,1),o) =
((T_rollUpos)./(Qpsi_posU)/(pi/4*(13.3)^3)+(T_rollUneg)./(Qpsi_negU)/(pi/4*(13.3)^3))/2;% calculates coefficient from balance data

% error analysis
% Q uncertainties from AEDC
if 215<=(round (data_pos(1,7)/10))*10<=225 && str2num(M)==0.8
Uq=1.2786; end
if 215<=(round (data_pos(1,7)/10))*10<=225 && str2num(M)==0.95
Uq=1.0391; end
if 215<=(round (data_pos(1,7)/10))*10<=225 && str2num(M)==1.1
Uq=0.7639; end
if 435<=(round (data_pos(1,7)/10))*10<=445 && str2num(M)==0.8
Uq=1.3004; end
if 435<=(round (data_pos(1,7)/10))*10<=445 && str2num(M)==0.95
Uq=1.0541; end
if 435<=(round (data_pos(1,7)/10))*10<=445 && str2num(M)==1.1
Uq=0.7708; end
if 875<=(round (data_pos(1,7)/10))*10<=885 && str2num(M)==0.8
Uq=1.4525; end
if 875<=(round (data_pos(1,7)/10))*10<=885 && str2num(M)==0.95
Uq=1.1365; end
if 875<=(round ((data_pos(1,7)/10))*10)<=885 && str2num(M)==1.1
Uq=0.8102; end

%computing uncertainty in pressure and balance derived counter
torque coefficient
for a=1:length(CCT_fin)
  U_CCT_fin(a,o)=
  sqrt((1/((Qpsi_posU(a)+Qpsi_negU(a))/2*pi/4*13.3^3)*27.80799)^2 +
  (TU_pos(a)/(Qpsi_posU(a)^2*pi/4*13.3^3)*Uq/144+TU_neg(a)/(Qpsi_negU(a)^2*pi/4*13.3^3)*Uq/144)^2);
end

U_CCT_roll(a,o)=sqrt((1/((Qpsi_posU(a)+Qpsi_negU(a))/2*pi/4*13.3^3)*
13.2827)^2 +
(T_rollUpos(a)/(Qpsi_posU(a)^2*pi/4*13.3^3)*Uq/144+T_rollUneg(a)/(Qpsi_negU(a)^2*pi/4*13.3^3)*Uq/144)^2);
end

%setting up next run
if str2num(M)==0.8
  if o==1
    p=num2str(str2num(file1(17:19))+50);
    q=num2str(str2num(file1(5:7))*2);
  elseif o==2
    p=num2str(str2num(file1(17:19))+25);
    q=num2str(str2num(file1(5:7))*2);
  end
  file1(17:19)=p;
  file1(5:7)=q;
end

if str2num(M)==0.95
  if o==1
    p=num2str(str2num(file1(18:20))+30);
    q=num2str(str2num(file1(5:7))*2);
  elseif o==2
    p=num2str(str2num(file1(18:20))+45);
    q=num2str(str2num(file1(5:7))*2);
  end
  file1(18:20)=p;
  file1(5:7)=q;
end

if str2num(M)==1.1
  if o==1
    p=num2str(str2num(file1(18:20))+10);
    q=num2str(str2num(file1(5:7))*2);
  elseif o==2
    p=num2str(str2num(file1(18:20))+65);
    q=num2str(str2num(file1(5:7))*2);
  end
  file1(18:20)=p;
  file1(5:7)=q;
%plots balance and pressure derived counter torque coefficient vs. J
and saves it
...in the directory CTplots
hold on
%determining final data point for each set of data
r=find(CCT_roll(:,1)==0,1)-1;
if isempty(r) r=size(CCT_roll,1); end
s=find(CCT_roll(:,2)==0,1)-1;
if isempty(s) s=size(CCT_roll,1); end
if str2num(config)==2
    t=find(CCT_roll(:,3)==0,1)-1;
    if isempty(t) t=size(CCT_roll,1); end
end
%increment for error bar
b=1:40:r;
c=1:40:s;
if str2num(config)==2
d=1:40:t;
end
%plotting coefficients and error bars
plot(U(1:r,1),CCT_fin(1:r,1), 'r', 'LineWidth', 1);%Q220 pressure
plot(U(1:r,1),CCT_roll(1:r,1), 'b', 'LineWidth', 1);%Q220 balance
plot(U(1:s,2),CCT_fin(1:s,2), 'r--', 'LineWidth', 1);%Q440 pressure
plot(U(1:s,2),CCT_roll(1:s,2), 'b--', 'LineWidth', 1);%Q440 balance
if str2num(config)==2%Q880
    plot(U(1:t,3),CCT_fin(1:t,3), 'r:', 'LineWidth', 1);
    plot(U(1:t,3),CCT_roll(1:t,3), 'b:', 'LineWidth', 1);
    errorbar(U(d,3),CCT_fin(d,3),U_CCT_fin(d,3),U_CCT_fin(d,3),'r.);
    errorbar(U(d,3),CCT_roll(d,3),U_CCT_roll(d,3),U_CCT_roll(d,3),'b.);
end
%Q220 & Q440 error bars
errorbar(U(b,1),CCT_fin(b,1),U_CCT_fin(b,1),U_CCT_fin(b,1),'r.);
errorbar(U(b,1),CCT_roll(b,1),U_CCT_roll(b,1),U_CCT_roll(b,1),'b.');</p>
errorbar(U(c,2),CCT_fin(c,2),U_CCT_fin(c,2),U_CCT_fin(c,2),'r.');</p>
errorbar(U(c,2),CCT_roll(c,2),U_CCT_roll(c,2),U_CCT_roll(c,2),'b.');</p>
if str2num(config)==2
    legend (["Q220 Pressure C_C_T'", ['Q220 Balance C_C_T'], ['Q440 Pressure C_C_T'], ['Q440 Balance C_C_T'], 'Location', 'Best')
else legend (["Q220 Pressure C_C_T'", ['Q220 Balance C_C_T'], ['Q440 Pressure C_C_T'], ['Q440 Balance C_C_T'], ['Q880 Pressure C_C_T'], ['Q880 Balance C_C_T'], 'Location', 'Best')
end
text(0.025, 0.985,['\alpha = ' AOA ''], 'sc');%displays angle of attack
text(0.025, 0.955,['Roll angle= ' rollangle ''], 'sc'); % displays roll angle
grid on
title(['Counter Torque Coefficient at Mach' M ', Configuration' config '']);
xlabel('Nozzle Dynamic Pressure Ratio, J');
ylabel('Counter Torque Coefficient (C_C_T)');
saveas(gcf, ['CTplots/Mach' M ', Config' config ' AOA ' rollangle 'roll.jpg'], 'jpg');
hold off
close

FAodd_posCT.m

function T_FAodd_pos = FAodd_pos

% global variable declaration
global dir run_pos J_pos FA_p_pos tap_coords fin_area AOA_pos
       AOA_neg M data_pos tare_pressures

% data filter
p = FA_p_pos(:, [1:2:47]); % maintains pressure data for Fin A, odd pressure taps, positive AOA
x = tap_coords(:, 1); % maintains x coordinates for Fin A, odd pressure taps
y = tap_coords(:, 2); % maintains y coordinates for Fin A, odd pressure taps
z = tap_coords(:, 3); % maintains z coordinates for Fin A, odd pressure taps

% reading appropriate tare pressure from file
if str2num(dir(10:length(dir))) == 8 && str2num(dir(5:7)) == 220
  tare_pressure = tare_pressures(5);
elseif str2num(dir(10:length(dir))) == 95 && str2num(dir(5:7)) == 220
  tare_pressure = tare_pressures(13);
elseif str2num(dir(10:length(dir))) == 11 && str2num(dir(5:7)) == 220
  tare_pressure = tare_pressures(21);
elseif str2num(dir(10:length(dir))) == 8 && str2num(dir(5:7)) == 440
  tare_pressure = tare_pressures(29);
elseif str2num(dir(10:length(dir))) == 95 && str2num(dir(5:7)) == 440
  tare_pressure = tare_pressures(37);
elseif str2num(dir(10:length(dir))) == 11 && str2num(dir(5:7)) == 440
  tare_pressure = tare_pressures(45);
elseif str2num(dir(10:length(dir))) == 8 && str2num(dir(5:7)) == 880
  tare_pressure = tare_pressures(53);
elseif str2num(dir(10:length(dir))) == 95 && str2num(dir(5:7)) == 880
  tare_pressure = tare_pressures(61);
elseif str2num(dir(10:length(dir))) == 11 && str2num(dir(5:7)) == 880
  tare_pressure = tare_pressures(69);
else disp('Error identifying tare pressure')
end

%function calls
[vertices, area_vectors, r, a_surf] = vert_and_area(x, y, z);%calls
function to determine triangle vertices,
...area vectors, triangle centroids, and area on Fin A, Odd within
tapped region
p_triangle = triangle_pressures(p, vertices);%calls function to
determine mean pressures for each triangle
T_tot = moment(p_triangle, area_vectors, r);%calls function to
determine torque for the entire surface, Fin A: Odd

%calculations
T_tot_x = -1.*T_tot(:,1);

%plots roll torque for Fin A, Odd, positive AOA versus J and saves
it in directory with
%DAT file
plot (J_pos, T_tot_x, 'LineWidth', 1)
grid on
ylabel('Counter Torque [in-lb_f]
xlabel('Nozzle Dynamic Pressure Ratio, J');
title(['Run #' run_pos ' M ' M ', AOA=' AOA_pos ' degrees: Fin A,
Odd Taps']);
saveas(gcf,[dir '/RUN0' run_pos 'finAodd'], 'jpg');
close

%scales torque to approximate the torque over the entire Fin A, Odd
T_FAodd_pos = T_tot_x*(fin_area/a_surf) - tare_pressure;

finpress_plot.m

%finpress_plot.m
%plots fin pressure coefficient distribution

clear all
clc
%reading input file
dir = input('Enter directory name (ex. C2_Q440_M8): ','s');%the
directory with DAT file
run = input('Enter the run number (ex. 0426): ','s');%the DAT file
BIG=load([dir '/RUN0' run '.dat']);%loads entire DAT file
mkdir(dir, 'finplots')

%tap coordinates
Xodd=[0.06 1.28 1.18 1.24 2.73 2.84 3.08 3.45 3.94 5.14 5.56 6.11 6.78 7.58 7.54 8.28 9.17 10.2 11.39 9.96 11.07 13.11 13.04 15.19];
Xeven=[0.34 1.52 1.46 1.55 2.97 3.11 3.39 3.78 4.3 5.38 5.83 6.41 7.12 7.95 7.78 8.56 9.48 10.55 11.79 10.2 11.36 13.49 13.39 15.61];
Yeven=[7.44 6.27 7.4 8.53 6.2 7.35 8.45 9.64 10.81 6.1 7.27 8.45 9.64 10.85 5.99 7.19 8.4 9.63 10.89 5.89 7.1 9.62 8.33 10.31];

% reading tare values
tares=load([dir '/RUN00' num2str(str2num(run)-2) '.dat']);
tare_odd=tares(2,135:2:181);
tare_even=tares(2,136:2:182);

% plotting fin B
Aodd=BIG(:,135:2:181);
Aeven=BIG(:,136:2:182);

% setting J values to be plotted
J_1=8;
J_2=16;
J_3=36;

if BIG(end,427) < J_2
  k=1;
elseif BIG(end,427) < J_3
  k=2;
else
  k=3;
end

for j=1:k
  if j==1
    J=J_1;
  elseif j==2
    J=J_2;
  elseif j==3
    J=J_3;
  end

  % Q uncertainties from AEDC
  if 215<=(round (BIG(1,7)/10))*10<=225 && str2num(dir(10:end))==8
    Uq=1.2786; end
  elseif 215<=(round (BIG(1,7)/10))*10<=225 && str2num(dir(10:end))==95
    Uq=1.0391; end
  elseif 215<=(round (BIG(1,7)/10))*10<=225 && str2num(dir(10:end))==11
    Uq=0.7639; end
  elseif 435<=(round (BIG(1,7)/10))*10<=445 && str2num(dir(10:end))==8
    Uq=1.3004; end
  elseif 435<=(round (BIG(1,7)/10))*10<=445 && str2num(dir(10:end))==95
    Uq=1.0541; end
  elseif 435<=(round (BIG(1,7)/10))*10<=445 && str2num(dir(10:end))==11
    Uq=0.7708; end
  elseif 875<=(round (BIG(1,7)/10))*10<=885 && str2num(dir(10:end))==8
    Uq=1.4525; end
if 875<=(round (BIG(1,7)/10))*10<=885 &&
str2num(dir(10:end))===95 Uq=1.1365; end
if 875<=(round (BIG(1,7)/10))*10<=885 &&
str2num(dir(10:end))===11 Uq=0.8102; end

%plotting fin at prescribed J
i=find (floor(BIG(:,427))==J);
Zodd=(Aodd(i(1),:)-tare_odd)/BIG(i(1),7); %normalized odd
presures
Zeven=(Aeven(i(1),:)-tare_even)/BIG(i(1),7); %normalized even

%generates uniform grid over entire tapped surface
[zi,zi] = meshgrid(0:.1:16,5:.1:11); %generates uniform grid over entire tapped surface
zi_odd=griddata(Xodd,Yodd,Zodd,xi,yi,'linear'); %interpolates X-Y tap coordinates and pressures onto uniform grid
zi_even=griddata(Xeven,Yeven,Zeven,xi,yi,'linear'); %interpolates X-Y tap coordinates and pressures onto uniform grid

deltapressure=(zi_odd-zi_even);

for p=1:size(deltapressure,1) %computes uncertainty for each point in tapped region
    for q=1:size(deltapressure,2)
        uncertainty(p,q)=sqrt((1/BIG(i(1),12)*0.0375)^2+(deltapressure(p,q)/BIG(i(1),12)^2*Uq/144)^2);
    end
end
Umax=max(max(uncertainty,[],2)); %maximum uncertainty in tapped region
peakmax=min(min(deltapressure,[],2)); %determines maximum pressure coefficient
peakmin=(1-1/exp(1))*min(min(deltapressure,[],2)); %identifies region with > 67% of maximum pressure coefficient

n=find(deltapressure<peakmin);
xin=x(n); %x coordinates of peak region
yin=yi(n); %y coordinates of peak region
tri=delaunay(xin,yin); %triangulates peak region
tri_xcoord= xin(tri); %x coordinates of triangles
tri_ycoord= yin(tri); %y coordinates of triangles
area= tri2Darea(tri_xcoord, tri_ycoord); %determines area of triangles

ik=area/(sum(area)/length(area));
b = find(ik > 1.35 | ik<0.35); %finds triangles with area > or < 35% of mean area
area(b)=[]; %deletes triangles
area_peaks=area;

h=find (isnan(deltapressure)==0); %determines total tapped area
xik=x(h); %tapped area x coordinates
yik=yi(h); %tapped area y coordinates
tri_tot=delaunay(xik,yik); %triangulates tapped area
tri_xcoord_tot = xik(tri_tot); % x coordinates of triangles
tri_ycoord_tot = yik(tri_tot); % y coordinates of triangles
area = tri2Darea(tri_xcoord_tot, tri_ycoord_tot); % determines area
of triangles
ik = area / (sum(area) / length(area));

b = find(ik > 1.35 | ik < 0.35); % finds triangles > or < 35% of mean
area(b) = []; % deletes triangles

percent_peak = sum(area_peaks) / sum(area) * 100; % percent peak region
peak_mean = mean(deltapressure(n)); % mean pressure coefficient in
peak region

% generating slices A through E
x_1 = [0.34:0.001:11.8];
y_1 = 7.44 * ones(1, length(x_1));
z_1(:, :, j) = griddata(xi, yi, deltapressure, x_1, y_1, 'linear');
x_2 = [1.52:0.001:4.3];
y_2 = (4.54/2.78) * x_2 + 3.7877;
z_2(:, :, j) = griddata(xi, yi, deltapressure, x_2, y_2, 'linear');
x_3 = (0.5987:0.001:4.8);
y_3 = (3.334/3.914) * x_3 + 6.7112928;
z_3(:, :, j) = griddata(xi, yi, deltapressure, x_3, y_3, 'linear');
x_4 = (0.8281:0.001:5.3);
y_4 = (3.334/3.914) * x_4 + 6.2953858;
z_4(:, :, j) = griddata(xi, yi, deltapressure, x_4, y_4, 'linear');
x_5 = (1.1061:0.001:5.8);
y_5 = (3.334/3.914) * x_5 + 5.8594788;
z_5(:, :, j) = griddata(xi, yi, deltapressure, x_5, y_5, 'linear');

% shaded view of tapped region with slices and taps visible
hold on
plot(x_1, y_1, 'linewidth', 2)
plot(x_2, y_2, 'k', 'linewidth', 2)
plot(x_3, y_3, 'k--', 'linewidth', 2)
plot(x_4, y_4, 'm', 'linewidth', 2)
plot(x_5, y_5, 'r', 'linewidth', 2)
plot(Xodd, Yodd, 'xr')
plot(Xeven, Yeven, 'o')
text(0.025, 0.985, ['uncertainty = ' num2str(Umax) ' '], 'sc');
legend ('Slice A', 'Slice B', 'Slice C', 'Slice D', 'Slice E', 'odd
taps', 'even taps', 'location', 'SouthEast')

ylabel('Y [in]

pcolor(xi, yi, deltapressure) % creates shaded pressure plot
axis equal
axis off

colorbar
shading interp
title(['Run ' run ': DELTA JET ON/OFF FIN PRESSURE COEFFICIENT,
FIN B, AOA=' num2str(round (BIG(1,13))) ', Roll=' num2str(round

82
(BIG(1,14))) ', Qinf=' num2str(round (BIG(1,7))) ', Mach=' num2str(round((BIG(1,4))*100)/100) ', J=' num2str(J) ']);
saveas(gcf,[dir '/finplots' '/RUN0' run ' M ' num2str(round((BIG(1,4))*100)/100) ', J ' num2str(J) 'FinBplot.jpg'],'jpg');
end

end

%plotting pressure vs. distance along slice A
hold on
plot(sqrt((x_1-0.34).^2+(y_1-7.44).^2),z_1(:,:,1),'b')
if k~=1 plot(sqrt((x_1-0.34).^2+(y_1-7.44).^2),z_1(:,:,2),'k'); end
if k==3 plot(sqrt((x_1-0.34).^2+(y_1-7.44).^2),z_1(:,:,3),'m'); end
if k==3
legend(['Fin B: J=' num2str(J_1) '], ['Fin B: J=' num2str(J_2) ']', ['Fin B: J=' num2str(J_3) '], 'Location', 'Best')
elseif k==2 legend (['Fin B, J=' num2str(J_1) '], ['Fin B, J=' num2str(J_2) '], 'Location', 'Best')
else legend(['Fin B: J=' num2str(J_1) '], 'Location','Best')
end
xlabel('Distance along slice (west to east) [in]')
ylabel('Delta Pressure Coefficient')
hold off

%plotting pressure vs. distance along slice B
hold on
plot(sqrt((x_2-1.52).^2+(y_2-6.27).^2),z_2(:,:,1),'b')
if k~=1 plot(sqrt((x_2-1.52).^2+(y_2-6.27).^2),z_2(:,:,2),'k'); end
if k==3 plot(sqrt((x_2-1.52).^2+(y_2-6.27).^2),z_2(:,:,3),'m'); end
if k==3
legend(['Fin B: J=' num2str(J_1) '], ['Fin B: J=' num2str(J_2) ']', ['Fin B: J=' num2str(J_3) '], 'Location', 'Best')
elseif k==2 legend (['Fin B, J=' num2str(J_1) '], ['Fin B, J=' num2str(J_2) '], 'Location', 'Best')
else legend(['Fin B: J=' num2str(J_1) '], 'Location','Best')
end
xlabel('Distance along slice (west to east) [in]')
ylabel('Delta Pressure Coefficient')
hold off

83
title ('Run ' run ': Slice B, AOA=' num2str(round (BIG(1,13))) ', Roll=' num2str(round (BIG(1,14))) ', Mach=' num2str(round((BIG(1,4))*100)/100) '); saveas(gcf,[dir '/finplots' '/RUN0' run ' M ' num2str(round((BIG(1,4))*100)/100) ', sliceBplot'],'jpg'); close

%plotting pressure vs. distance along slice C
hold on
plot(sqrt((x_3-0.5987).^2+(y_3-7.221274).^2),z_3(:,:,1),'b')
if k~=1 plot(sqrt((x_3-0.5987).^2+(y_3-7.221274).^2),z_3(:,:,2),'k'); end
if k==3 plot(sqrt((x_3-0.5987).^2+(y_3-7.221274).^2),z_3(:,:,3),'m'); end
if k==3
legend(['Fin B: J=' num2str(J_1) ''], ['Fin B: J=' num2str(J_2) '], ['Fin B: J=' num2str(J_3) '], 'Location', 'Best')
elseif k==2 legend ('Fin B, J=' num2str(J_1) ', ' Location', 'Best')
else legend ('Fin B: J=' num2str(J_1) ', ' Location','Best')
end
xlabel('Distance along slice (west to east) [in]')
ylabel('Delta Pressure Coefficient')
hold off

%plotting pressure vs. distance along slice D
hold on
plot(sqrt((x_4-0.8281).^2+(y_4-7.000773).^2),z_4(:,:,1),'b')
if k~=1 plot(sqrt((x_4-0.8281).^2+(y_4-7.000773).^2),z_4(:,:,2),'k'); end
if k==3 plot(sqrt((x_4-0.8281).^2+(y_4-7.000773).^2),z_4(:,:,3),'m'); end
if k==3
legend(['Fin B: J=' num2str(J_1) '], ['Fin B: J=' num2str(J_2) '], ['Fin B: J=' num2str(J_3) '], 'Location', 'Best')
elseif k==2 legend ('Fin B, J=' num2str(J_1) ', ' Location', 'Best')
else legend ('Fin B: J=' num2str(J_1) ', ' Location','Best')
end
xlabel('Distance along slice (west to east) [in]')
ylabel('Delta Pressure Coefficient')
hold off
title ('Run ' run ': Slice D, AOA=' num2str(round (BIG(1,13))) ', Roll=' num2str(round (BIG(1,14))) ', Mach=' num2str(round((BIG(1,4))*100)/100) '']);
saveas(gcf,[dir '/finplots' '/RUN0' run ' M ' num2str(round((BIG(1,4))*100)/100) ', sliceDplot'],'jpg');
close

%plotting pressure vs. distance along slice E
hold on
plot(sqrt((x_5-1.1061).^2+(y_5-6.80167).^2),z_5(:,:,1),'b')
if k~=1 plot(sqrt((x_5-1.1061).^2+(y_5-6.80167).^2),z_5(:,:,2),'k'); end
if k==3 plot(sqrt((x_5-1.1061).^2+(y_5-6.80167).^2),z_5(:,:,3),'m'); end
if k==3
    legend(['Fin B: J=' num2str(J_1) ''], ['Fin B: J=' num2str(J_2) '], ['Fin B: J=' num2str(J_3) '], 'Location', 'Best')
elseif k==2 legend(['Fin B, J=' num2str(J_1) '], ['Fin B, J=' num2str(J_2) '], 'Location', 'Best')
else legend(['Fin B: J=' num2str(J_1) '], 'Location','Best')
end
xlabel('Distance along slice (west to east) [in]')
ylabel('Delta Pressure Coefficient')
hold off

title ('Run ' run ': Slice E, AOA=' num2str(round (BIG(1,13))) ', Roll=' num2str(round (BIG(1,14))) ', Mach=' num2str(round((BIG(1,4))*100)/100) '']);
saveas(gcf,[dir '/finplots' '/RUN0' run ' M ' num2str(round((BIG(1,4))*100)/100) ', sliceEplot'],'jpg');
close

VFI_torquesuncertainty.m

clear all
clc

%global variable declaration
global dir run J FA_p tap_coords fin_area pressure_precision
tap_coords = load(['fin_taps.dat']);%loads entire taps file that contains coordinates for the taps

%calculating area of whole fin
fin_dim = load('fin_shape.dat');%loads coordinates of fin edge points
fin_x = fin_dim(:,1);%maintains edge x-coordinates
fin_y = fin_dim(:,2);%maintains edge y-coordinates
fin_area = polyarea(fin_x,fin_y);%calculates area of whole fin

%plotting pressure vs. distance along slice E
hold on
plot(sqrt((x_5-1.1061).^2+(y_5-6.80167).^2),z_5(:,:,1),'b')
if k~=1 plot(sqrt((x_5-1.1061).^2+(y_5-6.80167).^2),z_5(:,:,2),'k'); end
if k==3 plot(sqrt((x_5-1.1061).^2+(y_5-6.80167).^2),z_5(:,:,3),'m'); end
if k==3
    legend(['Fin B: J=' num2str(J_1) '], ['Fin B: J=' num2str(J_2) '], ['Fin B: J=' num2str(J_3) '], 'Location', 'Best')
elseif k==2 legend(['Fin B, J=' num2str(J_1) '], ['Fin B, J=' num2str(J_2) '], 'Location', 'Best')
else legend(['Fin B: J=' num2str(J_1) '], 'Location','Best')
end
xlabel('Distance along slice (west to east) [in]')
ylabel('Delta Pressure Coefficient')
hold off

title ('Run ' run ': Slice E, AOA=' num2str(round (BIG(1,13))) ', Roll=' num2str(round (BIG(1,14))) ', Mach=' num2str(round((BIG(1,4))*100)/100) ']);
saveas(gcf,[dir '/finplots' '/RUN0' run ' M ' num2str(round((BIG(1,4))*100)/100) ', sliceEplot'],'jpg');
close
%data filter
for i=1:48
    FA_p = zeros(1,48);
    FA_p(1,i) = 1;
end

%function calls
T_FAodd(i) = FAoddUC;%calls function FAodd stores fin pressure, roll torques on Fin A, Odd
T_FAeven(i) = FAevenUC;%calls function FAeven stores fin pressure, roll torques on Fin A, Even
T_FA(i) = T_FAodd(i) + T_FAeven(i);
end

U=sqrt(2*sum((T_FA.*0.0375).^2));
U_0AOA=2*U
U_AOA=sqrt(2)*U

FAoddUC.m

function T_FAodd = FAoddUC
%global variable declaration
global dir run J FA_p tap_coords fin_area pressure_precision
%
data filter
p = FA_p(:,[1:2:47]);%maintains pressure data for Fin B, odd pressure taps
x = tap_coords(:,1);%maintains x coordinates for Fin A, odd pressure taps
y = tap_coords(:,2);%maintains y coordinates for Fin A, odd pressure taps
z = tap_coords(:,3);%maintains z coordinates for Fin A, odd pressure taps
%
function calls
[vertices, area_vectors, r, a_surf] = vert_and_area(x, y, z);%calls function to determine triangle vertices, ...
...area vectors, triangle centroids, and area on Fin A, Odd within tapped region
p_triangle = triangle_pressures(p, vertices);%calls function to determine mean pressures for each triangle
T_tot = moment(p_triangle, area_vectors, r);%calls function to determine torque for the entire surface, Fin A:Odd
%
calculations
T_tot_x = -1.*T_tot(:,1);
%
scales torque to approximate the torque over the entire Fin A, Odd
T_FAodd = T_tot_x*(fin_area/a_surf);
**Distribution**

1  MS 0825  B. Hassan, 1515  
1  MS 0825  C. W. Peterson, 1500  
1  MS 0825  W. P. Wolfe, 1515  
1  MS 0834  J. E. Johannes, 1512  
1  MS 0834  S. J. Beresh, 1512  
2  MS 0899  Technical Library, 4536  
1  MS 9018  Central Technical Files, 8945-1  

1  Leyen S. Chang  
   Department of Mechanical Engineering  
   Stanford University  
   Stanford, California 94305-3030