2019

Employing 2-D CFD & LRB Model Around Trees to Improve VAWT Placement

David Chukwuebuka Bassey

Minnesota State University, Mankato

Follow this and additional works at: https://cornerstone.lib.mnsu.edu/etds

Part of the Civil and Environmental Engineering Commons, and the Mechanical Engineering Commons

Recommended Citation
https://cornerstone.lib.mnsu.edu/etds/949
Employing 2-D CFD & LRB Model Around Trees to Improve VAWT Placement

By

David Bassey

An Alternate Plan Paper submitted In Partial Fulfilment of the

Requirements for the Degree of

Master of Science in

In

Mechanical Engineering

Minnesota State University, Mankato

Mankato, Minnesota

August 2019
Employing 2-D CFD & LRB Model Around Trees to Improve VAWT Placement

David Bassey

This alternate plan paper has been examined and approved by the following committee members:

____________________________________
Dr. Patrick Tebbe, P.E.,
Advisor

____________________________________
Dr. Namyong Lee,
Committee Member

____________________________________
Dr. Nazli Wodzinski,
Committee Member

____________________________________
Dr. Matthew Simones,
Committee Member,
Acknowledgements

I would not be where I am today if not for my teachers. I am profoundly grateful to Dr. Patrick Tebbe, for guiding me throughout from my undergraduate to my graduate program. His knowledge, experience, technical advice and probing questions are greatly instrumental to the success of this thesis.

Special thanks to my committee members, Dr. Nazli Wodzinski, Dr. Namyong Lee and Dr. Matthew Simones for providing me with the continued guidance, support, contributions and encouragement throughout my program in the MECE Department of MNSU. Thanks to the Xcel Energy Research Development Fund (RDF) for sponsoring this work. This research would not be possible without this funding.

Special thanks to my Kevin Schull for providing critique and engaging discussion during this research; and to Brian Graybill of Nortek Air Solutions, for providing vital industry insights and advice into making this project a success.

I am deeply grateful to have the best supporters in the world, my parents; Bassey and Dorathy Umoetok; for providing immense love and support throughout the course of my graduate studies. The support of my siblings, Joshua, Benjamin, Uboho, Sharon and Queen is also highly appreciated including helping to proof-read my work.

My appreciation also goes to my friends and colleagues, Ramiro, Fahim, Seun, Yacouba, Jeremiah, Devin, Heidi, Joshua, Jon and others too numerous to mention, for valuable interactions, warm friendship and the great times we have shared.
Dedication

This thesis is dedicated to God, the giver of all!
Abstract

In the placement of vertical axis wind turbines, trees are a constant presence in the vicinity. They are found to grow at different height and shape configurations. And in areas such as the Minnesota State University, Mankato (MNSU) campus, they serve as blockage to airflow; limiting the efficiency of installed turbines. This work sets the precedent for the validation of vegetative numerical models created for the Xcel Energy Research Development Fund (RDF) project.

Using two-dimensional (2-D) numerical simulations of porous cylinders placed in a rectangular medium of air, insight into the flow profile and distribution in the leeward side of the cylinder is gained. Mesh and turbulence intensity dependency test show less than 5% change in flow properties. Domain size and blockage ratio remained an important factor to consider during this simulation.

In this work, a numerical investigation was performed using design of experiment (DOE) principles, atmospheric flow and VAWT performance criteria to vary the incoming velocity and porosity to observe the effect on the wake velocity characteristics. It was observed that for inertial resistance coefficients greater than 200 m$^{-1}$, the porous cylinder behaves as an impermeable object limiting the flow of air through the porous zone.

Following the qualitative descriptions of the porous flow regime, comparisons are made with 2-D results from existing literature. For velocity, there is strong agreement with trend of the non-dimensional velocity flow along the centerline. However, the results from the pressure distribution and the turbulent properties are underestimated when compared due to the presence of enstrophy in 3-D flow. Using actuator disk theory and leaky Rankine bodies, the resulting velocity field is used to obtain an estimate of source and sink strength ratios for use in flow field estimation.
Contents

Acknowledgements .................................................................................................................. iv

Dedication ................................................................................................................................. v

Abstract ..................................................................................................................................... vi

List of Figures .......................................................................................................................... xii

List of Tables ............................................................................................................................ xxiv

Nomenclature ........................................................................................................................... xxv

Chapter 1 - Introduction ......................................................................................................... 1

Chapter 2 - Background ........................................................................................................... 5

Flow through non-porous media ............................................................................................... 5

Governing laws for non-porous (solid objects) flow ............................................................... 7

Review of Potential Flow Theory ........................................................................................... 8

Flow through porous media ....................................................................................................... 9

Principles governing porous flow ............................................................................................ 9

Forchheimer extension of Darcy’s flow regime ...................................................................... 10

Overview of CFD Modeling ..................................................................................................... 11

Discretization ........................................................................................................................... 12

Meshing ..................................................................................................................................... 12

Turbulence Modeling ............................................................................................................... 15
Velocity variation............................................................................................................................................. 74
Comparison with literature ............................................................................................................................ 82
Discussion...................................................................................................................................................... 90
Utilization of LRB to estimate Porous flow velocity field ............................................................................. 90

Chapter 6 - Conclusion .................................................................................................................................. 94
Further work/Recommendations ..................................................................................................................... 96
References..................................................................................................................................................... 97

Appendix A.................................................................................................................................................. 109
  Reduction of Navier-Stokes equation ........................................................................................................ 109
  Numerical setup ........................................................................................................................................ 113
    Numerical Convergence plots .................................................................................................................. 114

Appendix B – Simulation Results of velocity and resistance coefficient variation .................................... 116
Appendix C – MATLAB code (potentialflow.m) ......................................................................................... 159
Appendix D – MATLAB code (grabit.m) ..................................................................................................... 163
Appendix E - Derivation of source/sink strength ratio estimate .................................................................... 227
  Relationship between Actuator Disk Theory and Porous Flow .............................................................. 228
Appendix F – Tree classification in Minnesota .......................................................................................... 235
Appendix G - Experimental Validation method .......................................................................................... 240
  Calibration Procedure ............................................................................................................................... 240
Experimental results........................................................................................................... 241

Calibration Results............................................................................................................. 241

Measurements of velocity around tree............................................................................ 245
List of Figures

Figure 1: Flow past a cylinder a) laminar separation b) turbulent separation [1], Ch. 7 ............. 6

Figure 2: Drag coefficient at Re ≥ 10^4. [1] ................................................................................................. 9

Figure 3: Flow regimes in a porous media. 1 = Darcy’s law , 2 = Weak inertia, 3 = Forchheimer (strong inertia), 4 = transition from Forchheimer to turbulence, and 5 = turbulence. [29] .......... 11

Figure 4: Different VAWT configurations [49]............................................................................................ 21

Figure 5: Model of a basic tree [42]............................................................................................................. 22

Figure 6: Streantube expansion, velocity decrease and pressure jump as wind passes through the actuator disk........................................................................................................................................ 27

Figure 7: Schematic description of areas used in domain............................................................................... 36

Figure 8: Full view of the flow domain and geometry with face splits. ......................................................... 40

Figure 9: Close up of the cylinder geometry. ................................................................................................. 40

Figure 10: Model tree from ANSYS Geometry - Design Modeler ................................................................. 41

Figure 11: Placement of dimensions for geometry......................................................................................... 42

Figure 12: Full view of structured mesh ....................................................................................................... 43

Figure 13: Closeup of the mesh refined zone surrounding the porous zone ................................................. 43

Figure 14: Extreme closeup of the mesh porous zone area. ........................................................................... 44

Figure 15: Meshing model tree from ANSYS Meshing ................................................................................. 45

Figure 16: Mesh statistics of final mesh from ANSYS Meshing ................................................................. 46

Figure 17: Full view of the final mesh showing the skewness ranges......................................................... 47
Figure 18: Close-up of the final mesh showing areas of high skewness near the porous cylinder
-------------------------------------------------------------------------------------------------- 47

Figure 19: Full view of the final mesh showing the orthogonal quality ranges. ......................... 48

Figure 20: Close-up of the final mesh showing areas of high orthogonal quality near the porous
 cylinder .......................................................................................................................................... 48

Figure 21: Full view of the final mesh showing the aspect ratio ranges........................................ 49

Figure 22: Close-up of the final mesh showing areas of high aspect ratio near the porous cylinder
.................................................................................................................................................. 49

Figure 23: Mesh dependency for centerline velocity magnitude.................................................... 56

Figure 24: Contour plot of streamwise horizontal velocity around solid cylinder. .................... 57

Figure 25: Contours of streamwise horizontal velocity for porous zone at $c_i = 15$ m$^{-1}$ ......... 58

Figure 26: Comparison of wall and symmetry boundary condition for three resistance coefficient
 values. ................................................................................................................................................. 59

Figure 27: Variation of velocity along flow direction (the $x$-direction) with different inertial
 resistance coefficients. .................................................................................................................. 61

Figure 28: Variation of velocity along the flow direction (the $x$-direction) at height of 0.5h with
different inertial resistance coefficients. [67] ................................................................................. 62

Figure 29: Effect of resistance coefficient on the wake velocity at $x/D = 5.11, 10, 15.02, 20.....$ 63

Figure 30: Velocity streamlines for porous flow at $c_i = 15$ m$^{-1}$ ................................................. 64

Figure 31: Velocity vector plot for porous flow at $c_i = 15$ m$^{-1}$ .................................................. 64
Figure 32: Closeup view of velocity vectors for porous flow at $c_i = 15 \text{ m}^{-1}$ .............................................. 65
Figure 33: Velocity streamline plot for porous flow at $c_i = 75 \text{ m}^{-1}$ ......................................................... 65
Figure 34: Velocity vector plot for porous flow at $c_i = 75 \text{ m}^{-1}$ ................................................................. 66
Figure 35: Closeup view of velocity vectors for porous flow at $c_i = 75 \text{ m}^{-1}$ ..................................................... 66
Figure 36: Velocity streamlines for porous flow at $c_i = 150 \text{ m}^{-1}$ ................................................................. 67
Figure 37: Velocity vector plot for porous flow at $c_i = 150 \text{ m}^{-1}$ ................................................................. 67
Figure 38: Closeup view of velocity vectors for porous flow at $c_i = 150 \text{ m}^{-1}$ ..................................................... 68
Figure 39: Velocity streamlines for porous flow at $c_i = 300 \text{ m}^{-1}$ ................................................................. 69
Figure 40: Velocity vector plot for porous flow at $c_i = 300 \text{ m}^{-1}$ ................................................................. 69
Figure 41: Closeup view of velocity vectors for porous flow at $c_i = 300 \text{ m}^{-1}$ ..................................................... 70
Figure 42: Velocity streamline plot for porous flow at $c_i = 750 \text{ m}^{-1}$ ............................................................... 70
Figure 43: Velocity vector plot for porous flow at $c_i = 750 \text{ m}^{-1}$ ................................................................. 71
Figure 44: Closeup view of velocity vectors for porous flow at $c_i = 750 \text{ m}^{-1}$ ..................................................... 71
Figure 45: Velocity streamline plot for porous flow at $c_i = 1500 \text{ m}^{-1}$ ............................................................... 72
Figure 46: Velocity vector plot for porous flow at $c_i = 1500 \text{ m}^{-1}$ ................................................................. 72
Figure 47: Closeup view of velocity vectors for porous flow at $c_i = 1500 \text{ m}^{-1}$ ..................................................... 73
Figure 48: Velocity streamline plot for flow around a solid cylinder ................................................................. 73
Figure 49: Velocity vector plot for flow around a solid cylinder ................................................................. 74
Figure 50: Closeup view of velocity vectors for flow around a solid cylinder ......................................................... 74
Figure 51: Variation of velocity along flow direction at all initial velocities for $c_i=15 \text{ m}^{-1}$ ........ 75
Figure 52: Variation of velocity along flow direction at all initial velocities for $c_i=75 \text{ m}^{-1}$ ........ 75
Figure 53: Variation of velocity along flow direction at all initial velocities for $c_i=150 \text{ m}^{-1}$ ...... 76
Figure 54: Variation of velocity along flow direction at all initial velocities for $c_i=300 \text{ m}^{-1}$ ...... 77
Figure 55: Variation of velocity along flow direction at all initial velocities for $c_i=750 \text{ m}^{-1}$ ...... 77
Figure 56: Variation of velocity along flow direction at all initial velocities for $c_i=1500 \text{ m}^{-1}$ .... 78
Figure 57: Pressure coefficient comparison along centerline at velocity = 2.5 m/s .................... 79
Figure 58: Pressure coefficient comparison along centerline at velocity = 5 m/s ....................... 79
Figure 59: Pressure coefficient comparison along centerline at velocity = 7.5 m/s ................. 80
Figure 60: Pressure coefficient comparison along centerline at velocity = 10 m/s ............... 80
Figure 61: Pressure coefficient comparison along centerline at velocity = 12.5 m/s .......... 81
Figure 62: Pressure coefficient comparison along centerline at velocity = 15 m/s .......... 81
Figure 63: Comparison of $x$-velocity profiles along the wake of a tree from simulation (left) with figure 3 [20] (3-D flow around a rectangular cylinder, right).......................................................... 82
Figure 64: TKE profiles comparison with figure 4 [20]........................................................... 83
Figure 65: TI profiles along the wake of canopy at $c_i = 15 \text{ m}^{-1}$ and 0.56 m/s for $D = 0.01 \text{ m}$ .... 84
Figure 66: Vertical TKE profiles at all $c_i$ levels ($a = 15 \text{ m}^{-1}, b = 75 \text{ m}^{-1}, c = 150 \text{ m}^{-1}, d = 300 \text{ m}^{-1}, e = 750 \text{ m}^{-1}, f = 1500 \text{ m}^{-1}$).................................................................................................................. 85
Figure 67: Velocity comparison at 15 m$^{-1}$ with [67].......................................................... 87
Figure 70: Velocity comparison at 75 m$^{-1}$ with [67]......................................................... 88
Figure 69: Velocity comparison at 150 m$^{-1}$ with [67]................................................................. 88

Figure 70: Velocity comparison at 300 m$^{-1}$ with [67]........................................................................ 89

Figure 71: Velocity comparison at 1500 m$^{-1}$ with [67]...................................................................... 89

Figure 72: Simulated $x$-velocity ($c_i = 15$ m$^{-1}$) versus centerline velocity from potential flow .... 91

Figure 73: Streamline plot of stream function solution using potential flow ........................................ 91

Figure 74: Vector field of potential flow simulation .................................................................................... 92

Figure 75: Sink spacing, $S_s$ versus R-squared values at $c_i = 15$ m$^{-1}$.................................................. 93

Figure 85: Point cloud and plane reference for exporting grid data for potential flow comparison ....... 113

Figure 86: Area-weighted average of $x$-velocity for $c_i = 15$ m$^{-1}$ and incoming velocity = 0.56 m/s ............................................................................................................ 114

Figure 87: Mass flowrate imbalance on inlet vs outlet for $c_i = 15$ m$^{-1}$ and incoming velocity = 0.56 m/s ............................................................................................................ 114

Figure 88: Residual convergence for $c_i = 15$ m$^{-1}$ and incoming velocity = 0.56 m/s..................... 115

Figure 80: Contour plot of $x$-velocity along a solid cylinder at 2.5 m/s .............................................. 117

Figure 81: Contour plot of pressure coefficient along a solid cylinder at 2.5 m/s............................... 117

Figure 82: Contour plot of TKE along a solid cylinder at 2.5 m/s ...................................................... 117

Figure 83: Contour plot of $x$-velocity of a porous cylinder (15 m$^{-1}$) at 2.5 m/s.............................. 118

Figure 84: Contour plot of pressure coefficient of a porous cylinder (15 m$^{-1}$) at 2.5 m/s .......... 118

Figure 85: Contour plot of TKE of a porous cylinder (15 m$^{-1}$) at 2.5 m/s................................. 118
Figure 86: Contour plot of $x$-velocity of a porous cylinder (75 m$^{-1}$) at 2.5 m/s................. 119
Figure 87: Contour plot of pressure coefficient of a porous cylinder (75 m$^{-1}$) at 2.5 m/s .... 119
Figure 88: Contour plot of TKE of a porous cylinder (75 m$^{-1}$) at 2.5 m/s......................... 119
Figure 89: Contour plot of $x$-velocity of a porous cylinder (150 m$^{-1}$) at 2.5 m/s ............ 120
Figure 90: Contour plot of pressure coefficient of a porous cylinder (150 m$^{-1}$) at 2.5 m/s .... 120
Figure 91: Contour plot of TKE of a porous cylinder (150 m$^{-1}$) at 2.5 m/s....................... 120
Figure 92: Contour plot of $x$-velocity of a porous cylinder (300 m$^{-1}$) at 2.5 m/s............... 121
Figure 93: Contour plot of pressure coefficient of a porous cylinder (300 m$^{-1}$) at 2.5 m/s .... 121
Figure 94: Contour plot of TKE of a porous cylinder (300 m$^{-1}$) at 2.5 m/s....................... 121
Figure 95: Contour plot of $x$-velocity of a porous cylinder (750 m$^{-1}$) at 2.5 m/s............... 122
Figure 96: Contour plot of pressure coefficient of a porous cylinder (750 m$^{-1}$) at 2.5 m/s .... 122
Figure 97: Contour plot of TKE of a porous cylinder (750 m$^{-1}$) at 2.5 m/s....................... 122
Figure 98: Contour plot of $x$-velocity of a porous cylinder (1500 m$^{-1}$) at 2.5 m/s........... 123
Figure 99: Contour plot of pressure coefficient of a porous cylinder (1500 m$^{-1}$) at 2.5 m/s .... 123
Figure 100: Contour plot of TKE of a porous cylinder (1500 m$^{-1}$) at 2.5 m/s.................... 123
Figure 101: Contour plot of $x$-velocity along a solid cylinder at 5 m/s ............................ 124
Figure 102: Contour plot of pressure coefficient along a solid cylinder at 5 m/s.................... 124
Figure 103: Contour plot of TKE along a solid cylinder at 5 m/s ................................. 124
Figure 104: Contour plot of $x$-velocity of a porous cylinder (15 m$^{-1}$) at 5 m/s............... 125
Figure 105: Contour plot of pressure coefficient of a porous cylinder (15 m$^{-1}$) at 5 m/s ........... 125
Figure 106: Contour plot of TKE of a porous cylinder (15 m$^{-1}$) at 5 m/s......................... 125
Figure 107: Contour plot of $x$-velocity of a porous cylinder (75 m$^{-1}$) at 5 m/s..................... 126
Figure 108: Contour plot of pressure coefficient of a porous cylinder (75 m$^{-1}$) at 5 m/s........... 126
Figure 109: Contour plot of TKE of a porous cylinder (75 m$^{-1}$) at 5 m/s............................ 126
Figure 110: Contour plot of $x$-velocity of a porous cylinder (150 m$^{-1}$) at 5 m/s.................... 127
Figure 111: Contour plot of pressure coefficient of a porous cylinder (150 m$^{-1}$) at 5 m/s........... 127
Figure 112: Contour plot of TKE of a porous cylinder (150 m$^{-1}$) at 5 m/s............................ 127
Figure 113: Contour plot of $x$-velocity of a porous cylinder (300 m$^{-1}$) at 5 m/s..................... 128
Figure 114: Contour plot of pressure coefficient of a porous cylinder (300 m$^{-1}$) at 5 m/s.......... 128
Figure 115: Contour plot of TKE of a porous cylinder (300 m$^{-1}$) at 5 m/s............................ 128
Figure 116: Contour plot of $x$-velocity of a porous cylinder (750 m$^{-1}$) at 5 m/s..................... 129
Figure 117: Contour plot of pressure coefficient of a porous cylinder (750 m$^{-1}$) at 5 m/s........... 129
Figure 118: Contour plot of TKE of a porous cylinder (750 m$^{-1}$) at 5 m/s............................ 129
Figure 119: Contour plot of $x$-velocity of a porous cylinder (1500 m$^{-1}$) at 5 m/s................... 130
Figure 120: Contour plot of pressure coefficient of a porous cylinder (1500 m$^{-1}$) at 5 m/s........ 130
Figure 121: Contour plot of TKE of a porous cylinder (1500 m$^{-1}$) at 5 m/s............................ 130
Figure 122: Contour plot of $x$-velocity along a solid cylinder at 7.5 m/s............................... 131
Figure 123: Contour plot of pressure coefficient along a solid cylinder at 7.5 m/s...................... 131
Figure 124: Contour plot of TKE along a solid cylinder at 7.5m/s .......................... 131

Figure 125: Contour plot of x-velocity of a porous cylinder (15 m$^{-1}$) at 7.5m/s .................. 132

Figure 126: Contour plot of pressure coefficient of a porous cylinder (15 m$^{-1}$) at 7.5m/s ....... 132

Figure 127: Contour plot of TKE of a porous cylinder (15 m$^{-1}$) at 7.5m/s .......................... 132

Figure 128: Contour plot of x-velocity of a porous cylinder (75 m$^{-1}$) at 7.5m/s .................. 133

Figure 129: Contour plot of pressure coefficient of a porous cylinder (75 m$^{-1}$) at 7.5m/s ....... 133

Figure 130: Contour plot of TKE of a porous cylinder (75 m$^{-1}$) at 7.5m/s .......................... 133

Figure 131: Contour plot of x-velocity of a porous cylinder (150 m$^{-1}$) at 7.5m/s .................. 134

Figure 132: Contour plot of pressure coefficient of a porous cylinder (150 m$^{-1}$) at 7.5m/s ....... 134

Figure 133: Contour plot of TKE of a porous cylinder (150 m$^{-1}$) at 7.5m/s .......................... 134

Figure 134: Contour plot of x-velocity of a porous cylinder (300 m$^{-1}$) at 7.5m/s .................. 135

Figure 135: Contour plot of pressure coefficient of a porous cylinder (300 m$^{-1}$) at 7.5m/s ....... 135

Figure 136: Contour plot of TKE of a porous cylinder (300 m$^{-1}$) at 7.5m/s .......................... 135

Figure 137: Contour plot of x-velocity of a porous cylinder (750 m$^{-1}$) at 7.5m/s .................. 136

Figure 138: Contour plot of pressure coefficient of a porous cylinder (750 m$^{-1}$) at 7.5m/s ....... 136

Figure 139: Contour plot of TKE of a porous cylinder (750 m$^{-1}$) at 7.5m/s .......................... 136

Figure 140: Contour plot of x-velocity of a porous cylinder (1500 m$^{-1}$) at 7.5m/s .............. 137

Figure 141: Contour plot of pressure coefficient of a porous cylinder (1500 m$^{-1}$) at 7.5m/s ..... 137

Figure 142: Contour plot of TKE of a porous cylinder (1500 m$^{-1}$) at 7.5m/s .......................... 137
Figure 143: Contour plot of x-velocity along a solid cylinder at 10 m/s ............................ 138
Figure 144: Contour plot of pressure coefficient along a solid cylinder at 10 m/s.......... 138
Figure 145: Contour plot of TKE along a solid cylinder at 10 m/s .............................. 138
Figure 146: Contour plot of x-velocity of a porous cylinder (15 m$^{-1}$) at 10 m/s........ 139
Figure 147: Contour plot of pressure coefficient of a porous cylinder (15 m$^{-1}$) at 10 m/s .... 139
Figure 148: Contour plot of TKE of a porous cylinder (15 m$^{-1}$) at 10 m/s.................. 139
Figure 149: Contour plot of x-velocity of a porous cylinder (75 m$^{-1}$) at 10 m/s........... 140
Figure 150: Contour plot of pressure coefficient of a porous cylinder (75 m$^{-1}$) at 10 m/s ...... 140
Figure 151: Contour plot of TKE of a porous cylinder (75 m$^{-1}$) at 10 m/s................... 140
Figure 152: Contour plot of x-velocity of a porous cylinder (150 m$^{-1}$) at 10 m/s........... 141
Figure 153: Contour plot of pressure coefficient of a porous cylinder (150 m$^{-1}$) at 10 m/s ...... 141
Figure 154: Contour plot of TKE of a porous cylinder (150 m$^{-1}$) at 10 m/s.................. 141
Figure 155: Contour plot of x-velocity of a porous cylinder (300 m$^{-1}$) at 10 m/s........... 142
Figure 156: Contour plot of pressure coefficient of a porous cylinder (300 m$^{-1}$) at 10 m/s ...... 142
Figure 157: Contour plot of TKE of a porous cylinder (300 m$^{-1}$) at 10 m/s................... 142
Figure 158: Contour plot of x-velocity of a porous cylinder (750 m$^{-1}$) at 10 m/s........... 143
Figure 159: Contour plot of pressure coefficient of a porous cylinder (750 m$^{-1}$) at 10 m/s ...... 143
Figure 160: Contour plot of TKE of a porous cylinder (750 m$^{-1}$) at 10 m/s................... 143
Figure 161: Contour plot of x-velocity of a porous cylinder (1500 m$^{-1}$) at 10 m/s............ 144
Figure 162: Contour plot of pressure coefficient of a porous cylinder (1500 m$^{-1}$) at 10 m/s ..... 144
Figure 163: Contour plot of TKE of a porous cylinder (1500 m$^{-1}$) at 10 m/s ......................... 144
Figure 164: Contour plot of x-velocity along a solid cylinder at 12.5 m/s ................................. 145
Figure 165: Contour plot of pressure coefficient along a solid cylinder at 12.5 m/s.................... 145
Figure 166: Contour plot of TKE along a solid cylinder at 12.5 m/s ...................................... 145
Figure 167: Contour plot of x-velocity of a porous cylinder (15 m$^{-1}$) at 12.5 m/s ...................... 146
Figure 168: Contour plot of pressure coefficient of a porous cylinder (15 m$^{-1}$) at 12.5 m/s .... 146
Figure 169: Contour plot of TKE of a porous cylinder (15 m$^{-1}$) at 12.5 m/s ............................ 146
Figure 170: Contour plot of x-velocity of a porous cylinder (75 m$^{-1}$) at 12.5 m/s ..................... 147
Figure 171: Contour plot of pressure coefficient of a porous cylinder (75 m$^{-1}$) at 12.5 m/s .... 147
Figure 172: Contour plot of TKE of a porous cylinder (75 m$^{-1}$) at 12.5 m/s ........................... 147
Figure 173: Contour plot of x-velocity of a porous cylinder (150 m$^{-1}$) at 12.5 m/s ................. 148
Figure 174: Contour plot of pressure coefficient of a porous cylinder (150 m$^{-1}$) at 12.5 m/s .... 148
Figure 175: Contour plot of TKE of a porous cylinder (150 m$^{-1}$) at 12.5 m/s .......................... 148
Figure 176: Contour plot of x-velocity of a porous cylinder (300 m$^{-1}$) at 12.5 m/s ................. 149
Figure 177: Contour plot of pressure coefficient of a porous cylinder (300 m$^{-1}$) at 12.5 m/s .... 149
Figure 178: Contour plot of TKE of a porous cylinder (300 m$^{-1}$) at 12.5 m/s .......................... 149
Figure 179: Contour plot of x-velocity of a porous cylinder (750 m$^{-1}$) at 12.5 m/s ................. 150
Figure 180: Contour plot of pressure coefficient of a porous cylinder (750 m$^{-1}$) at 12.5 m/s .... 150
Figure 181: Contour plot of TKE of a porous cylinder (750 m$^{-1}$) at 12.5 m/s .................... 150
Figure 182: Contour plot of $x$-velocity of a porous cylinder (1500 m$^{-1}$) at 12.5 m/s .......... 151
Figure 183: Contour plot of pressure coefficient of a porous cylinder (1500 m$^{-1}$) at 12.5 m/s .. 151
Figure 184: Contour plot of TKE of a porous cylinder (1500 m$^{-1}$) at 12.5 m/s .................... 151
Figure 185: Contour plot of $x$-velocity along a solid cylinder at 15 m/s ............................... 152
Figure 186: Contour plot of pressure coefficient along a solid cylinder at 15 m/s .................... 152
Figure 187: Contour plot of TKE along a solid cylinder at 15 m/s ........................................ 152
Figure 188: Contour plot of $x$-velocity of a porous cylinder (15 m$^{-1}$) at 15 m/s .................. 153
Figure 189: Contour plot of pressure coefficient of a porous cylinder (15 m$^{-1}$) at 15 m/s .... 153
Figure 190: Contour plot of TKE of a porous cylinder (15 m$^{-1}$) at 15 m/s ........................... 153
Figure 191: Contour plot of $x$-velocity of a porous cylinder (75 m$^{-1}$) at 15 m/s ................. 154
Figure 192: Contour plot of pressure coefficient of a porous cylinder (75 m$^{-1}$) at 15 m/s .... 154
Figure 193: Contour plot of TKE of a porous cylinder (75 m$^{-1}$) at 15 m/s .......................... 154
Figure 194: Contour plot of $x$-velocity of a porous cylinder (150 m$^{-1}$) at 15 m/s ............... 155
Figure 195: Contour plot of pressure coefficient of a porous cylinder (150 m$^{-1}$) at 15 m/s .... 155
Figure 196: Contour plot of TKE of a porous cylinder (150 m$^{-1}$) at 15 m/s ....................... 155
Figure 197: Contour plot of $x$-velocity of a porous cylinder (300 m$^{-1}$) at 15 m/s ............... 156
Figure 198: Contour plot of pressure coefficient of a porous cylinder (300 m$^{-1}$) at 15 m/s .... 156
Figure 199: Contour plot of TKE of a porous cylinder (300 m$^{-1}$) at 15 m/s .......................... 156
Figure 200: Contour plot of x-velocity of a porous cylinder (750 m⁻¹) at 15 m/s ....................... 157

Figure 201: Contour plot of pressure coefficient of a porous cylinder (750 m⁻¹) at 15 m/s ...... 157

Figure 202: Contour plot of TKE of a porous cylinder (750 m⁻¹) at 15 m/s ......................... 157

Figure 203: Contour plot of x-velocity of a porous cylinder (1500 m⁻¹) at 15 m/s ............... 158

Figure 204: Contour plot of pressure coefficient of a porous cylinder (1500 m⁻¹) at 15 m/s ..... 158

Figure 205: Contour plot of TKE of a porous cylinder (1500 m⁻¹) at 15 m/s ..................... 158

Figure 206: Rankine oval [2]. ........................................................................................................ 227

Figure 207: Coordinate placement of source and sink ................................................................ 228

Figure 208: Calibration curve along test section at fan speed = 5.9 Hz ............................. 242

Figure 209: Calibration curve along test section at fan speed = 14.4 Hz ............................ 243

Figure 210: Calibration curve along test section at fan speed = 21.9 Hz ......................... 243

Figure 211: Raw data of measurements in the wind tunnel at 21.9 Hz ............................. 244

Figure 212: The velocity profile at each of the three heights measured from the bottom of the test section at 4 in., 4.5 in. and 5 in. (a) The measured velocity profile at 2.5 m/s (b) The measured velocity profile at 7.5 m/s (c) The measured velocity profile at 10 m/s (d) The measured velocity profile at 12.5 m/s ...................................................................................................................... 246

Figure 213: Comparison of normalized streamwise velocity of numerical results with [67] and [72] ............................................................................................................................................. 247

Figure 214: Comparison of streamwise experimental velocity with [72] for solid case. ........ 248
List of Tables

Table 1: Dimension values for geometry setup in figure 11 ................................. 42
Table 2: Mesh zone description for simulation ................................................. 45
Table 3: Boundary Conditions for the flow simulation ............................................ 50
Table 4: Solution Methods from ANSYS Fluent .................................................. 52
Table 5: Simulation controls setting Under-Relaxation Factors .............................. 52
Table 6: Latin square of parameters for numerical simulation .................................. 53
Table 7: Mesh dependency test of six mesh sizes ............................................... 56
Table 8: Error comparison of numerical simulation with [67] ................................. 87
Table 9: Source/sink strength estimation ............................................................. 93
Table 10: Run time duration and convergence for all simulation runs ....................... 116
Table 11: Tree classification in Minnesota based on canopy shape at maturity .......... 235
### Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>Aerodynamic porosity</td>
</tr>
<tr>
<td>ρ</td>
<td>Density</td>
</tr>
<tr>
<td>μ</td>
<td>Kinematic viscosity</td>
</tr>
<tr>
<td>λ</td>
<td>Inertial resistance coefficient</td>
</tr>
<tr>
<td>k</td>
<td>Permeability</td>
</tr>
<tr>
<td>k_{fh}</td>
<td>Forchheimer permeability</td>
</tr>
<tr>
<td>m_{so}</td>
<td>Source strength</td>
</tr>
<tr>
<td>m_{si}</td>
<td>Sink strength</td>
</tr>
<tr>
<td>S_s</td>
<td>Sink spacing</td>
</tr>
<tr>
<td>s_{sopt}</td>
<td>Optimum sink spacing</td>
</tr>
<tr>
<td>D</td>
<td>Diameter</td>
</tr>
<tr>
<td>r_u</td>
<td>Diameter upstream</td>
</tr>
<tr>
<td>r_w</td>
<td>Wake Diameter</td>
</tr>
<tr>
<td>2-D</td>
<td>Two dimensional plane</td>
</tr>
<tr>
<td>3-D</td>
<td>Three dimensional plane</td>
</tr>
<tr>
<td>LAD</td>
<td>Leaf Area Density</td>
</tr>
<tr>
<td>LAI</td>
<td>Leaf Area Index</td>
</tr>
<tr>
<td>NASDAQ</td>
<td>National Association of Securities Dealers Automated Quotations</td>
</tr>
<tr>
<td>TI</td>
<td>Turbulence intensity</td>
</tr>
<tr>
<td>TKE</td>
<td>Turbulence kinetic energy</td>
</tr>
<tr>
<td>PM</td>
<td>Porous Media</td>
</tr>
<tr>
<td>PIV</td>
<td>Particle image velocimetry</td>
</tr>
<tr>
<td>Re</td>
<td>Reynolds Number</td>
</tr>
<tr>
<td>Da</td>
<td>Darcy Number</td>
</tr>
</tbody>
</table>
Chapter 1 - Introduction

In building and environment aerodynamics, trees serve as blockage to airflow around objects, improve pedestrian comfort and aid in the deposition of particles. In recent decades, much work has been done to investigate the effect of trees in airflow through numerical simulation, wind tunnel, and field experiments [1-9]. Currently, Minnesota is among the leading states in the Midwest of the United States for the generation and distribution of renewable energy.

Xcel Energy Inc. (NASDAQ: XEL) is a public utility company based in Minneapolis, Minnesota that serves more than 3.3 million electricity customers and 1.8 million natural gas customers in several states across the Midwest and southern region of United States of America. As of 2019, the renewable energy mandate set forth in 2007 has been surpassed by Xcel Energy, exceeding 38 percent generation from clean energy sources. The new goal as outlined in the 2018 Xcel Energy carbon report is to reliably and cost-effectively cut carbon emissions 80 percent by 2030 with the renewable and carbon-free generation and energy storage technologies available today [10].

Wind energy has been a primitive form of harvesting energy. Examples can be seen from ancient times when windmills were used to draw water from beneath the earth surface, to modern day offshore floating wind turbines. From the development of horizontal wind axis turbines (HAWTs) to modern-day vertical axis wind turbines (VAWTs), the analysis of the aerodynamics of the flow around VAWTs tends to be more complex [11, 12]. VAWT is an omnidirectional wind turbine that generates electrical power with the axis or main shaft mounted vertically, or perpendicular to the flow direction. With fewer components that do not need to be placed high up in the air and low footprint in windfarm installations, they provide versatility to be placed in both urban and rural locations as well as provide large- and small-scale energy generation solutions [13].
The analysis of the flow regime around VAWTs is governed by the Navier-Stokes equations. A complete analysis considers the fluid flow and fluid-surface interactions to fully approximate the performance of a VAWT. To improve VAWT performance, consideration needs to be made for the surrounding structures of VAWT. A wide variety of techniques such as blade element methods, vortex methods, Navier-Stokes (grid) methods and leaky-Rankine body (LRB) have been reviewed and used to determine the aerodynamic performance of VAWT blade and array configurations. Among these techniques is the low-order model developed in [14] that utilizes the potential flow theory, the flow regime around a VAWT and its nearby wake (i.e. the region of recirculating or disturbed flow downstream of a body moving though a fluid or caused by the flow of a moving fluid) to compute the resulting velocity and lift components [1]. Potential flow can help reduce the computational time for predicting the performance of VAWTs when combined appropriately with well-defined and validated high fidelity computational fluid dynamics (CFD) simulations.

In the VAWT placement improvement research project currently ongoing at MNSU, a numerical approach using the panel method, point source/sink and vortex singularities has been developed to describe the flow properties around buildings at building height level utilizing potential flow assumptions [15]. This method although very descriptive of varying geometries does not include the effect of vegetative structures.

According to [16], who utilized a 3-D Reynolds-Averaged Navier-Stokes (RANS) standard turbulence method in a built environment in COMSOL Multiphysics, CFD simulations are good tools for estimating the mean wind speeds and the energy production at the site. In [17], the authors studied the uniform potential flow past a slightly deformed porous cylinder, their results show the computation of the drag force exerted by the exterior flow on the surface of the cylinder, depended on the thickness of the porous material and the permeability of the porous regions. From [18,19],
the drag on a porous cylinder can be investigated by using the Navier-Stokes equations on the outer region and Darcy’s law on the inner region.

However, given the improvements made in the application of potential flow to complex geometries, no known metric is available for porous media or inclusion of vegetative structures. To optimize the placement of VAWTs in the urban environment, research should consider the varying terrain, buildings, and vegetative structures to better predict the flow field and forces. Then in turn, the performance of the wind turbines can be better estimated [20-22].

The contribution of this alternate plan paper will aim to model the flow of air through and around vegetative structures like trees, with trees modeled as pervious objects. It explores the use of the porous zone formulation to aid in obtaining a more accurate representation of the flow variables (pressure and velocity) in the 2-D (x-y plane) environment compared to previously simulated potential flow and 2-D CFD results where trees are not fully considered. The primary objective of this alternate plan paper is to conduct a comparative assessment of the effect of a range of incoming wind velocities and inertial resistance coefficients on the velocity reduction around a single tree to improve the prediction of flow variables in the potential flow code created to determine the placement of VAWTs in the urban and rural environment. The data created is then used to determine the source and sink strengths ratio for the refinement of the flow field estimation of the panel code.

Chapter 2 and 3 covers pertinent background and literature review on the previous experimental and numerical work done in the field on trees. Chapter 4 covers the numerical method utilized. Chapter 5 presents the results and discussion from the numerical simulation. Finally, Chapter 6
presents conclusions, VAWT placement recommendations and future work that can be done in this area.
Chapter 2 - Background

This chapter provides the pertinent background into flow in non-porous and porous media, CFD, discretization, meshing, turbulence modeling, VAWTs and the DOE technique used in this study. It also provides a background into the nature of trees and relevance in this paper.

Flow through non-porous media

Porous media (PM) flow is everywhere, from the flow of blood and nutrients in bones to the flow of water beneath the earth surface, we see PM flow in biological and environmental systems. To understand the flow around a tree, we first must understand the flow around non-porous objects.

Consider the external flow over a cylinder with infinite depth as presented fluid mechanics literature in [1-3], the flow around the cylindrical object possesses a thin boundary layer in the windward (upstream) direction and then separates the flow moves leeward (downstream). This flow creates two regions: an inviscid and viscous region.

In high Reynolds number (Re ≥ 10^4) flows, the viscous effects could be confined to thin boundary layers near the solid surfaces plus separated flow and wake regions which occur in adverse gradients.

Inviscid theory provides excellent results for the flow in front of the body but not in the rear where separated flow could occur. An inviscid flow is a flow where the dissipative transport phenomenon of viscosity, mass diffusion and thermal conductivity are neglected.
Under a barotropic assumption (density is a function of pressures only), for low-Mach number flows the continuity and momentum equations for a frictionless incompressible flow can be represented as

\[
\frac{\partial u_i}{\partial x_i} = 0 \quad (1)
\]

\[
\rho \frac{\partial u_i u_j}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \left( \rho \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right) - \rho \frac{\partial}{\partial x_j} \left( u_i' u_j' \right) + S_i \quad (2)
\]

The above form of the equations in (1) and (2) are known as the conservative form. These partial differential equations albeit complex can be solved for the velocity and pressure subject to given boundary conditions. It is worth mentioning that most commercial software solve these equations in the integral form because they allow for discontinuities inside the fixed control volume.

The flow is presumed to be irrotational (i.e. there are no rotational elements in the flow, thus the curl of the velocity is zero) by making the following assumptions, neglect:

a. Viscous effects
b. Entropy gradients
c. Stratification
d. Non-inertial effects

\[ V = \nabla \phi \]  

(3)

Or

\[ u = \frac{\partial \phi}{\partial x} \quad v = \frac{\partial \phi}{\partial y} \quad w = \frac{\partial \phi}{\partial z} \]  

(4)

This then reduces the continuity equation to the Laplace equation

\[ \nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2} = 0 \]  

(5)

And the momentum equation becomes the Bernoulli equation.

\[ \frac{\partial \phi}{\partial t} + \frac{p}{\rho} + \frac{1}{2} V^2 + gz = \text{const} \quad \text{where} \ V = |\nabla \phi| \]  

(6)

For steady flows, \( \frac{\partial \phi}{\partial t} = 0 \) Boundary conditions on the equation are

a. Known velocity in the free stream or other open stream boundary

b. Velocity normal to the boundary at solid surfaces \( \frac{\partial \phi}{\partial n} = 0 \), \( n \) is perpendicular to the body

If a flow is described by only two coordinates, the stream function also exists as an alternate approach.

**Governing laws for non-porous (solid objects) flow**

From the Karman analysis of a flat plate, the momentum thickness is a measure of the total plate drag. Consider a flow around a solid circular cylinder with infinite depth as in figure 1.

The forces that act on a solid object in fluid flow are a result of surface (e.g. pressure) forces or body (e.g. gravity) forces. These forces give rise to the adverse pressure gradient in the windward
direction and favorable pressure gradient in the leeward side of the flow lead to separation and reattachment respectively.

Although the NS equations can provide reasonable assumptions of the flow field, the presence of permeability structures in the flow limits the effectiveness of the solutions provided. This is one of the limitations of using simple assumptions when considering porous media.

**Review of Potential Flow Theory**

Potential flow theory is essentially governed by the simplification of the four conservation equations of the Navier-Stokes equations based on the above assumptions. Potential flow describes the velocity field as the gradient of a scalar function: the velocity potential. As a result, a potential flow is characterized by an irrotational velocity field.

\[ V = \nabla \phi \]  

(7)

All potential flows are irrotational (curl of velocity is zero) and the continuity equation must be satisfied by \( \phi \). For incompressible flows where viscosity is neglected,

\[ \nabla \cdot V = 0 \]  

(8)

This results in the Laplace equations which is the continuity equation.

Some of the techniques for finding potential functions that satisfy the Laplace equations include:

1) Superposition of elementary functions (i.e. source/sink, uniform flow, line vortex, etc.)
2) Numerical analysis
3) Conformal mapping
4) Electrical analogs
5) Mechanical analogs

The potential flow is a simplification of nature and as such are not accurate near solid bodies due to boundary layer development.
Flow through porous media

The flow through porous media is mainly governed by the permeability of the material to the flow of fluid. Darcy’s law has been extensively researched and extensions have been added to account for regimes where the viscous linear behavior is dominated by the inertial quadratic behavior (i.e. high Reynolds number flows).

Figure 2: Drag coefficient at $Re \geq 10^4$. [1]

Principles governing porous flow

When airflow approaches the plant, the pressure in front of the canopy increases, and the blockage effect results in cross-flow perpendicular to the wind direction, which is also called the displacement flow. The large part of the airflow that penetrates the canopy with its velocity gradually reduced is called the bleeding flow. The zone with low wind velocity behind the canopy is called the quiet zone. Fluid within the tree is governed by the Darcy’s law proposed by French engineer Henry P. G. Darcy to explain the behavior of the flow of fluid through a porous membrane.

\[
Q = \frac{-kA(p_b - p_o)}{\mu L}
\]  

(9)

Where $Q =$ volumetric flow rate

\begin{align*}
  k &= \text{permeability of porous medium} \\
  \mu &= \text{fluid viscosity}
\end{align*}
\[ A = \text{cross sectional area} \]

\[ (p_b - p_o) = \text{pressure drop across medium} \]

\[ L = \text{length of sample} \]

It states that the volumetric flux is proportional to the hydraulic potential. The constant of proportionality is the hydraulic conductivity.

\[ u = \frac{k \partial \phi}{\mu \partial x} \]  \hspace{1cm} (10)

From soil mechanics, we can express the object and model of the tree with a different viscosity and porosity as in [23]. The fluid flowing through a volume is a function of the superficial velocity. Generally, due to the Darcy’s law we are solving the conservation of mass, which is predominantly a change in pressure.

**Forchheimer extension of Darcy’s flow regime**

For flows with Reynolds number higher than 1, the original form of the Darcy’s law does not hold because of the strong influence of inertia. This gave rise to what is commonly known as the inertial resistance coefficient or Forchheimer’s coefficient. This coefficient exhibits a quadratic behavior with the velocity flowing through the porous body.
In [29], Forchheimer extended the Darcy law regime for porous flows to account for the high Reynolds number flow regimes. Figure 3 shows the flow regimes in porous media. The linear regime is where Darcy’s law operates, the weak inertia is a regime in which the inertial force is of the same order as the viscous force; while Forchheimer region is where a pressure drop is proportional to the square of seepage velocity.

This term is important as it presents one of two coefficients that CFD algorithms use in describing the flow behavior in the porous zone.

**Overview of CFD Modeling**

Computational fluid dynamics (CFD) takes advantage of computing power to iteratively solve the conservation or non-conservation forms of mass, momentum and energy equations governing the flow (i.e. Navier-Stokes (NS) equations with necessary boundary conditions, discretization and meshing). Computational fluid dynamics has grown over the past century and is used to understand the flow around simple and complex flow using computers. CFD is currently being used as a multidisciplinary tool to aid researchers gain better understanding of these kinds of air flow to optimize performance for power generation, pedestrian comfort, building performance. Most
commercial code software being developed today have been tested extensively and have pros and cons for various types of flow. Some algorithms and codes are better suited for subsonic, transonic, supersonic flows and/or a combination of flow types. In conducting simulations in ANSYS Fluent, physics-based meshing with the default hydrodynamics physics reference will be used. This paper will utilize a part-based meshing method.

**Discretization**

There are three forms of discretization that is commonly used in CFD. They are:

1. Finite difference method (FDM)
2. Finite element method (FEM)
3. Finite volume method (FVM)

The reader is encouraged to explore the details of each method in [1–3, 30-31] where more information on these methods are available. FVM is being applied using the commercial research license provided in this thesis.

FVM method was developed by Patankar and Spalding (1972). It is a discretization/subdomain method with piecewise definition of the field variable whereby the flow is divided into control volumes and the flow variables are computed either in the conservative or non-conservative form of the fundamental flow equations. In FVM, the partial differential equations of the NS equations are represented in algebraic form at discrete place on a meshed geometry.

**Meshing**

Meshing (also known as grid/element generation) is defined as the process of dividing the given physical domain into smaller subdomains (called cells or elements). A set of points distributed over the problem domain for a numerical solution for a set of partial differential equations. The
type of grid will depend on the discretization technique, geometry of the domain and underlying physics.

Meshing is considered one of the most important parts of any numerical simulation. The use of appropriate mesh elements and features could mean the difference in accuracy between results and rate of convergence (or lack thereof). To achieve a reliable CFD solution to a specific problem, best practice meshing standards should be adhered to.

Where the flow is complex, it is standard practice to obtain results where specific flow variables don’t vary with mesh size variation. A change of less than 15 percent is considered acceptable in most cases for the relevant flow quantities such as velocity, pressure, residual and convergence properties.

Majority of the grids in CFD can be put into two categories, structured and unstructured grids. Structured grids are identified by regular connectivity in which grid points are placed orderly on the grid line while unstructured grids are identified by irregular connectivity. The grid lines in physical space pertain to constant coordinate values. Structured grids consist of elements that can be mapped to a rectangle in 2-D and hexahedral in 3-D while unstructured grids are expressed by triangles in 2-D and tetrahedral in 3-D. They are better than unstructured grids because of higher spatial resolution and better convergence. It is for these reasons a structured mesh with quadrilateral elements was chosen for this thesis.

Grid generation process generally follows these steps:

1. Problem domain is decomposed into subdomain (blocks).
2. In each block, requisite grid is generated
3. Check the mesh quality and modify as required.
**Mesh Refinement Requirements**

Some of the guidelines outlines in [32-41] that are important to consider are:

1. **Accuracy**: Skewness and aspect ratio of less than 0.95 and 100 respectively. A good mesh should have cells that are not too distorted, smooth cell transitions that are not stretched.

2. **Efficiency**: Lowest count desired for resolving overall flow features such as boundary layers. In order to resolve the velocity boundary layer, a minimum of 10-15 elements is needed across the boundary layer thickness.

3. **Easiness to generate**: Using quad-dominant mesh vs quad/tri hybrid mesh. Quad aligned mesh is more accurate than tri mesh with the same interval size. Although for complex flows, they lose their advantage.

**Quad vs. Tri Mesh**

Quad elements are better because they provide more accurate results when derived quantities need to be determined. They are interpolated to a higher degree in quad than in triangular elements, thus they are not suitable when anticipating large gradients. Although quad elements provide an extra pressure term even when it is not present in the problem, both types of elements can accommodate higher order formulation, such as in the 6-node triangle and 8-node rectangle, triangle meshes are easier to fill in around curved geometry and are preferred when knowledge of the pressure gradient is prevalent. Since the problem being solved in this work will need verification, it is highly recommended that quad mesh is used to provide better accuracy.

**Tolerances**

Since this paper’s simulation does not utilize mesh connections, a default *Snap to Boundary* tolerance with automatic mesh connections is recommended based on the element size factor. By
setting a high tolerance value, all bodies will be connected as low tolerances might lead to missed connections. Recommended practices are to measure the largest and smallest gaps between parts, note the curvature and mesh sizes. Using a global tolerance that is larger than the largest gap and smaller than the smallest mesh size; half of the smallest mesh size.

**Turbulence Modeling**

In many engineering applications, resolving the turbulence flow structure is not as important as obtaining the key flow quantities of engineering interest such as pressure distribution, time-averaged velocity, wall shear stress etc.

Turbulence is a phenomenon that exists in the 3-D space and can be modeled using three main approaches; direct numerical simulation (DNS), large eddy simulation (LES) and Reynolds-averages Navier-Stokes (RANS).

DNS solves all aspects of the flow including the large and small eddies.

LES solves the major parameters in the flow but averages the small eddies. The energy containing the large-scale eddies are computationally resolved, while the small-scale eddies, that tend to be more homogenous and isotropic are modeled. The LES equations are solved by applying a pseudo-spectral method in the horizontal directions and centered finite-difference method in the vertical direction. In [83], the computational cost required to perform LES computations is less than that of DNS but more than that of RANS.

RANS equations are time averaged equations of motion for fluid flow. The basic idea behind the equations is Reynolds decomposition, where an instantaneous quantity is decomposed into time-averaged and fluctuating quantities which cannot be calculated but must be modeled. Thus, the flow quantities are solved for while the eddies are averaged.
The third term in the equation above is called the Reynolds Stress, it contains the fluctuating component which hides the eddy effects. The above equation is not closed and thus involves modeling the Reynolds Stresses. The time averaged operator is a Reynolds operator, a mathematical operator given by averaging the fluid property over the fluid flow under the group of time translations.

The RANS equation needs to be closed by the application of turbulence models. Several closure models are provided using commercial numerical simulation software. A two-equation $k$-$\varepsilon$ model was used in this thesis. Several authors have conducted an in-depth review of turbulence models and their benefits and drawbacks in numerical simulations. The reader is referred to references for a detailed review of these models [31].

The transport equations for turbulent kinetic energy $k$ and its eddy dissipation rate $\varepsilon$ are solved for so that turbulent viscosity can be computed for RANS equations.

Turbulent kinetic energy is given by:

$$k = \frac{1}{2} \bar{u}_i' \bar{u}_i' = \frac{1}{2} (\bar{u}_x'u_x' + \bar{u}_y'u_y' + \bar{u}_z'u_z')$$

Dissipation rate of the turbulent kinetic energy is given by:

$$\varepsilon \equiv v \frac{\partial \bar{u}_i}{\partial x_j} \left( \frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right)$$

In the $k$-$\varepsilon$ model, the turbulent viscosity is assumed isotropic, the ratio between Reynolds stress and mean rate of deformations is the same in all directions.
The standard $k$-$\varepsilon$ turbulence model (Lauder and Spalding, 1974) is used and reduces the many unknowns and unmeasurable terms in the exact $k$-$\varepsilon$ equations to a set of equations which can be applied to a large number of turbulent applications. The standard $k$-$\varepsilon$ model is a two-equation turbulence model that allows the determination of turbulent length and time scale by solving two separate equations, one for kinetic energy and its dissipation rate.

For turbulent kinetic energy $k$

$$\frac{\partial (\rho k)}{\partial t} + \frac{\partial (\rho k u_i)}{\partial x_j} = \frac{\partial}{\partial x_j} \left[ \mu_t \frac{\partial k}{\partial x_j} \right] + 2\mu_t E_{ij} E_{ij} - \rho \varepsilon \quad (15)$$

For dissipation $\varepsilon$

$$\frac{\partial (\rho \varepsilon)}{\partial t} + \frac{\partial (\rho \varepsilon u_i)}{\partial x_j} = \frac{\partial}{\partial x_j} \left[ \mu_t \frac{\partial \varepsilon}{\partial x_j} \right] + C_{1\varepsilon} \frac{\varepsilon}{k} 2\mu_t E_{ij} E_{ij} - C_{1\varepsilon} \rho \frac{\varepsilon^2}{k} \quad (16)$$

Where

$u_i =$ velocity component in the corresponding direction

$E_{ij} =$ rate of deformation

$\mu_t =$ eddy viscosity

$$\mu_t = \rho C_{\mu} \frac{k^2}{\varepsilon} \quad (17)$$

The difference between RANS and LES momentum equations is the size of the eddy-viscosity provided by the underlying model. The most common $k$-$\varepsilon$ models are the standard and realizable models.
**Realizable $k$-$\varepsilon$ model**

This model provides improved predictions for the spreading of both planar and round jets. It exhibits superior performance for flows involving rotation, boundary layers and strand adverse pressure gradients, separation and recirculation. This model captures the mean flow of complex structures better than the standard or RNG $k$-$\varepsilon$ models.

Realizable $k$-$\varepsilon$ model satisfies the mathematical constraint on the Reynolds stresses, consistent with the physics of the turbulent flows. In order to predict the adverse pressure gradients, a $k$-$\omega$ turbulence model should be used to compare the results of the pressure distribution.

**Pressure Correction**

The concept of pressure correction is an iterative approach where some innovative physical reasoning is used to construct the next iteration from the results of the previous iteration. Several algorithms have been developed to determine the pressure distribution in a flow field. One of the most commonly used techniques is the semi implicit method for pressure-linked equations (SIMPLE) developed by Patankar and Spalding.

The pressure correction process is as follows:

1. Start the iterative process by guessing the pressure field, $(p^n)^*$
2. Use the $(p^n)^*$ to solve for the $(u^n)^*$ and $(v^n)^*$ from the momentum equations.
3. Use the continuity equations to construct a pressure correction $p'$ which is added to the $(p^n)^*$ to bring the velocity field in agreement with the continuity equation.
4. Calculate $p^{n+1}=(p^n)^* + p'$
5. Designate the new value of $p^{n+1}$ as $p'$ and repeat step 2
**Wall Functions**

Wall functions are functions that bridge the gap or link the solution variables at the near wall cells and corresponding quantities on the wall. They are mainly used in regions where there are not enough elements to describe the flow. Wall functions comprise of the law-of-the-wall for the mean velocity and temperature and formulae for the near-wall turbulent quantities.

ANSYS Fluent provides five approaches to wall functions: standard, scalable, non-equilibrium, enhanced and user-defined wall functions. Standard wall functions are widely used in industrial flow; they are dimensionless and are calculated based on the turbulence kinetic energy.

\[
y^* = \frac{\rho C_{\mu}^{1/4} k_p^{1/2} y_p}{\mu}
\]  

(18)

Where, \( \mu \) is the dynamic viscosity of the fluid

\( y_p \) is the distance from the centroid of the wall-adjacent cell to the wall, \( P \)

\( k_p \) is the turbulence kinetic energy at the wall-adjacent cell at centroid, \( P \)

\( u_p \) is the mean velocity of the fluid at the wall-adjacent cell at centroid, \( P \)

**2-D vs. 3-D flow**

In this section, we provide some background into the different features of 2-D vs. 3-D flow when considering the flow across a solid cylinder. To justify the 2-D slice of a tree which is inherently a 3-D simulation, we need to understand their flow dynamics.

In 3-D flows, the eddies transfer energy from the large eddies to the small eddies and in the 2-D flow, energy is being transferred from the small eddies to the large eddies.
Additionally, in 3-D flow, the height of the cylinder gives rise to the visible vortex shedding that shortens the distance of the recirculation region.

Currently, 2-D simulations of complex structures help provide basic understanding of the mean quantities of the flow. This is currently used in naval hydrodynamics, simulation of flow around VAWTs and airfoils.

**Review on VAWTs**

Vertical axis wind turbines (VAWTs) are a type of wind turbine whose rotor blades spin on an axis perpendicular to the ground. Recently, they have been suggested to achieve higher turbine output per unit land area than horizontal axis wind turbines (HAWTs). The main advantages of VAWTs is its single moving part (the rotor), with no need for yaw mechanisms leading to versatility in array configurations. They can be classified into two types based on the force driving mechanism; drag or lift driven VAWTs.

Drag force driven VAWTs (e.g. Savonius type) have cups or half drums fixed to a central shaft in opposing directions. Each cup/drum catches the wind and so turns the shaft, the process is repeated bringing the flow of the next cup in the wind. Typical maximum power coefficient values for this turbine vary between 30% to 45%. Savonius VAWT are more suitable for low-wind speed applications such as wind speed instruments.
Lift force driven VAWTs (e.g. Darrieus) consist of two or more aerofoil-shaped blades which are attached to a rotating shaft. The wind blowing on the aerofoil contours of the blade create aerodynamic lift which keeps the blades in rotation in the wind. The most common Darrieus VAWT shapes are the troposkien shape eggbeater-type which minimize bending stress in the blades, the simple straight bladed Darrieus VAWT (giromill or cyclo-turbine); utilized for its ease of aerodynamic analysis and the h-rotor; which is self-regulating in all wind speeds.

Some of the methods currently used in the aerodynamic analysis of VAWTs are cascade model, single streamtube model, multiple streamtube model, double multiple streamtube model and blade element model. Only the blade element model will be utilized in this thesis, the reader is referred to works in [28-34] for detailed description of these methods.

**Factors affecting performance of VAWTs**

To improve the performance efficiency of VAWTs, several approaches can be utilized [48-50]. Majority of these approaches include a simplification of the VAWT in the 2-D domain and employ CFD simulations. Prior to the installation of VAWTs, the region of interest must be analyzed for its wind power potential.
The degree to which the inquirer desires to have close to field results would determine what aspects of the flow are present. Buildings and vegetative structures are the most readily available structure when VAWT siting is conducted.

**Trees**

A tree, as in figure 5 is comprised of three basic parts; crown, bole and roots. The crown contains its leaves, twigs and branches and is sometimes referred to as canopy especially when considering multiple vegetative structures. The bole is the part of the tree between the ground and the first branches and consists of the trunk, the root lies below the ground, provides moment resistance and is not of interest in this study.

*Figure 5: Model of a basic tree [42]*

In [1], a tree has a flexible structure that allows it to adapt its shape in high winds and thus reduce drag and damage with its root providing resistance to wind-induced bending moments. They provide shade screening, noise reduction and protection from deposition of particles.

Some trees have more consistent windbreak properties while some have varying properties due to shedding of leaves [27, 28].
Previous investigations have shown that there is an area around a tree in the neutral atmospheric boundary layer known as the speed-up zone; where air accelerates, the quiet zone where the speed is zero and bleeding flow that passes through the foliage.

**Trees in Minnesota**

According to the Minnesota Department of Natural Resources, Minnesota has 52 native species. As of 2016, the number of trees in Minnesota’s forest land was 14.7 billion. This land is home to a wide variety of native and introduced species. Trees in Minnesota can be classified in to coniferous or deciduous. Coniferous trees (cedar, fir, hemlock, juniper, tamarack and spruce) have seeds in cones and needles that stay year-round. Deciduous trees (ash, birch, cherry, American elm, hackleberry, oak, poplar and willow) have covered seeds and leaves that drop in the fall.

The height and diameter of a mature tree is dependent on the age of the tree as well as the health condition of the tree. On average, a mature and healthy coniferous tree will grow over 40 feet. This height is well within the range to cause significant obstruction to airflow. Research and conventional knowledge have shown that this is a desired effect when considering pedestrian comfort at the base of the tree.

The table shown in Appendix F shows a list of the most common trees in Minnesota and the shape geometry of the trees at maturity. From the table 11, it is indicated that coniferous trees have a shape that can be approximated as a cone and most deciduous trees have a shape that ranges from cylindrical to upside-down bottom cone.

**Porosity**

Consider a sample of total volume $V$. Define the volume of the solid phase to be $V_s$, and the volume of the pore phase (the holes) to be, $V_p$ with $V = V_s + V_p$. The volume fraction is a normalized
variable that is generally more useful. The volume fraction of the pore phase is commonly called the porosity and is denoted \( \Phi = \frac{V_p}{V} \). Porosity is defined as the fraction of the total volume that is not occupied by the solid matter. For trees, this is the fraction of the leaf coverage area to the total volume of the tree. In [22], fifteen types of porosity common in the hydrocarbon industry were outlined. Defining porosity is of high importance in this study because trees act as permeable structures that extract energy from the air flow. This porosity is a function of the height, shape and foliage configurations and as such vary between tree species. For trees of mature age, certain forms of porosity are difficult to obtain without destroying the vegetation in question.

Optical, cubic and/or aerodynamic porosity alone is inadequate to determine resistance coefficient according to [70]. Porosity is one of the most important parameters that affect the flow around a tree. Based on the equation presented in [86], the leaf area density is a function of the viscous and inertial resistance. Although it is hard to determine the aerodynamic resistance experimentally, it can be deduced from the cubic or surface porosity [97]. The optical porosity is a much-desired property that can be derived from photograph [43].

**Permeability**

The permeability of trees is as a result of the flexibility of its leaves and branches in the presence of wind. When considering the tree canopy as a whole, the local deflections of the canopy are small compared to the individual leaves. For flows in higher speeds, the flow speeds are large enough around the canopy that the linear term which follows Darcy’s law is ignored.

For flows with Reynolds number less than 1 where the Darcy regime is valid, permeability of the porous media plays a huge role in the pressure drop across the domain. The permeability of the
porous domain (also referred to as viscous resistance) ensures that viscosity has a larger effect on the flow compared to inertial forces

**Determination of LAI**

As presented in the previous section, the Forchheimer coefficient can be obtained as a product of the drag coefficient, the lead area index (LAI) and the velocity squared. Even though this paper does not cover the determination of the drag coefficient or LAI, it is worth providing the reader with a basic understanding of how to obtain this parameter.

In the determination of the porosity, the drag coefficient and the leaf area index are important parameters in determining porosity of trees [44]. LAI is represented by the integral of the foliage area density, \( a(z) \), which is the area of plant surface plant surface per unit volume of space. LAI is the simplest useful measure of canopy area density. Plants with LAI greater than 5 are considered dense canopies.

Several methods have been developed to determine LAI. These methods can either destructive or non-destructive methods. Some are digital cover photography (DCP) method, fisheye method.

DCP method uses a simple DSLR camera and narrow field-of-view lens and minimal post-production and calculation to determine the crown porosity, LAI and clumping factor [43].

**Design of Experiment (Latin Square)**

In the design of experiments involving discrete extraneous variables, the objective is to design an experiment where the variation sequence of the extraneous variables plays little role in the estimation of the results.
A Latin square is an $n \times n$ array filled with $n$ different symbols, each occurring exactly once in each row and exactly once in each column. It is a way to vary the parameters affecting a phenomenon of interest so that it can be represented in a square and there are no repetitions.

For example,

$$\begin{bmatrix}
A & B & C \\
C & A & B \\
B & C & A
\end{bmatrix}$$

**Review on Actuator Disk Theory**

Actuator disk theory (also known as momentum theory) was developed by W.J. Rankine (1865), Alfred Greenhill (1888) and Robert Froude (1889). It is a theory that describes a mathematical model of an ideal actuator disk such as a propeller or helicopter rotor. A constant velocity is induced along the axis of rotation of the rotor, modeled as an infinitely thin disc. As such the flow of air around the rotor occurs within a streamtube.

The rotor extracts momentum and energy from the wind; and shear effects within the streamtube are neglected. Additionally, viscous losses are neglected to inhibit the generation of vorticity. The actuator disk is stationary (no rotation) and the wind flow in the streamtube is steady.
Figure 6: Streantube expansion, velocity decrease and pressure jump as wind passes through the actuator disk

**Potential Flow relationship to Actuator Disk Theory**

This method is commonly used today and was implemented in [14]. In this work, the authors used a lower order approximation of the velocity field to obtain the source and sink strength for a VAWT. By representing the velocity field in the complex plane, the flow field can be represented by a uniform flow, with source and sink placed away from each other.

The flow field of a leaky Rankine body is such that the sink strength is larger than the source strength.

To obtain the sink spacing, $S_s$, a curve fit was performed on the streamwise centerline velocity and then the sink spacing with the smallest variance and largest $R^2$ is chosen. The smallest residual for
the simulated porous wake velocity and potential flow wake velocity, $U_{4,sim} - U_{4,pflow}$ is also computed.

Once the sink spacing has been determined, using Bernoulli equation for pressure coupling and mass flow rate through the disk. An algebraic set of equations were then solved for each inertial resistance coefficient level.

The two-dimensional flow around the porous cylinder can be represented as the superposition of a uniform flow, source and sink pair of unequal strength. This is essentially a net sink flow. In fluid mechanics, the velocity field $(u, v)$ can be represented as

$$u, v(x, y) = U_\infty e^{-i\alpha} + \left[ \frac{m_{so}}{2\pi(x,y)} - \frac{m_{si}}{2\pi(x,y-x_s,y_s)} \right]$$

(19)

Where,

$U_\infty$ and $\alpha$ are the incoming velocity and direction,

$m_{so}, m_{so}$ are the source and sink strengths respectively

$x, y$ and $x_s, y_s$ are the location of the source and sink spacing respectively

In complex form, the velocity field can be mapped as

$$u, v = w \text{ and } x, y = z$$

Where $u + iv = w$ and $x + iy = z$, we have

$$w(z) = U_\infty e^{-i\alpha} + \left[ \frac{m_{so}}{2\pi(z)} - \frac{m_{si}}{2\pi(z-z_s)} \right]$$

(20)

For a source located at the origin, $x$ with incoming horizontal velocity $U_1$, the $y$-location component drops off.
The centerline velocity then becomes a function of the axial velocity which is

\[ U(x) = U_1 + \left[ \frac{m_{so}}{2\pi x} - \frac{m_{si}}{2\pi(x-x_s)} \right] \]  

(21)

This form of representing the velocity is used because we wish to make the potential flow representation analogous to the actuator disk for which the velocity is only valid along the centerline.

In this representation,

- \( U_1 \) is the freestream velocity,
- \( U_2 \) is the velocity some distance (\( r_{u} \)) upstream of the source where the object begin to effect by a velocity drop,
- \( U_3 \) is the velocity just to the right (leeward) of the sink; and
- \( U_4 \) is the wake velocity at a distance (\( r_{w} \)) far away from the source/sink pair.

We want to represent \( U_2 \) and \( U_4 \) in terms of the velocity field in equation (21) to compute the source and sink strength and sink spacing.

\[ U(-r_u) = U_2 = U_1 - \frac{m_{so}}{2\pi(r_u)} + \frac{m_{si}}{2\pi(r_u+s_u)} \]  

(22)

\[ U(r_w) = U_4 = U_1 + \frac{m_{so}}{2\pi(r_w)} - \frac{m_{si}}{2\pi(r_w-s_w)} \]  

(23)

To give the reader an idea of the distances, \( r_u \) and \( r_w \), \( r_u \) is the point where the incoming velocity begins to drop due to the source. This is assumed to be at least 6\( D \) (\( D \) is the diameter of the porous object) upstream. \( r_w \) is the point far from the source/sink pair where the velocity tries to return to freestream (at least 24\( D \) downstream). See Appendix E for full derivation of source/sink strengths.
Chapter 3 - Literature Review

The aim of this alternate plan paper is to develop a 2-D model based on pre-existing numerical models to model the flow of air through single stand-alone vegetative structures such as trees. Trees in this simulation will be modelled as a two-layer structure, with a trunk modeled as a solid object and the crown modeled as porous zones.

Some of the assumptions made are:

1. Structural deflection of the canopy is being neglected in order to maintain the rigidity of the structure for fluid surface interaction.

2. Inlet velocity value considered will be based on the gentle breeze to moderately strong wind conditions from the NOAA database.

3. Turbulence model used will be the $k-\varepsilon$ model but will ignore the wall effects.

Simulation of Trees and Canopies

Previously mentioned in Chapter 2, trees are objects that occur in nature. But the varying shape of deciduous trees make it easier for them to sway in the presence of moderate wind speeds. Coniferous trees, due to its shape limits the flow of air towards the bottom and allows for sway towards the top.

To understand the flow of air and deposition of particles through a tree, one must ascertain whether it is at all necessary. As such, previous research [19, 59] shows that the drag coefficient (a dimensionless quantity that describes the resistance of an object in a fluid) a measure of the reduced by 60 percent with the presence of trees. The review on urban tree modelling in [20] highlights the importance of representing vegetation to capture the effects of wind flow in urban areas. In the placement of wind turbines, site and resource assessment is one of the risk factors to consider as
outlined in the risk review study of onshore wind energy in Northern European forests [24]. Three approaches have been used in previous works to include the effect of trees on the estimation of flow properties in building and environment aerodynamics; basic, implicit and explicit approach. The basic approach neglects the effect of trees. The implicit approach includes the effects of trees as a surface parameterization. The explicit approach includes the trees, representing them as a porous media.

In the computational modeling of wind flow over the Calgary campus in [63], it was found that trees caused deviations in wind speed and turbulence and excluding trees led to higher wind power density (WPD). In another study, by varying tree heights with the wind flow over a building, [19] found that there is a reduced turbulence over the roof and the absence of a negative $\partial u/\partial z$, where $u$ is the mean horizontal velocity in the $x$-direction and $z$ is the distance in the $z$-direction.

Miao et al. [62] studied the wind flow and wind forces on trees in urban parks and concluded that tree shape must be considered and captured when developing CFD model to calculate accurate forces that acts upon a tree. The aerodynamic modeling of trees for small scale wind studies using different materials experimentally studied. They used twelve different model trees with crowns made of wood wool, sisal fibre and porous foam of 10 ppi (pores per inch) accounting for varying packing densities of the models [50].

To effectively represent vegetative structures, certain variables must be considered. The vegetative properties are (porosity, area, tree type) and wind conditions (velocity magnitude and direction). The investigation of flow considerations around model trees of different materials and incoming velocities using laser Doppler velocimetry (LDV) found out that the vertical velocity magnitudes had a maximum of 4% of the maximum horizontal velocity magnitude.
Numerical simulation of trees

Numerical simulation of tree canopies has been extensively studied. Previous work performed by Ruck and Schmitt [66] investigated the air flow through and around trees with two (cone and ball-shaped) canopy geometries served as the basis for later cited work in other related fields such as pollutant dispersion and pedestrian comfort [78].

The current aim of the RDF grant received by the MN State Mankato VAWT project is to create a model for the optimized placement of VAWTs in the urban environment. A more accurate model of the environment must include the vegetative, as well as manmade structures, i.e. buildings, towers, silos etc. as observed in analysis by Salim et al. [80] studying the dynamic effects of trees on the wind flow using three approaches.

Based on computational studies by Terziev et al. [22] and Mohamed and Wood [20], the flow of wind on a building is greatly damped by the presence of a vegetative structure. Studies from Ishikawa et al. [64] show that the average magnitude of the mean velocity in the x-y plane is decreased and after a certain distance, it straightens up and assumes a somewhat linear profile.

Kenjereš and Kulie [72] provided a numerically robust and physically accurate method for modeling turbulent flow in urban areas over the Delft University of Technology site. In this work, the authors use a passive element approach to reduce the intensity of wind gust.

A mechanical model was developed in [98] to predict wind damage for individual tree stands within a forest. Although they obtained reasonable predictions for the critical wind speeds, they were unable to account for damage in volume and tree number (i.e. population).
**Wind tunnel investigation of trees**

A preliminary literature review shows that past work has been done experimentally and numerically [52, 64, 67, 77-80]. Although majority of this work were also done computationally, they are split between three turbulence viscous models and involved the 3-D domain (x-y-z plane) to be able to compare with experimental results.

In [97], the authors conducted investigations around trees with three pine tree configurations measuring aerodynamic porosity and drag coefficient. They did not find any significant differences in terms of measuring distance and wind tunnel velocity on the aerodynamic porosity of the tree. They found aerodynamic porosity for single trees are generally higher and their effect on the flow is generally attributed to their resistance. They also found drag coefficient for single trees to be 0.55.

The experimental study in [64] identified two main characteristics of a living tree; permeability and flexibility. Experimental study done in [78] revealed that the porosity of a tree depends on the leaf area density (LAD) as well as other parameters, such as permeability and flexibility.

The focus of these studies had been on air pollution around vegetative structures. The studies which consider the flow of air through a vegetative structure, do so experimentally and in cases where they are done. They do not consider numerically the normalized velocity in the horizontal direction.

A study on the airflow around single and multiple plants by Liu et al. [70] showed that plant canopies with large bottoms and small tops have a better shelter efficiency, i.e. better wind reduction efficiency because of a larger upwards cross flow component.
The arrangement used in [94] was made of open rings made from metallic mesh in a boundary layer wind tunnel. Even though they did not mimic the shape, drag coefficient or LAI of the dense homogenous forest areas, they sought to achieve the aerodynamic properties inside and above the canopy. Wind tunnel validation of simulation of trees as porous medium was performed using hot film anemometer to measure the airflow [70].

Most experiments conducted around single trees in the wind tunnel used dead trees except for the work in [64], the authors investigated a young tree in a wind tunnel to obtain the drag coefficient variation with incoming wind velocity. They found out that the drag coefficient ranges from 0.8 to 1.2 within 5 to 15 m/s.

Experimental investigations in [72] were conducted using a rectangular flume. The authors investigated the wake structure of porous structures using 16 different geometry cases. They observe that the velocity drop was similar to solid cases, although they did not identify a stable wake distance for each case amongst other turbulent properties.

**Field investigation of trees**

Currently, as a part of the VAWT improvement project [15], field observations of the velocity field using anemometers placed around the built environment. In [74] performed an extensive investigation observing the behavior of airflow around trees for a span of 3 years.

Previous field experiment performed focused on either trees as forest areas and doing so to obtain a value for different properties such as drag coefficient [18,19,97], LAI [43], aerodynamic porosity [92].

What these field experiments has concluded is that the flow around the tree is affected by the tree foliage and its branches/needles. Coniferous trees tend to provide a larger absorption of the
momentum at the bottom of their foliage. However, since most field experiment were conducted in moderate wind conditions, tree flexibility played a factor in their results.
Chapter 4 - Method

This chapter deals with the numerical method utilized in modeling canopy flow in the 2-D domain using commercial software; ANSYS Workbench v19.1. It details the numerical setup of the flow geometry using ANSYS Design Modeler, meshing setup using ANSYS Meshing and the prescription of boundary conditions and solution methods using ANSYS Fluent.

Materials and Methods

The materials used in this paper will be mainly computer resources (for the numerical section) this thesis covers numerical simulations and post processing of results and ABS material for 3D printing tree models for the wind tunnel experiment (Appendix G).

In the numerical simulation, a single tree is modeled as a single canopy consisting of its trunk and foliage in 2-D as if a sectional slice was obtained at various heights through a coniferous tree. This reduces the inherent complexity of the real flow.

![Figure 7: Schematic description of areas used in domain](image)

This resulting 2-D model will then serve as the boundary for the tree. An enclosing boundary will be used for the boundary conditions based on literature and past experimental results as shown in the literature review.
ANSYS Fluent solves the modified Navier-Stokes equations using the finite volume method (FVM) at all element points. Using the assumptions for the steady and incompressible flow around a porous cylinder, the computation time is reduced while inferring reasonable results. ANSYS also contains various RANS and LES (only used in 3-D enabled flow) turbulence models that could be utilized.

This study utilized the following software packages ANSYS Workbench (Design Modeler, Meshing and Fluent) and Microsoft Office 365. Additionally, using MATLAB the results of the resulting flow field are compared to identify the tree and its wake region giving a representation of a tree as its source and sink strengths.

**Numerical Approach**

The numerical approach to simulating the PM flow will follow similar approach for the 3-D LES simulations that have been outlined in [21, 44, 45].

A 2-D model of a tree will be created and then an arbitrary resistance coefficient will be determined. An arbitrary inertial resistance will be assigned to the body. It is shown in [64] that the normalized leaf density varied with tree height depending on the leaf coverage. This study assumes that the tree has no foliage to reduce the associated computational cost as in [31].

Before the modelling and simulation of the trees is carried out, a mesh and grid independence study of the results will be done to observe the changes in the results and to validate unknown cases.

Since this study is a 2-D study, the drag coefficient, normalized horizontal velocity will be compared with previous experimental and numerical results. To identify the different regions of bleeding flow, displacement flow and reversed flow in and around the porous region.
Single Tree Simulation

Single trees are simulated using a single geometrical shape for the foliage area and the trunk. This is referred to as tree canopy in this thesis. The tree canopy is represented as a fluid under the cell zone conditions in ANSYS Fluent in order to activate the porous zone formulation. This tree canopy is modeled as a momentum sink comprising of viscous and inertial loss terms.

\[ S_i = -\left( \sum_{j=1}^{3} D_{ij} \mu v_j + \sum_{j=1}^{3} C_{ij} \frac{1}{2} \rho |v| v_j \right) \]

(25)

Where \( S_i \) is the source/sink term for the \( i \)th \((x, y)\) momentum equation, \(|v|\) is the magnitude of the velocity, \( C \) and \( D \) are prescribed matrices. The porosity is accounted for through the pressure loss coefficient \((\lambda)\). Various trees possess different porosity at different heights. In this simulation, tree porosity is obtained at mid height for cylindrical coniferous trees, 1/3 height for coniferous trees and 3/4 height for summer deciduous trees.

Numerical Modeling

Numerical flow simulations are performed in ANSYS FLUENT, using best practices and simulation techniques from Gan and Salim [65] and [44, 76-86, 95, 89].

In order to choose a numerical model, comparison was made to the wind tunnel experiments conducted in [45] and used as a benchmark.

To represent the flow as a typical atmospheric boundary layer flow in 2-D, the terrain was not considered in this 2-D numerical simulation. The inlet flow had a constant free-stream flow ranging from 2.5 m/s for light breeze to 12.5 m/s for strong breeze based on three key priorities.

1. VAWT operating conditions
2. Pedestrian level comfort on the Hardiness scale
3. Literature comparison range

The grid is fine and uniform inside the inner area. The computational domain and mesh given in Figure 8 and 12 is generated using quadrilateral elements and complies with the recommendations based on the wall $y+$ approach [99]. To reduce computational cost and maximize the numerical accuracy at regions of high solution gradients, the resolution is increased progressively at the vicinity of the tree canopy.

**Numerical Simulation Setup**

The setup for the simulation was conducted using ANSYS Workbench 19.1 in the VAWT Research lab at Minnesota State University, Mankato.

**Geometry Setup**

The tree was simulated as a cylinder 0.1 m wide diameter. The boundary for the flow domain was set at 4 m length by 2 m width. Best practice guidelines based on building, environmental and aerospace industry were used to determine a sufficiently large domain [37, 38]. Extensive care was taken to avoid recirculation of flow in the extended wake region.
The domain was split into various faces to enable a structured mesh to be developed later. A total of three sketches was made on the $XY$ plane.
Geometry Creation Steps

The first sketch was the cylinder geometry using “circle” feature. Using “Concept”, the sketch was converted to a surface and set to material. The second sketch was the fluid domain using “two-corner rectangle” and “circle” features. Using the “Concept” menu, the sketch was converted to a surface and set to frozen. The third sketch was a line body using the line feature. This sketch is to aid with the face splits and was suppressed afterwards to avoid meshing errors. A total of 12 face splits were made on the geometry. Figure 10 shows the model tree of the Design Modeler from ANSYS Geometry.

Figure 10: Model tree from ANSYS Geometry - Design Modeler
Each sketch was dimensioned prior to conversion to a line body or surfaces. The dimensions used for the geometry, the face splits and its placement are given in the table 1 and figure 11.

![Figure 11: Placement of dimensions for geometry](image)

### Table 1: Dimension values for geometry setup in figure 11

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Value (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>100</td>
</tr>
<tr>
<td>H2</td>
<td>200</td>
</tr>
<tr>
<td>V3</td>
<td>1000</td>
</tr>
<tr>
<td>L4</td>
<td>2000</td>
</tr>
<tr>
<td>L5</td>
<td>4000</td>
</tr>
<tr>
<td>H2</td>
<td>1100</td>
</tr>
<tr>
<td>H16</td>
<td>200</td>
</tr>
<tr>
<td>H19</td>
<td>1000</td>
</tr>
<tr>
<td>H20</td>
<td>4000</td>
</tr>
<tr>
<td>V17</td>
<td>900</td>
</tr>
<tr>
<td>V18</td>
<td>200</td>
</tr>
<tr>
<td>V6</td>
<td>80</td>
</tr>
</tbody>
</table>
**Meshing setup**

In CFD simulation involving external flow, two types of meshes are most common; structured and unstructured. A fully structured mesh was obtained for the simulation using ANSYS Meshing. Using the face splits obtained from the geometry section. Several decompositions were made to obtain the structured mesh shown in figure 12 through 14.

![Figure 12: Full view of structured mesh](image)

![Figure 13: Closeup of the mesh refined zone surrounding the porous zone](image)
Mesh Creation Steps

1. The default mesh size was set to 0.025 m. A separate face size was set for each zone. Zone A, B, C and D were set to 0.02, 0.01, 0.005 and 0.0025 m respectively. The size setting was set to “Hard” to enforce the size feature throughout the zones.

2. A face meshing was applied to each zone with quad dominance and default Advanced settings were used

3. An edge sizing was applied to vertical boundaries and horizontal edge boundaries to force the element sizing constraints.

To achieve this structured mesh, multiple face meshes, face sizings and edge sizings were used. A detail of this is outlined in the mesh model tree in figure 15.

Figure 14: Extreme closeup of the mesh porous zone area.
Table 2: Mesh zone description for simulation

<table>
<thead>
<tr>
<th>Mesh Zone</th>
<th>Location Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Outer boundary</td>
</tr>
<tr>
<td>B</td>
<td>Lateral and horizontal refined region</td>
</tr>
<tr>
<td>C</td>
<td>Rectangular near-cylinder region</td>
</tr>
<tr>
<td>D</td>
<td>Porous cylinder</td>
</tr>
</tbody>
</table>

Figure 15: Meshing model tree from ANSYS Meshing
Mesh Statistics

Figure 12 shows the mesh statistics for the final mesh created. This mesh size is referred to as fine in the mesh dependency study and is the mesh used for the remainder of the numerical simulation.

Mesh Quality

The final mesh chosen for this study after the mesh refinement study was performed had a total node and element count of 21346 and 20,905 respectively. The maximum skewness of 0.5 in Figure 18 occurs at the area of refinement close to the porous cylinder.

Minimum and maximum values of orthogonal quality were from 0.7147 to 1 in figure 19 and 20.

The aspect ratio in the refined zone reach a maximum of 2.49 and a lower range of 1 in figure 20 and 21.
Figure 17: Full view of the final mesh showing the skewness ranges.

Figure 18: Close-up of the final mesh showing areas of high skewness near the porous cylinder
Figure 19: Full view of the final mesh showing the orthogonal quality ranges.

Figure 20: Close-up of the final mesh showing areas of high orthogonal quality near the porous cylinder
Mesh and Boundary conditions

The computational domain is divided into two main areas. The outer area represents the atmospheric boundary layer while the inner area represents the tree canopy. A schematic view is given in figure 7. The tree canopy consists a single cylindrical body that is face split to attain a structured grid.
Table 3: Boundary Conditions for the flow simulation

<table>
<thead>
<tr>
<th>Named selection</th>
<th>Boundary Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wall</td>
<td>Symmetry</td>
</tr>
<tr>
<td>Porous_media</td>
<td>Wall for solid and fluid for porous flow</td>
</tr>
<tr>
<td>Inlet</td>
<td>Velocity_inlet</td>
</tr>
<tr>
<td>Outlet</td>
<td>Outflow</td>
</tr>
</tbody>
</table>

All the dimensions have been normalized with tree diameter, $D$. The inlet and outlet boundary is positioned 10D upstream and 300D downstream of the tree canopy respectively. Side boundaries are positioned 50D off the tree canopy in the lateral plane. Since the numerical simulation is in 2-D, no top boundary is present or applicable to this simulation.

At the inlet, a uniform inlet velocity is used. The Neumann boundary conditions (pressure change is zero) at the outlet and side boundaries are well-imposed with the selection of the computational domain.

An outflow boundary condition is imposed at the outlet. The outflow condition is used because all the flow variables at the outlet are zero. This is also consistent with the guidelines for atmospheric boundary layer simulations [55, 76, 82-84, 96]. The sides of the computational domain are set to symmetry to represent slip condition and a much better representation of the boundary layer conditions of a tree in open space.

**Fluent Setup**

To setup ANSYS Fluent, double precision was used in a parallel computation setup. For the comparison cases for varying porosity and velocity, parallel computation was used with 5 cores.
The viscous model used throughout this study was the realizable $k$-$\varepsilon$ turbulent model.

In the flow around a solid, the cell zone of the porous cylinder was changed to a solid with no roughness height and a no-slip wall condition. For the porous flow cases, the cell zone was set to fluid. The porous zone was enabled and an inertial resistance coefficient was set.

The incoming flow condition via the boundary conditions was set to 0.56 m/s for the test and initial model comparison cases with the following boundary conditions in Table 3. This was similar to the incoming friction velocity used in wind tunnel measurements and 3-D numerical simulations by [72] and [67] respectively.

**Solution Methods**

SIMPLE spatial discretization was used with first order upwind for both the turbulent kinetic energy and the turbulent dissipation rate. The controls were set to specific values to relax the solution variables as shown in table 5. The calculation was run for 3000 iterations adding options for additional data file quantities for skin friction coefficient, turbulence intensity, vorticity magnitude, production of $k$, turbulent viscosity, wall ystar, wall yplus, x-wall shear stress, y-wall shear stress and wall shear stress.

Furthermore, the superficial velocity formulation is used with the reference cell set to the fluid domain. The reference values for the temperature and pressure are modified based on an average tree height of 45 ft based on the barometric formula.

\[
P = P_{atm} (1 - 2.25588 \times 10^{-5} \times h)^{5.2588}
\]  \hspace{1cm} (26)

\[
T = T_{atm} - 6.5 \frac{K}{km} \times (h)
\]  \hspace{1cm} (27)
The steady-state RANS solutions are obtained using the convergence criterion for all flow properties set to 1e-6 for all wind and porosity conditions. The solution methods applied are summarized in table 4.

*Table 4: Solution Methods from ANSYS Fluent*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheme</td>
<td>SIMPLE</td>
</tr>
<tr>
<td>Gradient</td>
<td>Least Squares Cell Based</td>
</tr>
<tr>
<td>Pressure</td>
<td>Standard</td>
</tr>
<tr>
<td>Momentum</td>
<td>First Order Upwind</td>
</tr>
<tr>
<td>Turbulent Kinetic Energy</td>
<td>First Order Upwind</td>
</tr>
<tr>
<td>Turbulent Dissipation Rate</td>
<td>First Order Upwind</td>
</tr>
</tbody>
</table>

To minimize numerical diffusion, second order upwind should be selected for all momentum and turbulent parameters. However, at high $c_i$ levels, this leads to a divergent solution with oscillating flow variables.

*Table 5: Simulation controls setting Under-Relaxation Factors*

<table>
<thead>
<tr>
<th>Controls</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure</td>
<td>0.3</td>
</tr>
<tr>
<td>Density</td>
<td>1</td>
</tr>
<tr>
<td>Body Forces</td>
<td>1</td>
</tr>
<tr>
<td>Momentum</td>
<td>0.6</td>
</tr>
<tr>
<td>Turbulent Kinetic Energy</td>
<td>0.8</td>
</tr>
<tr>
<td>Turbulent Dissipation Rate</td>
<td>0.7</td>
</tr>
<tr>
<td>Turbulent Viscosity</td>
<td>1</td>
</tr>
</tbody>
</table>
Determination of test points

A Latin square was used to determine the variables to consider and run. Without considering the case for the solid (impermeable) cylinder, a total of 36 runs were conducted. Table 6 shows the final Latin square varying the velocity and porosity parameters.

Table 6: Latin square of parameters for numerical simulation

<table>
<thead>
<tr>
<th>VELOCITY (m/s)</th>
<th>15</th>
<th>75</th>
<th>150</th>
<th>300</th>
<th>750</th>
<th>1500</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>R1</td>
<td>R7</td>
<td>R13</td>
<td>R19</td>
<td>R25</td>
<td>R31</td>
</tr>
<tr>
<td>5</td>
<td>R2</td>
<td>R8</td>
<td>R14</td>
<td>R20</td>
<td>R26</td>
<td>R32</td>
</tr>
<tr>
<td>7.5</td>
<td>R3</td>
<td>R9</td>
<td>R15</td>
<td>R21</td>
<td>R27</td>
<td>R33</td>
</tr>
<tr>
<td>10</td>
<td>R4</td>
<td>R10</td>
<td>R16</td>
<td>R22</td>
<td>R28</td>
<td>R34</td>
</tr>
<tr>
<td>12.5</td>
<td>R5</td>
<td>R11</td>
<td>R17</td>
<td>R23</td>
<td>R29</td>
<td>R35</td>
</tr>
<tr>
<td>15</td>
<td>R6</td>
<td>R12</td>
<td>R18</td>
<td>R24</td>
<td>R30</td>
<td>R36</td>
</tr>
</tbody>
</table>

Velocity range is from 2.5 m/s to 15 m/s, porosity (resistance coefficient, $c_i$) ranges from 15 m$^{-1}$ to 1500 m$^{-1}$. 
Chapter 5 - Results and Discussion

Numerical simulation results

Domain Dependency (DD)

To better understand the blockage effects of the porous zone and the fluid domain on the solution. A domain dependency test was performed at three inertial resistance coefficient values (15, 150 and 1500 m$^{-1}$) at 0.56 m/s inlet velocity and default turbulence intensity (5 %). During this test, best practice guidelines were taken from [35-37] for simulating flows around solid cylinders. This entailed placing the upstream, downstream and lateral boundaries at a minimum of 5D, 15D and 40D away from the area of interest.

The domain sizes; 1.9 m by 0.6 m, 4 m by 2 m and 10 m by 2 m showed no variation in the solution variables for the low $c_i$ cases.

Turbulence Intensity Dependency (TID)

As indicated earlier in the previous section, the flow of interest is in the high Reynolds number regime ($Re > 10^4$). Air flows of this nature are generally turbulent and need to be defined.

A test of four turbulence intensity values were compared with constant inertial resistance coefficient, $c_i = 15$ m$^{-1}$ and velocity $U_\infty = 0.56$ m/s. The results of this test show that lower turbulence intensities reduce the onset of the far wake ($x/D \geq 10D$) recovery. This confirms the approach taken by wind tunnel experiments by [67] and [72] who performed wind tunnel and flume experiments around porous media with turbulence intensities lower than 0.8%.
**Mesh Dependency (MD)**

The computations made for the mesh dependency were performed based on simulation techniques and best practices outlined from the works in [65, 79] and have been discussed in the previous chapter of this paper.

The porous cylinder was set to fluid in order to use the porous zone formulation [33]. The resistance coefficients were set to 0 [m^{-2}] for the viscous resistance and 15/[m^{-1}] for the inertial resistance. This was to model the flow over a lightly dense canopy in the turbulence flow regime. As presented in the previous section, the choice of viscous and inertial resistance is based on the nonlinear behavior of the Darcy’s law in this region.

In refining the mesh, the regions surrounding the porous cylinder was refined using gradual increments and element divisions based on the diameter, D, as the normalizing length. The mesh contained a minimum element size, h = 0.0025 m. The computational domain was 4 m length by 1 m width. The canopy diameter, D = 0.1 m.

Mesh dependency study was conducted for the numerical simulation for 4 sizes of mesh from coarse to fine. The mesh contains a rectangular area of refinement in the area of interest for capturing the details of the flow. The fine mesh was chosen because the difference in the mean velocity and the pressure coefficient is less than 10 percent difference between the fine and very fine formulation.
After the mesh dependency test was performed, the fine mesh was used for all other simulations with maximum skewness, orthogonal quality and aspect ratio of 0.5, 1 and 2.499 respectively. This mesh had 90,505 elements and 273,056 nodes. The statistics for the different mesh formulations are given in table 7.

Table 7: Mesh dependency test of six mesh sizes.

<table>
<thead>
<tr>
<th>Mesh Size</th>
<th>Nodes</th>
<th>Elements</th>
<th>Velocity at 3000mm (m/s)</th>
<th>Pressure Coefficient at 3000mm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coarse</td>
<td>63708</td>
<td>22645</td>
<td>3.561</td>
<td>-1.040</td>
</tr>
<tr>
<td>Medium</td>
<td>273056</td>
<td>90505</td>
<td>3.591</td>
<td>-0.042</td>
</tr>
<tr>
<td>Fine</td>
<td>1090101</td>
<td>362340</td>
<td>4.627</td>
<td>-0.40</td>
</tr>
<tr>
<td>Fine X2</td>
<td>1166884</td>
<td>387869</td>
<td>4.878</td>
<td>0.11</td>
</tr>
</tbody>
</table>

The blue rectangle in figure 20 shows the position of the tree in the flow domain. The results from the simulation runs converge monotonically and are shown in the Appendix A.
Solid vs. porous formulation

Figure 24 and 25 show the results for the simulation for the flow around a solid cylinder versus the porous cylinder.

The contour plot of the velocity along the x-direction given in figure 23. We can see that the flow resolves for the flow conditions given. This is expected for the solid that has the recirculation zones right behind the object and possesses a wake that is symmetrical about the y-axis. This flow behaviour has been well researched and agree with flow visualization results in literature.

![Contour plot of streamwise horizontal velocity around solid cylinder.](image)

The porous flow formulation was used in this setup to ascertain the difference in the velocity speed-up regions and pressure distribution. As expected, the wake flow does not resolve and agrees with
the results from research work conducted in [67]. Additionally, there is a tendency for the flow to go around the porous wall versus through the canopy in most cases.

![Figure 25: Contours of streamwise horizontal velocity for porous zone at $c_i = 15 m^4$](image)

Previous research showed that the flow in a rectangular domain whether 2-D or 3-D was conducted using two different types of boundary condition for the lateral (or side) walls. An investigation into this showed that a reduction in the velocity is observed which is due to the wall interaction. Figure 26 below shows a comparison of the velocity along the horizontal direction of flow with three porosity variations.

It was decided based on the representation of the atmospheric boundary layer (ABL) characteristics of canopy flows that symmetry boundary condition would best represent the realistic nature of the flow. Although in cases where wind tunnel or water channel experiment comparison is needed, wall boundary conditions would be appropriate.
Figure 26: Comparison of wall and symmetry boundary condition for three resistance coefficient values.

To investigate the flow behavior though the porous zone; its wake characterization, the locations of the speed-up regions, parameter variation was conducted. For the 2-D simulation, the fine mesh with extended boundaries was used. Several runs with coefficient of inertial resistance, $c_i$ ranging from 15 to 1500 m$^{-1}$ was conducted.

**Porosity variation**

Following the DD, MD and TID tests, the porosity was varied from lightly dense to highly dense canopies. In the numerical simulation, the porosity varied proportionally to the recovery velocity of porous cylinder.

At $c_i = 15$ m$^{-1}$, the velocity drops to $0.53u_\infty$ and gradually recovers to $0.78u_\infty$. From figure 67, 80 to 82, 104 to 106, 125 to 127, 146 to 148, 167 to 169, and 188 to 190; the contour plots of velocity indicate the maximum velocity attained are around the sides (at approx. 82 °) of the
cylinder and reach $1.03u_\infty$. The onset of turbulence in the wake region enables the flow to recover, albeit gradually.

At $c_i = 75$ m$^{-1}$, the velocity drops to $0.05u_\infty$ and gradually recovers to $0.71u_\infty$. From figure 68, 89 to 91, 110 to 112, 131 to 133, 152 to 154, 173 to 175, and 194 to 196; the contour plots of velocity indicate the maximum velocity attained are around the sides of the cylinder and reach $1.099u_\infty$.

At $c_i = 150$ m$^{-1}$, the velocity drops to $-0.02u_\infty$ and gradually recovers to $0.79u_\infty$. From figure 69, the contour plots of velocity indicate the maximum velocity attained are around the sides of the cylinder and reach $1.145u_\infty$.

At $c_i = 300$ m$^{-1}$, the velocity drops to $-0.08u_\infty$ and gradually recovers to $0.82u_\infty$. From figure 70, 92 to 94, 113 to 115, 134 to 136, 155 to 157, 176 to 178, and 197 to 199; the contour plots of velocity indicate the maximum velocity attained are around the sides of the cylinder and reach $1.174u_\infty$. The centerline velocity fluctuates before leveling at the far stream value. The contour plots indicate the onset of high turbulent mixing that is evident in the presence of isolated eddies in the flow.

At $c_i = 750$ m$^{-1}$, the velocity drops to $0.58u_\infty$ and gradually recovers to $0.78u_\infty$. From figure 95 to 97, 116 to 118, 137 to 139, 158 to 160, 179 to 181, and 200 to 202; the contour plots of velocity indicate the maximum velocity attained are around the sides of the cylinder and reach $1.332u_\infty$. The presence and location of isolated eddies in the flow change with each run, and the turbulence characteristics are maintained.

At $c_i = 1500$ m$^{-1}$, the velocity drops to $-0.15u_\infty$ and fluctuates as it recovers to $0.97u_\infty$. From figure 71, 98 to 100, 119 to 121, 140 to 142, 161 to 163, 182 to 184, and 203 to 205; the contour plots of velocity indicate the maximum velocity attained are around the sides of the cylinder and reach
1.434u_\infty. The flow through the canopy continues to maintain its oscillating TKE peak points caused by the speed up of the flow as it exits the canopy.

For lower resistance coefficients (15 to 150 m\(^{-1}\)), there is a lower drop in the incoming velocity and rapid recovery of the velocity magnitude.

For higher resistance coefficients (300 to 1500 m\(^{-1}\)), there is a higher drop in the velocity magnitude and delayed recovery of the velocity magnitude. There is a presence of near and far wake due to the flow in the lee of the canopy.

A special case for D = 0.01 m was run at the same free stream velocity with the current domain size. The results shown in figure 27 show a more stable behavior than in smaller domain sizes. There is a 5 to 25\% drop in the far wake of the tree for variation of the resistance coefficients from 15 m\(^{-1}\) to 1500 m\(^{-1}\).

![Figure 27: Variation of velocity along flow direction (the x-direction) with different inertial resistance coefficients.](image_url)
Figure 28: Variation of velocity along the flow direction (the x-direction) at height of 0.5h with different inertial resistance coefficients. [67]

A comparison was also made for the pressure distribution along the centerline. This result shows agreement with the work by [67]. To do this, a MATLAB script was utilized to grab the points from an image. The MATLAB code is provided in Appendix D (grabit.m).

Following the simulation, the effect of the porosity (resistance coefficient) indicate the location of stable asymptotic behavior as the porosity is increased. Figure 29 below shows this effect at four location, x/D = 5.11, 10, 15.02 and 20. The result show the velocity reduction behavior of the canopy tapers at about 300 m$^{-1}$. 
Figure 29: Effect of resistance coefficient on the wake velocity at $x/D = 5.11, 10, 15.02, 20$

By varying the porosity of a canopy, a reduction in the onset of turbulence intensity and momentum of air is observed. The wake deficit behavior behaves asymptotically stable as the resistant coefficient increases. The assumption is being made for only the stable wake region of the flow. This means that plants with high foliage density would behave similar and canopies with lower foliage density have linear velocity reduction rate.

Figures 30 through 50 provide the results of the streamline and velocity vector plots for the numerical simulation varying the inertial resistance coefficients and the solid cylinder.

The streamline and vector plots at $c_i = 15 \text{ m}^{-1}$ to $150 \text{ m}^{-1}$ indicate a drop in the velocity in the lee of the canopy with no recirculation zones in the flow. In each case, the velocity gradually recovers but not to an asymptotically stable value. At these $c_i$ levels, the canopy extracts momentum from the flow but still allows most of the fluid flowing the domain to pass through. The presence of the speed-up region along the sides of the tree canopy is a driving factor for the increase in turbulence properties further downstream in the lee of the flow.
At higher $c_i$’s (300 m$^{-1}$ to 1500 m$^{-1}$), there are recirculation regions present in the flow at different locations due to both turbulent mixing and the pressure exchange in the flow. Most fluid particles flowing into the porous canopy exit sideways and some make it through the entire canopy length signaling resistance. Upon exit, these fluid particles have a higher intensity than the surrounding flow and at forces and rapid onset of turbulent mixing.

*Figure 30: Velocity streamlines for porous flow at $c_i = 15$ m$^{-1}$*

*Figure 31: Velocity vector plot for porous flow at $c_i = 15$ m$^{-1}$*
Figure 32: Closeup view of velocity vectors for porous flow at \( c_i = 15 \text{ m}^{-1} \)

Figure 33: Velocity streamline plot for porous flow at \( c_i = 75 \text{ m}^{-1} \)
Figure 34: Velocity vector plot for porous flow at $c_i = 75 \text{ m}^{-1}$

Figure 35: Closeup view of velocity vectors for porous flow at $c_i = 75 \text{ m}^{-1}$
Figure 36: Velocity streamlines for porous flow at $c_i = 150 \text{ m}^{-1}$

Figure 37: Velocity vector plot for porous flow at $c_i = 150 \text{ m}^{-1}$
Figure 38: Closeup view of velocity vectors for porous flow at $c_i = 150 \text{ m}^{-1}$

Observing the flow through the porous canopy at 300 m$^{-1}$, there are two recirculation regions spaces asymmetrically in the lee. At 750 m$^{-1}$, air flowing into the canopy begins to flow out at the sides of the canopy experiences an adverse pressure gradient combined with momentum sink term. The speed-up regions above the canopy extend over the top of the right leading to the onset of recirculation in the flow. A similar behavior was observed at 1500 m$^{-1}$, however, only one isolated recirculation region further downstream was observed.

The results of the streamline and vector field plot for the solid cylinder indicates two recirculation regions symmetric about the $y$-axis and the speed up regions along the top and bottom of the cylinder.
Figure 39: Velocity streamlines for porous flow at $c_i = 300 \text{ m}^{-1}$

Figure 40: Velocity vector plot for porous flow at $c_i = 300 \text{ m}^{-1}$
Figure 41: Closeup view of velocity vectors for porous flow at $c_i = 300 \text{ m}^{-1}$

Figure 42: Velocity streamline plot for porous flow at $c_i = 750 \text{ m}^{-1}$
Figure 43: Velocity vector plot for porous flow at $c_i = 750 \text{ m}^{-1}$

Figure 44: Closeup view of velocity vectors for porous flow at $c_i = 750 \text{ m}^{-1}$
Figure 45: Velocity streamline plot for porous flow at $c_i = 1500 \text{ m}^{-1}$

Figure 46: Velocity vector plot for porous flow at $c_i = 1500 \text{ m}^{-1}$
Figure 47: Closeup view of velocity vectors for porous flow at \( c_i = 1500 \text{ m}^{-1} \)

Figure 48: Velocity streamline plot for flow around a solid cylinder
Velocity variation

The flow around a porous cylinder was varied for velocities ranging for the three priorities presented in Chapter 4; the range of velocities experienced on trees, operation range of VAWTs and numerical to experimental similarity conditions.
The simulation results below show the effect of varying the incoming velocity on the flow around the porous cylinder. Figures 51 below indicates that there is no change in the streamwise velocity when the incoming velocity of 2.5 m/s for $c_i = 15 \text{ m}^{-1}$.

Figure 51: Variation of velocity along flow direction at all initial velocities for $c_i = 15 \text{ m}^{-1}$

Figure 52: Variation of velocity along flow direction at all initial velocities for $c_i = 75 \text{ m}^{-1}$
Figure 52 shows that there is minimal variation in the centerline velocity at the far wake region \((x/D \geq 5D)\) for \(c_i = 75 \text{ m}^{-1}\) when the input velocity is changed. The lowest far-wake velocity occurs when \(U_\infty = 2.5 \text{ m/s}\).

\[\text{Figure 53: Variation of velocity along flow direction at all initial velocities for } c_i = 150 \text{ m}^{-1}\]

At \(c_i = 150 \text{ m}^{-1}\) (figure 53), there is a slight variation in the velocity recovery path between \(x/D = 5D \text{ to } 22D\) in the lee, however, at all input velocities, the same far wake velocity is observed.

Cases with the \(c_i = 300 \text{ m}^{-1}\) (figure 54) indicate an oscillating centerline velocity as it recovers. The velocity recovery path is different or all input velocities, however, they all asymptotically stabilize at \(0.82u_\infty\).
Figure 54: Variation of velocity along flow direction at all initial velocities for $c_i = 300 \text{ m}^{-1}$

Figure 55 and 56 show the results of the velocity variation at $c_i = 750$ and $1500 \text{ m}^{-1}$ respectively. The velocity drop and recovery is characterized by oscillating motion that hovers around the freestream velocity (in $c_i = 750 \text{ m}^{-1}$) and below freestream ($c_i = 1500 \text{ m}^{-1}$).

Figure 55: Variation of velocity along flow direction at all initial velocities for $c_i = 750 \text{ m}^{-1}$
Figure 56: Variation of velocity along flow direction at all initial velocities for $c_1 = 1500 \text{ m}^{-1}$

Following this, the variation of velocity did not have the acclaimed effect on all porosities that was predicted from literature mainly because it is also in the turbulent regime and the level of assumptions made about the flow.

The effect of varying velocity should be the same over a porous cylinder experimentally validated.

Figure 57 through 62 show the pressure coefficient variation along the centerline of the porous cylinder at velocities 2.5, 5, 7.5, 10, 12.5 and 15 m/s respectively. It is observed that pressure coefficient magnitude increases as the resistance coefficient increases. In the lee of the cylinder, the pressure exerted as a result of the bleeding flow gradually returns to an asymptotically stable value at 20D upstream. This behavior is only observed at $c_1 = 15 \text{ m}^{-1}$ and 75 m$^{-1}$. At $c_1 = 150 \text{ m}^{-1}$ and 300 m$^{-1}$, the pressure in the lee of the cylinder begins to experience a favorable pressure gradient indicated by an increase in the pressure that unchanging for a distance of 5D and onwards return to freestream values.
Figure 57: Pressure coefficient comparison along centerline at velocity = 2.5 m/s

Figure 58: Pressure coefficient comparison along centerline at velocity = 5 m/s
Figure 59: Pressure coefficient comparison along centerline at velocity = 7.5 m/s

Figure 60: Pressure coefficient comparison along centerline at velocity = 10 m/s
The pressure coefficient at $c_i = 750 \text{ m}^{-1}$ and $1500 \text{ m}^{-1}$ is characterized by three bottom peaks (or troughs) in the lee of the cylinder and eventually recovering to an asymptotically stable value. It is worth noting that the pressure coefficient at $c_i = 75 \text{ m}^{-1}$ has a maximum pressure coefficient that is comparable to a solid cylinder.
**Comparison with literature**

The numerical results obtained from the simulation provides results follow a similar trend with the results from works by [45, 20]. Vertical slices along the flow domain were taken at six locations in the lee of the canopy and compared for the porosity cases with \( D = 0.1 \) m \((x/D = 1, 2, 3, 4, 5\) and 10) and a special case of \( D = 0.01 \) m \((x/D = 10, 20, 30, 40, 50\) and 100),

![Comparison of x-velocity profiles along the wake of a tree from simulation (left) with figure 3 [20] (3-D flow around a rectangular cylinder, right).](image)

*Figure 63: Comparison of x-velocity profiles along the wake of a tree from simulation (left) with figure 3 [20] (3-D flow around a rectangular cylinder, right).*

From the comparison of the turbulent kinetic energy in the lee of the flow with the results from [20], Vertical profiles of the longitudinal and streamwise velocity, turbulent kinetic energy and intensity was obtained and normalized.

Closer to the lee of the canopy, the turbulent intensity is represented by a high spike and drop. The magnitude of the normalized turbulent intensity is similar in trend to that calculated from [20] at \( D = 0.01 \) m.

Care was taken in this comparison since the previous authors mentioned simulated the flow around trees represented as rectangular blocks and the porous formulation used by other authors.
Figure 63 provides a comparison of the velocity wake profile of the canopy simulation in this study compared to the 3-D flow around a rectangular cylinder in [20]. The results show that the velocity drops as the flow travels along the canopy’s lee. Howbeit, the formation and digestion of big eddies by small eddies (enstrophy) in the flow generates these differences.

The peaks of the closest layers of turbulent intensity is under predicted by my simulation and has a 40 percent difference with [20].

![Figure 64: TKE profiles comparison with figure 4 [20].](image)

When the geometry dimensions are modified resulting in a smaller blockage ratio and extended region, the trend in the TKE that is similar to the results obtained around a 3-D rectangular porous cylinder (figure 64).
Figure 65: TI profiles along the wake of canopy at $c_i = 15 \text{ m}^4$ and 0.56 m/s for $D = 0.01 \text{ m}$

Figure 65 show the TI profiles for $D = 0.01 \text{ m}$, this is evident by two peaks ($TI = 0.15$) symmetric about the y-axis centerline.
Figure 66: Vertical TKE profiles at all c; levels (a = 15 m$^{-1}$, b = 75 m$^{-1}$, c = 150 m$^{-1}$, d = 300 m$^{-1}$, e = 750 m$^{-1}$, f = 1500 m$^{-1}$)
Figure 66 provides the TKE result for the variation of all porosity levels with D = 0.1 m. The results of the TKE profiles indicate two peaks with increasing turbulence intensity along regions close to the cylinder (1D laterally). Marching along the positive x-direction the turbulence intensity peaks increase. This is only observed in $c_i = 15$ and 75 m$^{-1}$. For the remainder of the porosity cases, there is an asymmetrical turbulence peak coupled with a gaussian distributed TKE (at $x/D = 20$).

A comparison with the streamwise velocity with the experimental and numerical investigations from [72] and [67] respectively shows close correlation at 15 m$^{-1}$ to 150 m$^{-1}$. Between the experimental results obtained from [72], the region where there is a decrease in pressure is identical to the results with a 20 percent difference.

After the simulation runs were concluded, comparison with the results from [67] for each porosity level showed a lower decrease in the streamwise velocity and a lower estimate of the far-wake velocity. At 10D away from the porous cylinder, there is a difference of 13.33 percent for $c_i = 15$ m$^{-1}$ and the lowest velocity drop difference of 13.11 percent. At $c_i = 15$ m$^{-1}$, there is a 17.65 percent difference in the far wake velocity and 55 percent difference in the lowest velocity. At $c_i$ levels higher than this, the velocity drop difference in much more pronounced.

The far wake behavior at $c_i = 300$ m$^{-1}$ gives the closest agreement with a difference of 3.95 percent, however, its velocity drop shows the largest difference. It is worth noting that the experimental results from [67] are much closer to the results obtained in this study.
Table 8: Error comparison of numerical simulation with [67]

<table>
<thead>
<tr>
<th>Porosity (1/m)</th>
<th>Far wake diff (%)</th>
<th>Lowest velocity drop diff (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>13.33</td>
<td>13.11</td>
</tr>
<tr>
<td>75</td>
<td>17.65</td>
<td>85.19</td>
</tr>
<tr>
<td>150</td>
<td>15.00</td>
<td>111.76</td>
</tr>
<tr>
<td>300</td>
<td>3.95</td>
<td>260.00</td>
</tr>
<tr>
<td>1500</td>
<td>-27.16</td>
<td>-50.00</td>
</tr>
</tbody>
</table>

Table 8 shows the error comparison results with the numerical simulation results from [67]. At porosity levels greater than 150 m\(^{-1}\), the porous cylinder begins to behave as a solid cylinder because the porous cylinder limits fluid flow through it and when it passes through, its behavior mimics that of a solid cylinder flow.

Additionally, at higher porosity levels, there is a presence of vortex shedding (generally a 3-D turbulence phenomenon, see Chapter 2) which indicates an inherent transient turbulent flow which cannot be visualized in a 2-D simulation.

*Figure 67: Velocity comparison at 15 m\(^{-1}\) with [67]*
Figure 68: Velocity comparison at 75 m$^{-1}$ with [67]

Figure 69: Velocity comparison at 150 m$^{-1}$ with [67]
Figure 70: Velocity comparison at 300 m\(^{-1}\) with [67]

Figure 71: Velocity comparison at 1500 m\(^{-1}\) with [67]
Discussion

This section presents relevant discussion on the numerical and experimental results and how these results are applied to potential flow theory.

The results of the numerical simulation show strong trend agreement with literature. However, there is an under-prediction of the wake velocity and the recovery path which is due to the underlying assumptions made in this model.

To adapt the results of the velocity deficit to potential flow, the velocity vector flow field is compared to obtain from potential flow. The source and sink strengths for each resistance coefficients and its ratio can then be applied to improve flow field estimation of the de-singularized panel method.

**Utilization of LRB to estimate Porous flow velocity field**

The source and sink strength ratios can be obtained from a curve fit of the sink spacing and calculated based on the actuator disk theory and potential flow theory.

Using the equations derived for the source/sink strength ratio is used to obtain the optimum sink spacing ($ss_{opt}$), a combined technique of trial and error and iterative forward marching approach is used in selecting the residual at $U_4$ and variance along the centerline velocity.

It can be shown (see Appendix E) that the $ss_{opt}$ is a function of the source strength, sink strength, porous cylinder diameter, freestream and wake velocity.

$$ss_{opt} = f(m_{so}, m_{so}, U_1, U_4, D)$$

Selecting the source/sink strength ratios and sink spacing that best represents the velocity field is the goal of this approximation.
Figure 72: Simulated x-velocity \((c_i = 15 \text{ m}^{-1})\) versus centerline velocity from potential flow

Figure 73: Streamline plot of stream function solution using potential flow

Figure 72 shows the centerline velocity plots from the slow simulation at \(c_i = 15\text{m}^{-1}\) versus the potential flow solution.
Figure 74: Vector field of potential flow simulation

Figure 73 and 74 show the streamline and velocity vector plots respectively in potential flow. Figure 73 shows that the resulting potential flow is a net sink flow. In figure 74, the affected region of the source/sink pair is the voided region. The sink spacing that gives the lowest variance value is used.
Sink spacing versus variance was plotted to determine for each case, the optimum spacing. It was found at $\sigma^2 = 0.123$ in figure 75.

The procedure outlined in this section was performed for all porosity cases. Table 9 shows the source and sink strengths and their ratio.

For all levels of porosity, the source/sink strength ratio is constant. However, the reader is reminded that this is an initial guess that should be corrected based on required fidelity of computations.
Chapter 6 - Conclusion

Trees occur in the environment in different shapes and height configurations. The porous media formulation was successfully able to model the flow behavior of the air in accordance with previous numerical and experimental results using the ANSYS Fluent CFD package.

Because of the complexity of the flow around vegetative structures, the tree was modeled as a single porous fluid region with resistance coefficients accounting for the porosity. RANS equations were used as the basis for all simulations and validations with turbulence was modeled with the realizable $k-\varepsilon$ model and standard wall functions.

Since the previous work by researchers investigated the 3-D nature of the flow, an assumption was made to ignore the vertical component of the velocity. It is important to note that this vertical component of the velocity is mainly responsible for the rapid reattachment of the wake due to the enstrophy. Thus, the results of the normalized 2-D flow properties when compared followed a similar trend to the 3-D case but was underpredicted.

In this study, mesh, domain and turbulence intensity dependency test was conducted to provide solutions that are insensitive to variations in input parameters such as the Reynolds number and porosity. The mass imbalance and residuals were monitored. The residuals converge monotonically for cases with low resistance coefficients (15 to 75 m$^{-1}$).

The numerical experiment carried out on a tree canopy for variations in porosity (resistance coefficients) and velocity based on geometric similarity show that increasing the Reynolds number of the flow had very little effect on the resulting centerline velocity profile at lower porosities. However, as the velocity is increased; there is a higher drop in velocity in the lee of the canopy.
Increasing the resistance coefficients results in the reduction of the magnitude of the far-wake velocity after the flow recovers downstream. This is consistent with literature findings; however, this velocity reduction does not apply for high $c_i$ levels (300 to 1500 m$^{-1}$) because the porous canopy behaves more as a solid deflecting much of the airflow around it. As such, the speed-up regions are present around the sides of the tree canopy and its affected region grows with increasing resistance coefficient.

The attempt to validate the numerical results via wind tunnel experiment did not provide enough information about the velocity field due to measurement limitations and inadequate model preparation. The overall reliability of the experimental results presented here are somewhat satisfactory, but not good enough. To improve the reliability, further work should be done in a sediment (water) channel to obtain the turbulence properties as well as create the ABL conditions not available in a wind tunnel.

The maximum discrepancy between literature and the simulations ranged from 3-160% depending on input parameters considered. For cases with low $c_i$, a 2-D simulation is sufficient to estimate the flow field around a single canopy. Calculations using the actuator disk theory, allow the results from these numerical simulations to optimize potential flow codes where the trees are at or above the height of buildings being modeled. For flows around single tree canopies, the source to sink strength ratio for most porosities is 1:1.44.

The results presented in this thesis is valid for numerical simulation of trees, coniferous and deciduous that have a cylindrical canopy cross section. The velocity profiles estimated herein apply from the mid-section to the tree top of the canopy where the largest near-wake distance is observed.
In summary, it has been demonstrated in this thesis that accurate 2-D CFD can model the important flow field properties for vegetative structures with the proper choice of mesh density, boundary conditions and turbulence model.

**Further work/Recommendations**

This alternate plan paper will require further verification of the data provided using experimental means preferably one that does not have flow disturbance issues such as LDV or particle image velocimetry (PIV).

Further refinement of the adaptation to potential flow can be obtained as well to investigation of the effects of multiple trees and with buildings in the same domain.

Consequent numerical simulation of the built environment for the optimization of placement of VAWTs should include tree canopies as a porous zone for increasing the accuracy of flow simulations with field results. Whether in 2-D or 3-D CFD, the effects of trees on the turbulence properties can be better predicted.

When there is a need to place VAWTs in the vicinity of a tree. It is recommended that placement around canopies with high foliage density should be avoided and within 10D of canopies of low foliage density. Placement should be in the speedup regions to the sides of the canopy. Finally, how close one should place the VAWT around vegetation should rely on sound engineering decision with pre-concluded wind flow simulations.
References


[38] Lecture 9: Best Practice Guidelines, 16.0 Release, ANSYS


Berlin Heidelberg 2017. Pp 29-49


[103] Wu, Ming-Hsun, Wen, Chih-Yung, Yen, Ruey-Hor, Weng, Ming-Cheng and Wang, An-
Bang. Experimental and numerical study of the separation angle for flow around a circular
Appendix A

Reduction of Navier-Stokes equation

In the x-component we have,

\[ \rho \frac{Du}{Dt} = - \frac{\partial p}{\partial x} \]

\[ \rho \frac{\partial u}{\partial t} + \rho u \frac{\partial u}{\partial x} + \rho v \frac{\partial u}{\partial y} + \rho w \frac{\partial u}{\partial z} = - \frac{\partial p}{\partial x} \]

Assuming steady flow;

\[ \frac{\partial u}{\partial t} = 0 \]

\[ u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} = - \frac{1}{\rho} \frac{\partial p}{\partial x} \]

Multiply through by dx

\[ u \frac{\partial u}{\partial x} dx + v \frac{\partial u}{\partial y} dx + w \frac{\partial u}{\partial z} dx = - \frac{1}{\rho} \frac{\partial p}{\partial x} dx \]

From equations along a streamline, following Newtons conservation laws, we have:

\[ udz - wdx = 0 \]

\[ udy - vdx = 0 \]

Substituting into above equation we have

\[ u \frac{\partial u}{\partial x} dx + u \frac{dy}{dx} \frac{\partial u}{\partial y} dx + u \frac{dz}{dx} \frac{\partial u}{\partial z} dx = - \frac{1}{\rho} \frac{\partial p}{\partial x} dx \]
\[ u \left( \frac{\partial u}{\partial x} \, dx + \frac{\partial u}{\partial y} \, dy + \frac{\partial u}{\partial z} \, dz \right) = -\frac{1}{\rho} \frac{\partial p}{\partial x} \, dx \]

\[ u \, du = -\frac{1}{\rho} \frac{\partial p}{\partial x} \, dx \]

Or

\[ \frac{1}{2} d(u^2) = -\frac{1}{\rho} \frac{\partial p}{\partial x} \, dx \quad (A.1) \]

In the y-component we have,

\[ \rho \frac{Dv}{Dt} = -\frac{\partial p}{\partial y} \]

\[ \rho \frac{\partial v}{\partial t} + \rho u \frac{\partial v}{\partial x} + \rho v \frac{\partial v}{\partial y} + \rho w \frac{\partial v}{\partial z} = -\frac{\partial p}{\partial y} \]

Assuming steady flow;

\[ \frac{\partial v}{\partial t} = 0 \]

\[ u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + w \frac{\partial v}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial y} \]

Multiply through by \( dy \)

\[ u \frac{\partial v}{\partial x} \, dy + v \frac{\partial v}{\partial y} \, dy + w \frac{\partial v}{\partial z} \, dy = -\frac{1}{\rho} \frac{\partial p}{\partial y} \, dy \]

From equations along a streamline
\[ v dz - w dy = 0 \]

\[ u dy - v dx = 0 \]

Substituting into above equation we have

\[
v \frac{dx}{dy} \frac{\partial v}{\partial x} dy + v \frac{\partial v}{\partial y} dy + v \frac{dz}{dy} \frac{\partial u}{\partial z} dy = -\frac{1}{\rho} \frac{\partial p}{\partial y} dy
\]

\[
v \left( \frac{\partial v}{\partial x} dx + \frac{\partial v}{\partial y} dy + \frac{\partial v}{\partial z} dz \right) = -\frac{1}{\rho} \frac{\partial p}{\partial y} dy
\]

\[ v \ dv = -\frac{1}{\rho} \frac{\partial p}{\partial y} dy \]

Or

\[
\frac{1}{2} d(v^2) = -\frac{1}{\rho} \frac{\partial p}{\partial y} dy \quad \text{(A.2)}
\]

In the z-component we have,

\[
\rho \frac{Dw}{Dt} = -\frac{\partial p}{\partial z}
\]

\[
\rho \frac{\partial w}{\partial t} + \rho u \frac{\partial w}{\partial x} + \rho v \frac{\partial w}{\partial y} + \rho w \frac{\partial w}{\partial z} = -\frac{\partial p}{\partial z}
\]

Assuming steady flow;

\[ \frac{\partial w}{\partial t} = 0 \]
\[
\frac{\partial w}{\partial x} + v \frac{\partial w}{\partial y} + w \frac{\partial w}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial z}
\]

Multiply through by \(dz\)

\[
u \frac{\partial w}{\partial x} dz + v \frac{\partial w}{\partial y} dz + w \frac{\partial w}{\partial z} dz = -\frac{1}{\rho} \frac{\partial p}{\partial z} dz\]

From equations along a streamline

\[vdz - wdy = 0\]

\[udz - wdx = 0\]

Substituting into above equation we have

\[w \frac{dx}{dz} \frac{\partial w}{\partial x} dz + w \frac{dy}{dz} \frac{\partial w}{\partial y} dz + w \frac{\partial w}{\partial z} dz = -\frac{1}{\rho} \frac{\partial p}{\partial z} dz\]

\[w \left( \frac{\partial w}{\partial x} dx + \frac{\partial w}{\partial y} dy + \frac{\partial w}{\partial z} dz \right) = -\frac{1}{\rho} \frac{\partial p}{\partial z} dz\]

\[w \, dw = -\frac{1}{\rho} \frac{\partial p}{\partial z} dz\]

Or

\[\frac{1}{2} d(w^2) = -\frac{1}{\rho} \frac{\partial p}{\partial z} dz\]  \hspace{1cm} (A.3)

Combining Eq (A.1) + (A.2) + (A.3), we have
\[ \frac{1}{2} d(u^2 + v^2 + w^2) = -\frac{1}{\rho} \left( \frac{\partial p}{\partial x} dx + \frac{\partial p}{\partial y} dy + \frac{\partial p}{\partial z} dz \right) \]

\[ \frac{1}{2} d(V^2) = -\frac{1}{\rho} dp \]

\[ -\frac{1}{2} \rho V dV = dp \]

The above equation is the Euler equation and it applies to inviscid flow with no body forces. For incompressible flow the \( \rho \) is constant and the equation can be integrated along any two points on a streamline.

**Numerical setup**

Below are extra readouts from the numerical setup in ANSYS Workbench and Fluent.

*Figure 76: Point cloud and plane reference for exporting grid data for potential flow comparison*
Numerical Convergence plots

This section provides the residual convergence plots, report definitions on the area-weighted average of the x-velocity along the centerline and the mass flow imbalance on the inlet and outlet for all the numerical simulations.

Figure 77: Area-weighted average of x-velocity for \( c_i = 15 \, \text{m}^{-1} \) and incoming velocity = 0.56 m/s

Figure 78: Mass flowrate imbalance on inlet vs outlet for \( c_i = 15 \, \text{m}^{-1} \) and incoming velocity= 0.56 m/s
In figure 43-45, the residuals converge monotonically, there is also a net zero mass flow rate imbalance and the $x$-velocity asymptotically approach 0.42 m/s. This indicates that the solution obtained is stable for the number of iterations the solution converged at, which is 625.
Appendix B – Simulation Results of velocity and resistance coefficient variation

This section of the appendix contains the results of the numerical simulation for the thesis. For each simulated run, contour plots of velocity in x-direction, pressure coefficient, turbulence kinetic energy (TKE), vector plot and streamlines were produced.

Table 10: Run time duration and convergence for all simulation runs

<table>
<thead>
<tr>
<th>RUN NO.</th>
<th>VELOCITY</th>
<th>POROSITY</th>
<th>DISCRETIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>2.5</td>
<td>15</td>
<td>1ST</td>
</tr>
<tr>
<td>R2</td>
<td>2.5</td>
<td>75</td>
<td>1st</td>
</tr>
<tr>
<td>R3</td>
<td>2.5</td>
<td>150</td>
<td>1st</td>
</tr>
<tr>
<td>R4</td>
<td>2.5</td>
<td>300</td>
<td>1st</td>
</tr>
<tr>
<td>R5</td>
<td>2.5</td>
<td>750</td>
<td>1st</td>
</tr>
<tr>
<td>R6</td>
<td>2.5</td>
<td>1500</td>
<td>1st</td>
</tr>
<tr>
<td>R7</td>
<td>5</td>
<td>15</td>
<td>1st</td>
</tr>
<tr>
<td>R8</td>
<td>5</td>
<td>75</td>
<td>1st</td>
</tr>
<tr>
<td>R9</td>
<td>5</td>
<td>150</td>
<td>1st</td>
</tr>
<tr>
<td>R10</td>
<td>5</td>
<td>300</td>
<td>1st</td>
</tr>
<tr>
<td>R11</td>
<td>5</td>
<td>750</td>
<td>1st</td>
</tr>
<tr>
<td>R12</td>
<td>5</td>
<td>1500</td>
<td>1st</td>
</tr>
<tr>
<td>R13</td>
<td>7.5</td>
<td>15</td>
<td>1st</td>
</tr>
<tr>
<td>R14</td>
<td>7.5</td>
<td>75</td>
<td>1st</td>
</tr>
<tr>
<td>R15</td>
<td>7.5</td>
<td>150</td>
<td>1st</td>
</tr>
<tr>
<td>R16</td>
<td>7.5</td>
<td>300</td>
<td>1st</td>
</tr>
<tr>
<td>R17</td>
<td>7.5</td>
<td>750</td>
<td>1st</td>
</tr>
<tr>
<td>R18</td>
<td>7.5</td>
<td>1500</td>
<td>1st</td>
</tr>
<tr>
<td>R19</td>
<td>10</td>
<td>15</td>
<td>1st</td>
</tr>
<tr>
<td>R20</td>
<td>10</td>
<td>75</td>
<td>1st</td>
</tr>
<tr>
<td>R21</td>
<td>10</td>
<td>150</td>
<td>1st</td>
</tr>
<tr>
<td>R22</td>
<td>10</td>
<td>300</td>
<td>1st</td>
</tr>
<tr>
<td>R23</td>
<td>10</td>
<td>750</td>
<td>1st</td>
</tr>
<tr>
<td>R24</td>
<td>10</td>
<td>1500</td>
<td>1st</td>
</tr>
<tr>
<td>R25</td>
<td>12.5</td>
<td>15</td>
<td>1st</td>
</tr>
<tr>
<td>R26</td>
<td>12.5</td>
<td>75</td>
<td>1st</td>
</tr>
<tr>
<td>R27</td>
<td>12.5</td>
<td>150</td>
<td>1st</td>
</tr>
<tr>
<td>R28</td>
<td>12.5</td>
<td>300</td>
<td>1st</td>
</tr>
<tr>
<td>R29</td>
<td>12.5</td>
<td>750</td>
<td>1st</td>
</tr>
<tr>
<td>R30</td>
<td>12.5</td>
<td>1500</td>
<td>1st</td>
</tr>
<tr>
<td>R31</td>
<td>15</td>
<td>15</td>
<td>1st</td>
</tr>
<tr>
<td>R32</td>
<td>15</td>
<td>75</td>
<td>1st</td>
</tr>
<tr>
<td>R33</td>
<td>15</td>
<td>150</td>
<td>1st</td>
</tr>
<tr>
<td>R34</td>
<td>15</td>
<td>300</td>
<td>1st</td>
</tr>
<tr>
<td>R35</td>
<td>15</td>
<td>750</td>
<td>1st</td>
</tr>
<tr>
<td>R36</td>
<td>15</td>
<td>1500</td>
<td>1st</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RUN TIME (MINS)</th>
<th>ITER CONVERGE @ #</th>
<th>MESH SIZE</th>
<th>Mass Flow rate (outlet) (kg/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.25</td>
<td>544</td>
<td>EXT</td>
<td>-3.06</td>
</tr>
<tr>
<td>3.00</td>
<td>1176</td>
<td>EXT</td>
<td>-3.06</td>
</tr>
<tr>
<td>7.31</td>
<td>3000</td>
<td>EXT</td>
<td>-3.06</td>
</tr>
<tr>
<td>6.76</td>
<td>3000</td>
<td>EXT</td>
<td>-3.06</td>
</tr>
<tr>
<td>7.08</td>
<td>3000</td>
<td>EXT</td>
<td>-3.06</td>
</tr>
<tr>
<td>7.06</td>
<td>3000</td>
<td>EXT</td>
<td>-3.06</td>
</tr>
<tr>
<td>1.30</td>
<td>535</td>
<td>EXT</td>
<td>-6.125</td>
</tr>
<tr>
<td>2.90</td>
<td>1248</td>
<td>EXT</td>
<td>-6.125</td>
</tr>
<tr>
<td>6.25</td>
<td>3000</td>
<td>EXT</td>
<td>-6.125</td>
</tr>
<tr>
<td>6.76</td>
<td>3000</td>
<td>EXT</td>
<td>-6.125</td>
</tr>
<tr>
<td>7.08</td>
<td>3000</td>
<td>EXT</td>
<td>-6.125</td>
</tr>
<tr>
<td>7.06</td>
<td>3000</td>
<td>EXT</td>
<td>-6.125</td>
</tr>
<tr>
<td>1.28</td>
<td>531</td>
<td>EXT</td>
<td>-9.18</td>
</tr>
<tr>
<td>2.90</td>
<td>1278</td>
<td>EXT</td>
<td>-9.18</td>
</tr>
<tr>
<td>6.25</td>
<td>3000</td>
<td>EXT</td>
<td>-9.18</td>
</tr>
<tr>
<td>6.76</td>
<td>3000</td>
<td>EXT</td>
<td>-9.18</td>
</tr>
<tr>
<td>7.08</td>
<td>3000</td>
<td>EXT</td>
<td>-9.18</td>
</tr>
<tr>
<td>6.80</td>
<td>3000</td>
<td>EXT</td>
<td>-9.18</td>
</tr>
<tr>
<td>6.83</td>
<td>3000</td>
<td>EXT</td>
<td>-9.18</td>
</tr>
<tr>
<td>1.28</td>
<td>531</td>
<td>EXT</td>
<td>-9.18</td>
</tr>
<tr>
<td>2.90</td>
<td>1278</td>
<td>EXT</td>
<td>-9.18</td>
</tr>
<tr>
<td>6.25</td>
<td>3000</td>
<td>EXT</td>
<td>-9.18</td>
</tr>
<tr>
<td>6.76</td>
<td>3000</td>
<td>EXT</td>
<td>-9.18</td>
</tr>
<tr>
<td>7.08</td>
<td>3000</td>
<td>EXT</td>
<td>-9.18</td>
</tr>
<tr>
<td>1.50</td>
<td>532</td>
<td>EXT</td>
<td>-12.25</td>
</tr>
<tr>
<td>3.00</td>
<td>1286</td>
<td>EXT</td>
<td>-12.25</td>
</tr>
<tr>
<td>8.82</td>
<td>3000</td>
<td>EXT</td>
<td>-12.25</td>
</tr>
<tr>
<td>8.13</td>
<td>3000</td>
<td>EXT</td>
<td>-12.25</td>
</tr>
<tr>
<td>7.82</td>
<td>3000</td>
<td>EXT</td>
<td>-12.25</td>
</tr>
<tr>
<td>1.46</td>
<td>528</td>
<td>EXT</td>
<td>-15.32</td>
</tr>
<tr>
<td>7.12</td>
<td>1309</td>
<td>EXT</td>
<td>-15.32</td>
</tr>
<tr>
<td>9.08</td>
<td>3000</td>
<td>EXT</td>
<td>-15.32</td>
</tr>
<tr>
<td>7.50</td>
<td>3000</td>
<td>EXT</td>
<td>-15.32</td>
</tr>
<tr>
<td>8.90</td>
<td>3000</td>
<td>EXT</td>
<td>-15.32</td>
</tr>
<tr>
<td>9.50</td>
<td>3000</td>
<td>EXT</td>
<td>-15.32</td>
</tr>
<tr>
<td>1.26</td>
<td>521</td>
<td>EXT</td>
<td>-18.38</td>
</tr>
<tr>
<td>2.93</td>
<td>1313</td>
<td>EXT</td>
<td>-18.38</td>
</tr>
<tr>
<td>6.75</td>
<td>3000</td>
<td>EXT</td>
<td>-18.38</td>
</tr>
<tr>
<td>6.62</td>
<td>3000</td>
<td>EXT</td>
<td>-18.38</td>
</tr>
<tr>
<td>6.56</td>
<td>3000</td>
<td>EXT</td>
<td>-18.38</td>
</tr>
<tr>
<td>6.56</td>
<td>3000</td>
<td>EXT</td>
<td>-18.38</td>
</tr>
</tbody>
</table>
Velocity = 2.5 m/s

- **SOLID**

Figure 80: Contour plot of x-velocity along a solid cylinder at 2.5 m/s

Figure 81: Contour plot of pressure coefficient along a solid cylinder at 2.5 m/s

Figure 82: Contour plot of TKE along a solid cylinder at 2.5 m/s
15 m\(^{-1}\)

Figure 83: Contour plot of x-velocity of a porous cylinder (15 m\(^{-1}\)) at 2.5 m/s

Figure 84: Contour plot of pressure coefficient of a porous cylinder (15 m\(^{-1}\)) at 2.5 m/s

Figure 85: Contour plot of TKE of a porous cylinder (15 m\(^{-1}\)) at 2.5 m/s
Figure 86: Contour plot of x-velocity of a porous cylinder ($75 \, m^{-1}$) at 2.5 m/s

Figure 87: Contour plot of pressure coefficient of a porous cylinder ($75 \, m^{-1}$) at 2.5 m/s

Figure 88: Contour plot of TKE of a porous cylinder ($75 \, m^{-1}$) at 2.5 m/s
Figure 89: Contour plot of x-velocity of a porous cylinder (150 m$^3$) at 2.5 m/s

Figure 90: Contour plot of pressure coefficient of a porous cylinder (150 m$^3$) at 2.5 m/s

Figure 91: Contour plot of TKE of a porous cylinder (150 m$^3$) at 2.5 m/s
Figure 92: Contour plot of x-velocity of a porous cylinder (300 m$^3$) at 2.5 m/s

Figure 93: Contour plot of pressure coefficient of a porous cylinder (300 m$^3$) at 2.5 m/s

Figure 94: Contour plot of TKE of a porous cylinder (300 m$^3$) at 2.5 m/s
Figure 95: Contour plot of x-velocity of a porous cylinder (750 m$^{-1}$) at 2.5 m/s

Figure 96: Contour plot of pressure coefficient of a porous cylinder (750 m$^{-1}$) at 2.5 m/s

Figure 97: Contour plot of TKE of a porous cylinder (750 m$^{-1}$) at 2.5 m/s
Figure 98: Contour plot of x-velocity of a porous cylinder (1500 m$^{-1}$) at 2.5 m/s

Figure 99: Contour plot of pressure coefficient of a porous cylinder (1500 m$^{-1}$) at 2.5 m/s

Figure 100: Contour plot of TKE of a porous cylinder (1500 m$^{-1}$) at 2.5 m/s
Velocity = 5 m/s

**SOLID**

**Figure 101:** Contour plot of x-velocity along a solid cylinder at 5 m/s

**Figure 102:** Contour plot of pressure coefficient along a solid cylinder at 5 m/s

**Figure 103:** Contour plot of TKE along a solid cylinder at 5 m/s
15 m$^{-1}$

**Figure 104:** Contour plot of x-velocity of a porous cylinder (15 m$^{-1}$) at 5 m/s

**Figure 105:** Contour plot of pressure coefficient of a porous cylinder (15 m$^{-1}$) at 5 m/s

**Figure 106:** Contour plot of TKE of a porous cylinder (15 m$^{-1}$) at 5 m/s
Figure 107: Contour plot of x-velocity of a porous cylinder (75 m$^{-1}$) at 5 m/s

Figure 108: Contour plot of pressure coefficient of a porous cylinder (75 m$^{-1}$) at 5 m/s

Figure 109: Contour plot of TKE of a porous cylinder (75 m$^{-1}$) at 5 m/s
Figure 110: Contour plot of x-velocity of a porous cylinder (150 m$^{-1}$) at 5 m/s

Figure 111: Contour plot of pressure coefficient of a porous cylinder (150 m$^{-1}$) at 5 m/s

Figure 112: Contour plot of TKE of a porous cylinder (150 m$^{-1}$) at 5 m/s
Figure 113: Contour plot of x-velocity of a porous cylinder (300 m$^{-1}$) at 5 m/s

Figure 114: Contour plot of pressure coefficient of a porous cylinder (300 m$^{-1}$) at 5 m/s

Figure 115: Contour plot of TKE of a porous cylinder (300 m$^{-1}$) at 5 m/s
Figure 116: Contour plot of x-velocity of a porous cylinder (750 m$^1$) at 5 m/s

Figure 117: Contour plot of pressure coefficient of a porous cylinder (750 m$^1$) at 5 m/s

Figure 118: Contour plot of TKE of a porous cylinder (750 m$^1$) at 5 m/s
1500 m$^{-1}$

Figure 119: Contour plot of x-velocity of a porous cylinder (1500 m$^{-1}$) at 5m/s

Figure 120: Contour plot of pressure coefficient of a porous cylinder (1500 m$^{-1}$) at 5m/s

Figure 121: Contour plot of TKE of a porous cylinder (1500 m$^{-1}$) at 5m/s
Velocity = 7.5 m/s

**SOLID**

Figure 122: Contour plot of x-velocity along a solid cylinder at 7.5 m/s

Figure 123: Contour plot of pressure coefficient along a solid cylinder at 7.5 m/s

Figure 124: Contour plot of TKE along a solid cylinder at 7.5 m/s
Figure 125: Contour plot of x-velocity of a porous cylinder ($15 \text{ m}^{-1}$) at 7.5m/s

Figure 126: Contour plot of pressure coefficient of a porous cylinder ($15 \text{ m}^{-1}$) at 7.5m/s

Figure 127: Contour plot of TKE of a porous cylinder ($15 \text{ m}^{-1}$) at 7.5m/s
Figure 128: Contour plot of x-velocity of a porous cylinder (75 m$^{-1}$) at 7.5m/s

Figure 129: Contour plot of pressure coefficient of a porous cylinder (75 m$^{-1}$) at 7.5m/s

Figure 130: Contour plot of TKE of a porous cylinder (75 m$^{-1}$) at 7.5m/s
150 m$^{-1}$

Figure 131: Contour plot of x-velocity of a porous cylinder (150 m$^{-1}$) at 7.5m/s

Figure 132: Contour plot of pressure coefficient of a porous cylinder (150 m$^{-1}$) at 7.5m/s

Figure 133: Contour plot of TKE of a porous cylinder (150 m$^{-1}$) at 7.5m/s
Figure 134: Contour plot of x-velocity of a porous cylinder (300 m$^3$) at 7.5m/s

Figure 135: Contour plot of pressure coefficient of a porous cylinder (300 m$^3$) at 7.5m/s

Figure 136: Contour plot of TKE of a porous cylinder (300 m$^3$) at 7.5m/s
Figure 137: Contour plot of x-velocity of a porous cylinder (750 m$^{-1}$) at 7.5 m/s

Figure 138: Contour plot of pressure coefficient of a porous cylinder (750 m$^{-1}$) at 7.5 m/s

Figure 139: Contour plot of TKE of a porous cylinder (750 m$^{-1}$) at 7.5 m/s
Figure 140: Contour plot of x-velocity of a porous cylinder (1500 m\(^{-1}\)) at 7.5 m/s

Figure 141: Contour plot of pressure coefficient of a porous cylinder (1500 m\(^{-1}\)) at 7.5 m/s

Figure 142: Contour plot of TKE of a porous cylinder (1500 m\(^{-1}\)) at 7.5 m/s
Velocity = 10 m/s

**SOLID**

*Figure 143: Contour plot of x-velocity along a solid cylinder at 10 m/s*

*Figure 144: Contour plot of pressure coefficient along a solid cylinder at 10 m/s*

*Figure 145: Contour plot of TKE along a solid cylinder at 10 m/s*
$15 \text{ m}^{-1}$

**Figure 146:** Contour plot of $x$-velocity of a porous cylinder ($15 \text{ m}^{-1}$) at 10 m/s

**Figure 147:** Contour plot of pressure coefficient of a porous cylinder ($15 \text{ m}^{-1}$) at 10 m/s

**Figure 148:** Contour plot of TKE of a porous cylinder ($15 \text{ m}^{-1}$) at 10 m/s
75 m$^{-1}$

**Figure 149:** Contour plot of x-velocity of a porous cylinder (75 m$^{-1}$) at 10 m/s

**Figure 150:** Contour plot of pressure coefficient of a porous cylinder (75 m$^{-1}$) at 10 m/s

**Figure 151:** Contour plot of TKE of a porous cylinder (75 m$^{-1}$) at 10 m/s
$150\ m^3$

*Figure 152: Contour plot of $x$-velocity of a porous cylinder ($150\ m^3$) at $10\ m/s$*

*Figure 153: Contour plot of pressure coefficient of a porous cylinder ($150\ m^3$) at $10\ m/s$*

*Figure 154: Contour plot of TKE of a porous cylinder ($150\ m^3$) at $10\ m/s$*
$300 \, m^{-1}$

**Figure 155**: Contour plot of x-velocity of a porous cylinder ($300 \, m^{-1}$) at 10 m/s

**Figure 156**: Contour plot of pressure coefficient of a porous cylinder ($300 \, m^{-1}$) at 10 m/s

**Figure 157**: Contour plot of TKE of a porous cylinder ($300 \, m^{-1}$) at 10 m/s
$750 \text{ m}^{-1}$

**Figure 158**: Contour plot of x-velocity of a porous cylinder ($750 \text{ m}^{-1}$) at 10 m/s

**Figure 159**: Contour plot of pressure coefficient of a porous cylinder ($750 \text{ m}^{-1}$) at 10 m/s

**Figure 160**: Contour plot of TKE of a porous cylinder ($750 \text{ m}^{-1}$) at 10 m/s
Figure 161: Contour plot of x-velocity of a porous cylinder (1500 m$^{-1}$) at 10 m/s

Figure 162: Contour plot of pressure coefficient of a porous cylinder (1500 m$^{-1}$) at 10 m/s

Figure 163: Contour plot of TKE of a porous cylinder (1500 m$^{-1}$) at 10 m/s
Velocity = 12.5 m/s

**SOLID**

**Figure 164**: Contour plot of x-velocity along a solid cylinder at 12.5 m/s

**Figure 165**: Contour plot of pressure coefficient along a solid cylinder at 12.5 m/s

**Figure 166**: Contour plot of TKE along a solid cylinder at 12.5 m/s
Figure 167: Contour plot of x-velocity of a porous cylinder (15 m$^1$) at 12.5 m/s

Figure 168: Contour plot of pressure coefficient of a porous cylinder (15 m$^1$) at 12.5 m/s

Figure 169: Contour plot of TKE of a porous cylinder (15 m$^1$) at 12.5 m/s
75 m$^{-1}$

Figure 170: Contour plot of x-velocity of a porous cylinder (75 m$^{-1}$) at 12.5 m/s

Figure 171: Contour plot of pressure coefficient of a porous cylinder (75 m$^{-1}$) at 12.5 m/s

Figure 172: Contour plot of TKE of a porous cylinder (75 m$^{-1}$) at 12.5 m/s
Figure 173: Contour plot of x-velocity of a porous cylinder (150 m$^3$) at 12.5 m/s

Figure 174: Contour plot of pressure coefficient of a porous cylinder (150 m$^3$) at 12.5 m/s

Figure 175: Contour plot of TKE of a porous cylinder (150 m$^3$) at 12.5 m/s
**Figure 176**: Contour plot of x-velocity of a porous cylinder (300 m\(^3\)) at 12.5 m/s

**Figure 177**: Contour plot of pressure coefficient of a porous cylinder (300 m\(^3\)) at 12.5 m/s

**Figure 178**: Contour plot of TKE of a porous cylinder (300 m\(^3\)) at 12.5 m/s
Figure 179: Contour plot of $x$-velocity of a porous cylinder ($750 \text{ m}^3$) at 12.5 m/s

Figure 180: Contour plot of pressure coefficient of a porous cylinder ($750 \text{ m}^3$) at 12.5 m/s

Figure 181: Contour plot of TKE of a porous cylinder ($750 \text{ m}^3$) at 12.5 m/s
Figure 182: Contour plot of x-velocity of a porous cylinder (1500 m$^{-1}$) at 12.5 m/s

Figure 183: Contour plot of pressure coefficient of a porous cylinder (1500 m$^{-1}$) at 12.5 m/s

Figure 184: Contour plot of TKE of a porous cylinder (1500 m$^{-1}$) at 12.5 m/s
Velocity = 15 m/s

**SOLID**

Figure 185: Contour plot of x-velocity along a solid cylinder at 15 m/s

Figure 186: Contour plot of pressure coefficient along a solid cylinder at 15 m/s

Figure 187: Contour plot of TKE along a solid cylinder at 15 m/s
$15 \text{ m}^{-1}$

Figure 188: Contour plot of x-velocity of a porous cylinder ($15 \text{ m}^{-1}$) at 15 m/s

Figure 189: Contour plot of pressure coefficient of a porous cylinder ($15 \text{ m}^{-1}$) at 15 m/s

Figure 190: Contour plot of TKE of a porous cylinder ($15 \text{ m}^{-1}$) at 15 m/s
Figure 191: Contour plot of x-velocity of a porous cylinder (75 m⁻¹) at 15 m/s

Figure 192: Contour plot of pressure coefficient of a porous cylinder (75 m⁻¹) at 15 m/s

Figure 193: Contour plot of TKE of a porous cylinder (75 m⁻¹) at 15 m/s
150 m$^{-1}$

Figure 194: Contour plot of x-velocity of a porous cylinder (150 m$^{-1}$) at 15 m/s

Figure 195: Contour plot of pressure coefficient of a porous cylinder (150 m$^{-1}$) at 15 m/s

Figure 196: Contour plot of TKE of a porous cylinder (150 m$^{-1}$) at 15 m/s
Figure 197: Contour plot of x-velocity of a porous cylinder (300 m$^{-1}$) at 15 m/s

Figure 198: Contour plot of pressure coefficient of a porous cylinder (300 m$^{-1}$) at 15 m/s

Figure 199: Contour plot of TKE of a porous cylinder (300 m$^{-1}$) at 15 m/s
Figure 200: Contour plot of x-velocity of a porous cylinder (750 m\(^{-1}\)) at 15 m/s

Figure 201: Contour plot of pressure coefficient of a porous cylinder (750 m\(^{-1}\)) at 15 m/s

Figure 202: Contour plot of TKE of a porous cylinder (750 m\(^{-1}\)) at 15 m/s
Figure 203: Contour plot of x-velocity of a porous cylinder (1500 m$^{-1}$) at 15 m/s

Figure 204: Contour plot of pressure coefficient of a porous cylinder (1500 m$^{-1}$) at 15 m/s

Figure 205: Contour plot of TKE of a porous cylinder (1500 m$^{-1}$) at 15 m/s
Appendix C – MATLAB code (*potentialflow.m*)

This section of the appendix provides the MATLAB code used to estimate the source and sink strength ratios and its comparison with the porous velocity profile.

The Excel workbook “ANSYS2PF_data” is obtained by exporting the point cloud data from the porous velocity field.

```matlab
clc
clear
clf
% This code outputs the streamlines, velocity vector field and curve fit
% for a source and sink pair in uniform flow.

% This method assumes that there is a net-sink flow around a porous body.
% This is analogous to the LRB method used in Araya et al. (2014)

% Equations for source/sink and uniform flow stream functions obtained from
%Aerodynamics for Engineering Students 2013 Ed by Houghton et al.

% Created by David Bassey 2019

% The user defines dimensions for objects and output is an array and figure of the potential flow solution.
res=4; % resolution based on ANSYS data in mm
Xmax=372; % Domain x limit
Ymax=148; % Domain y limit

% Grid
x = 24:res:Xmax; % x-grid
y = 52:res:Ymax; % y-grid

% Flow
% contributions

% Uniform flow profile (magnitude, angle)
U=[4,0];
```
%Source/Sink descriptions (x location, y location, strength)
%Each new source is a new row of the array
So=[120,100,1.55*U(1); 140,100,-2.54*U(1)];

psi = zeros(length(y),length(x)); %Initialize stream function grid
vx = zeros(length(y),length(x)); %Initialize x velocity
vy = zeros(length(y),length(x)); %Initialize y velocity
Borders=ones(length(y),length(x)); %Sets the working area
c=1; %Flow contribution counter

%Uniform flow
for m = 1:length(x)
    for n = 1:length(y)
        psi(n,m,c) = U(1)*(y(n)*cosd(U(2)) - x(m)*sind(U(2)));
        vx(n,m,c) = U(1)*cosd(U(2));
        vy(n,m,c) = U(1)*sind(U(2));
    end
end

%Source/Sink
for k = 1:size(So,1)
    if So(k,3) ~= 0
        c=c+1; %increment contribution
        for m = 1:length(x)
            for n = 1:length(y)
                if sqrt((x(m)-So(k,1))^2+(y(n)-So(k,2))^2)<So(k,3)
                    X=x(m)-So(k,1);
                    Y=y(n)-So(k,2);
                    psi(n,m,c) = (So(k,3)/(2*3.142))*atan2(Y,X);
                    vx(n,m,c)=(So(k,3)/2*pi)*((X)/(X^2+Y^2));
                    vy(n,m,c)=(So(k,3)/2*pi)*((Y)/(X^2+Y^2));
                end
            end
        end
    end
end
end
else
end
end

% Plot source/sink effect region based on strength
% Cylinder(So,res);

% Creation of boundary for solution matrices
solution=zeros(size(Borders));
xvelocity=zeros(size(Borders));
yvelocity=zeros(size(Borders));

% Results
% Collation of all stream function results
for b=1:size(psi,3)
solution=solution+psi(:,:,b);
xvelocity=xvelocity+vx(:,:,b);
yvelocity=yvelocity+vy(:,:,b);
end

xvelocity=xvelocity.*Borders;
yvelocity=yvelocity.*Borders;
v=sqrt(xvelocity.^2+yvelocity.^2); % compute the magnitude of the x and y velocity

% Extraction of simulation results at porosity level
ANSYS=xlsread(['ANSYS2PF_data.xlsx','Vx_Por15','B14:CK14']);
Simul = ANSYS/0.56; % U(1); % NON DIMENSIONALISED ANSYS RESULTS
x_simul = 100 *
xlsread(['ANSYS2PF_data.xlsx','Vx_Por15','B1:CK1']);

% Extraction of literature results at porosity level
ANSYS2=xlsread(['ANSYS_data.xlsx','Vx_Liu','H2:DU2']);
x_simul2 = 50+ 100 *
xlsread(['ANSYS_data.xlsx','Vx_Liu','H1:DU1']);

centerv = xvelocity(13,:); % centerline velocity from potential flow
Centerv_ND=centerv/U(1); % non dimensionalise centerline velocity based on incoming velocity
mean_pflow = mean(Centerv_ND,'omitnan');
mean_porflow = mean(Simul,'omitnan');

% Calculate R squared for each value
SS_res = (Centerv_ND-Simul).^2;

%Average R-squared value omitting NaN values
Av_SSres = mean(SS_res,'omitnan');

SS_tot = (Centerv_ND - mean_porflow).^2;
Av_SStot = mean(SS_tot,'omitnan');
R_sq = 1 - (Av_SSres/Av_SStot);

%plot results
figure(1)
contour(x,y,solution,40) % streamline plot
% quiver(x,y,xvelocity,yvelocity) % velocity vector plot
xlabel('X (cm)')
ylabel('Y (cm)')
grid on
axis equal

figure (2)
% comparison of source/sink strength with simulation results
plot(x,Centerv_ND,'*k-',x_simul,Simul,'om-')
xlabel('X (cm)')
ylabel('U/U_inf')
grid on
ylim([0.3 1.1]) % y axis limit set for better visualization

%Highest R-square is used
disp('Mean of calculated velocity')
disp(mean_pflow)

disp('Mean of simulated porous velocity')
disp(mean_porflow)

disp('Residual sum of squares')
disp(Av_SSres)

disp('Total sum of squares')
disp(Av_SStot)

disp('R-squared')
disp(R_sq)
Appendix D – MATLAB code (*grabit.m*)

This section of the appendix provides the MATLAB code used to extract the published results used to compare to simulation results in this thesis. This code was provided via The MathWorks, Inc.

```matlab
function grabit(fname)

%GRABIT Extracts data points from an image file.
% GRABIT starts a GUI program for extracting data from an image file.
% It is capable of reading in BMP, JPG, TIF, GIF, and PNG files (anything
% that is readable by IMREAD). Multiple data sets can be
% extracted from a
% single image file, and the data is saved as an n-by-2 matrix
% variable in
% the workspace. It can also be renamed and saved as a MAT file.
%
% Following steps should be taken:
% 1. Load the image file.
% 2. Calibrate axes dimensions. You will be prompted to
% select 4 points
% on the image. Zoom and pan enabled.
% 3. Grab points by clicking on points. Right-click to delete
% a point.
%   Image can be zoomed and panned.
% 4. Multiple data sets will remain in memory so long as the
% GUI is open.
% Variables can be renamed, saved to file, or edited in
% Array Editor.
%
% Panning is achieved by clicking and dragging on the image.
% Double-click
% to center view. Right click and drag to zoom in and out. In
% addition,
% there are keyboard shortcuts for zooming:
% <a> - zoom in
```
% <b> - zoom out
%  <space> - reset view
%
% This code will also work for extracting data points from a
tilted or a
% skewed image (even upside-down or mirrored). The calibration
stage
% ensures that the imperfect orientation or quality of the image is
% accounted for.
%
% The types of files that will most likely work are BMP, JPG,
TIF, GIF (up
% to 8-bit), and PNG files. Basically, any format supported by
the IMREAD
% is accepted.
%
% GRABIT(FILENAME) will start the GUI program and open the image
file
% FILENAME.
%
% Type GRABIT('-sample') to load a sample image.
%
% VERSIONS:
%  v1.0 - first version
%  v1.1 - use imshow instead of image (takes care of colormap)
%  v1.5 - convert to a GUI version
%  v1.6 - added functionality to open a file from the command
window and
%        embedded a sample image file to the function
%  v1.6.1 - changed cross cursor to crosshair
%  v1.6.2 - brought back 'image' in case the user doesn't have
Image Toolbox
%  v1.6.5 - fixed zoom problem in R14
%  v2.0 - major code change. added zoom feature during
calibration. added
%        panning feature. (March 3, 2006)
%  v2.1 - store sample image as HEX to reduce file size. (March
6, 2006)
%  v2.1.1 - animate view change and zooming for a better visual
perception
%        (March 11, 2006)
%  v2.1.5 - added features: double-click to center view. right-
click and
%        drag to zoom. other minor code changes. (March 16, 2006)
% v2.2 - fixed loadImageFcn bug. (May 4, 2006)
% v2.3 - fixed bug to work with grayscale image (Jan, 2007)
%

% Created in Matlab R13. Tested up to R2006b
% % Copyright 2003
% % Jiro Doke
% % To Do: Capability to deal with logarithmic axes
%

%------------------------------------------------------------------------
% Initialize
%------------------------------------------------------------------------
sh = get(0, 'ShowHiddenHandles');
set(0, 'ShowHiddenHandles', 'on');

% close existing windows
im  =  findobj('type', 'figure', 'tag', 'GrabitGUI');
if ishandle(im)
    close(im);
end
set(0, 'ShowHiddenHandles', sh);

% background colors
bgcolor1 = [.8, .8, .8];
bgcolor2 = [ 1,  1,  1];
bgcolor3 = [.7, .7, .7];
bgcolor4 = [ 1, .5, .5];

%------------------------------------------------------------------------
% Custom cursor pointers
%------------------------------------------------------------------------
zoomPointer = [
    2 2 2 2 2 2 2 NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN
    2 1 1 1 1 1 2 NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN
    2 1 2 2 2 2 NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN
];
```
zoomInOutPointer = [
    NaN NaN NaN 2 2 NaN NaN NaN NaN NaN NaN
    NaN 1 1 1 NaN NaN
    NaN NaN 2 1 1 2 NaN NaN NaN NaN NaN
    1 2 2 2 1 NaN
    NaN 2 1 1 1 1 2 NaN NaN 1
    2 2 1 2 2 1
    2 1 1 1 1 1 1 2 NaN 1
    2 1 1 2 1 2 1 1 2 NaN 1
    2 1 1 1 2 1
    NaN 2 2 1 1 2 2 NaN NaN 1
    NaN NaN 2 1 1 2 NaN NaN NaN 1
    1 2 2 2 1 NaN
    NaN NaN 2 1 1 2 NaN NaN 1
    1 1 1 1 NaN NaN
    NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN
];
```
NaN  NaN  2  1  1  2  NaN  NaN  NaN  NaN
1  2  2  2  1  NaN
NaN  2  2  1  1  2  2  NaN  NaN  1
2  2  2  2  2  1
  2  1  2  1  1  2  1  2  NaN  1
  2  1  1  1  1  1  1  2  NaN  1
2  2  2  2  2  1
NaN  2  1  1  1  1  2  NaN  NaN  1
1  2  2  2  1  NaN
NaN  NaN  2  1  1  2  NaN  NaN  1  1
1  1  1  1  NaN  NaN
NaN  NaN  NaN  2  2  NaN  NaN  NaN  NaN  1  1
NaN  NaN  NaN  NaN  NaN  NaN
];

% closed hand pointer (from Jérôme Briot)
closedHandPointer = [
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
    NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
];
NaN NaN NaN NaN NaN 2 1 2 2 2 2 2
2 2 1 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN 2 1 2 2 2 2 2
2 2 1 2 NaN NaN NaN NaN NaN NaN
];

% X-axis Origin pointer
xoPointer = [
    2 2 2 2 2 2 2 NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    2 1 1 1 1 1 2 NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    2 1 1 2 2 2 2 NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    2 1 2 NaN 2 1 2 NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    2 2 2 NaN NaN 2 2 NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
];

% X-axis Max pointer
xmPointer = [
    2 2 2 2 2 2 2 NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    2 2 2 2 NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    2 1 1 1 NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    2 1 NaN NaN NaN 1 NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
];
NaN NaN NaN NaN NaN NaN NaN
2 1 1 1 1 1 2 NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 2 2 2 2 2 NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 2 1 2 NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 2 NaN 2 1 NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 2 NaN NaN 2 2 NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 1 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
2 2 NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN
```matlab
% Y-axis Max pointer
ymPointer = [
    2 2 2 NaN NaN 2 2 NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN 1 1 NaN NaN NaN 1 1
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
    NaN NaN NaN NaN NaN NaN
];
```
NaN   NaN   NaN   NaN   NaN   1   1   1   1   NaN
NaN   1   NaN   NaN   NaN   1
NaN   NaN   NaN   NaN   NaN   1   1   1   NaN   NaN
NaN   1   1   NaN   1   1
NaN   NaN   NaN   1   1   1   1   NaN   NaN
NaN   1   NaN   1   NaN   1
NaN   NaN   NaN   1   1   1   NaN   NaN   NaN   NaN
NaN   NaN   NaN   NaN   NaN   1   1   NaN   NaN   1

% zoom button icon
zoomIcon = [
    1   1   1   1   1   0   0   0   0   1
    1   1   1   1   1
    1   1   1   0   0   0   0   0   0   0
    0   1   1   1   1
    1   1   0   0   0   1   1   1   1   0
    0   1   1   1   1
    1   0   0   1   1   1   0   0   1   1
    1   0   0   1   1   1
    1   0   0   1   1   1   1   0   0   1   1
    1   0   0   1   1   1
    0   1   0   0   1   1
    0   0   1   0   0   0   0   0   0   0
    0   1   0   0   1   1
    0   0   1   0   0   0   0   0   0   0
    0   1   0   0   1   1
    0   0   1   1   1   1   1   0   0   1   1
    1   1   0   0   1   1
    1   0   0   1   1   1   1   0   0   1   1
    1   0   0   1   1   1
    1   0   0   1   1   1   1   0   0   1   1
    1   0   0   1   1   1
    1   0   0   1   1   1
    0   1   0   0   1   1
    0   0   1   0   0   1   1
    0   1   1   0   0   0   0   0   0
    0   0   0   0   1   1
    1   1   1   0   0   0   0   0   0
    0   0   0   0   0   1
    1   1   1   1   1   0   0   0   0   1
    1   0   0   0   0
    1   1   1   1   1   1
    1   1   0   0   0
    1   1   1   1   1   1
    1   1   1   1   1
    1   1   1   1   0   0   0
];
% create zoom icon (RGB matrix)
fgID = zoomIcon==0;
bID = ~fgID;
zI1 = zeros(size(zoomIcon), 3);
zI2 = zeros(size(zoomIcon), 3);
for id = 1:3
tmp = zoomIcon;
tmp(fgID) = 0;
tmp(bID) = bgcolor3(id);
zI1(:,:,id) = tmp;
tmp(bID) = bgcolor4(id);
zI2(:,:,id) = tmp;
end

% get screen size in pixels
un = get(0, 'units');
set(0, 'units', 'pixels');
screenSize = get(0, 'ScreenSize');
sW = screenSize(3);
sH = screenSize(4);
set(0, 'units', un);

% figure width and height (in pixels)
fW = sW-200;
fH = sH-100;

im = figure(...
    'units', 'pixels', ...
    'position', [100, 50, fW, fH], ...
    'backingstore', 'off', ...
    'doublebuffer', 'on', ...
    'name', 'Grabit', ...
    'numbertitle', 'off', ...
    'menubar', 'none', ...
    'color', bgcolor1, ...
    'pointer', 'arrow', ...
    'visible', 'off', ...
    'interruptible', 'off', ...
    'busyaction', 'cancel', ...
    'resizefcn', @figResizeFcn, ...
    'windowbuttonupfcn', @winBtnUpFcn, ...
    'keypressfcn', @keyPressFcn, ...
    'deletefcn', ...
    'delete(timerfind(''name'', ''BtnUpTimer''));', ...
    'tag', 'GrabitGUI', ...
    'defaultUicontrolUnits', 'pixels', ...
panelW = 0.8*fW;
uicontrol(...
    'style', ...
    'position', ...
    'backgroundcolor', bgcolor2);
uicontrol(...
    'style', ...
    'pushbutton', ...
    'callback', {@loadImageFcn, []}, ...
    'position', [15, fH-40, 200, 25], ...
    'tag', 'LoadImageBtn');

ucicontrol(...
    'style', ...
    'edit', ...
    'backgroundcolor', bgcolor2, ...
    'string', '', ...
    'position', [220, fH-40, panelW-215, 25], ...
    'horizontalalignment', 'left', ...
    'horizontalalignment', 'inactive', ...
    'tag', 'ImageFileLoc');

ucicontrol(...
    'style', ...
    'frame', ...
    'position', [15, fH-75, 3*(panelW-20)/4-5, 30], ...
    'backgroundcolor', bgcolor2);
ucicontrol(...
    'style', ...
    'togglebutton', ...
    'buttondownfcn', @calibrateImageFcn, ...
    'position', [20, fH-72, panelW/4-25, 25], ...
    'tag', 'CalibrateImageBtn');
calibFrameW = panelW/2-10;
uicontrol(...
    'style'    , 'text', ...
    'position', [calibFrameX+5, fH-70, 
    'string'  , 'Xo:', ...
    'fontweight','bold', ...
    'horizontalalignment' ,'right', ...
    'tooltipstring' , sprintf('Calibration:
Axis Origin'), ...
    'backgroundcolor' , bgcolor2);

disp(['NaN', 'Xm:', 'NaN', 'Yo:', '')
'tooltipstring', sprintf('Calibration:
Y-Axis Origin'), ...
'backgroundcolor', bgcolor2);
uicontrol(...
'style', 'text', ...
'position', ...
[calibFrameX+5+5*calibFrameW/8, fH-70, calibFrameW/8, 15], ...
'string', 'NaN', ...
'horizontalalignment', 'left', ...
'tag', 'hYoValue', ...
'tooltipstring', sprintf('Calibration:
Y-Axis Origin'), ...
'backgroundcolor', bgcolor2);
uicontrol(...
'style', 'text', ...
'position', ...
[calibFrameX+5+3*calibFrameW/4, fH-70, calibFrameW/8, 15], ...
'string', 'Ym:', ...
'fontweight', 'bold', ...
'horizontalalignment', 'right', ...
'tooltipstring', sprintf('Calibration:
Y-Axis Max'), ...
'backgroundcolor', bgcolor2);
uicontrol(...
'style', 'text', ...
'position', ...
[calibFrameX+5+7*calibFrameW/8, fH-70, calibFrameW/8, 15], ...
'string', 'NaN', ...
'horizontalalignment', 'left', ...
'tag', 'hYmValue', ...
'tooltipstring', sprintf('Calibration:
Y-Axis Max'), ...
'backgroundcolor', bgcolor2);
uicontrol(...
'style', 'togglebutton', ...
'string', 'Grab Points', ...
'buttondownfcn', @grabPointsFcn, ...
'position', [3*(panelW)/4, fH-72, panelW/4+5, 25], ...
'enable', 'off', ...
'tag', 'GrabPointsBtn');
uicontrol(...
'style', 'togglebutton', ...
'cdata', zI1, ...
'buttondownfcn', @zoomBtnFcn, ...
'position', [15, fH-105, 25, 25], ...
'enable', 'inactive', ...
'tag', 'ZoomStateBtn');
uicontrol(...
 'style', 'pushbutton', ...
 'string', 'Reset View', ...
 'callback', @resetViewFcn, ...
 'position', [45, fH-105, 100, 25], ...
 'tag', 'ResetViewBtn');
uicontrol(...
 'style', 'text', ...
 'position', [150, fH-108, panelW-145, 31], ...
 'horizontalalignment', 'center', ...
 'foregroundcolor', [0 0 .5], ...
 'backgroundcolor', bgcolor2, ...
 'fontsize', 10, ...
 'string', {'Click and drag to pan. Double-click to center. Right-click and drag to zoom.',...
 'Keyboard Shortcuts: <a> - zoom in, <z> - zoom out, <space> - reset view!');}

rPanelX = 0.82 * fW;
 rPanelW = fW - rPanelX - 10;
uicontrol(...
 'style', 'listbox', ...
 'callback', @selectVariableFcn, ...
 'position', [rPanelX+10, 100, rPanelW-20, 0.6*fH], ...
 'backgroundcolor', bgcolor2, ...
 'tooltipstring', sprintf('Double-click to edit
variable in Array Editor'), ...
 'tag', 'VariableList');
uicontrol(...
 'style', 'text', ...
 'string', 'Data in Memory', ...
 'position', [rPanelX+10, 0.6*fH+100, rPanelW-20, 20], ...
 'backgroundcolor', bgcolor1, ...
 'fontweight', 'bold');
uicontrol(...
 'style', 'pushbutton', ...
 'string', 'Save to file as...', ...
 'position', [rPanelX+10, 70, rPanelW-20, 25], ...
 'callback', @variableManipulationFcn,
'tooltipstring', sprintf('Save variable as a MAT file or\nDouble-precision, tab-delimited TXT file'), ...
'tag'  
uicontrol(...
'callback'  
...'tooltipstring'
'workspace', ...
'tag'  
uicontrol(...
'callback'  
...'tooltipstring'
'workspace', ...
'tag'  
axes(...
'imAxRatio = (fH - 200) / (0.8 * fW);
title('', 'fontunits', 'pixels', 'fontsize', 24, 'color', 'red');

axes(...
rPanelW-30, 0.4*fH-190], ...
'box'  
'tag'  
uicontrol(...
'style', 'text', ...
'string', 'Preview Box', ...
'fontweight', 'bold', ...
'position', [rPanelX+20, fH-40,
rPanelW-30, 25], ...
'backgroundcolor'

uicontrol(...
'style', 'edit', ...
'backgroundcolor', bgcolor4, ...
'string', 'Enter Value', ...
'position', [0 0 100 25], ...
'horizontalalignment', 'left', ...
'visible', 'off', ...
'callback', @coordinateEditFcn, ...
'tag', 'CoordinateEdit');

set(findobj(im, 'type', 'uicontrol'), 'units', 'normalized');
set(findobj(im, 'type', 'axes'), 'units', 'normalized');

% create handles structure
handles                           = guihandles(im);
handles.zoomPointer               = zoomPointer;
handles.zoomPointerHotSpot        = [2 2];
handles.zoomInOutPointer         = zoomInOutPointer;
handles.zoomInOutPointerHotSpot   = [9 9];
handles.closedHandPointer         = closedHandPointer;
handles.closedHandPointerHotSpot  = [9 9];
handles.xoPointer                 = xoPointer;
handles.xoPointerHotSpot          = [2 2];
handles.xmPointer                 = xmPointer;
handles.xmPointerHotSpot          = [2 2];
handles.yoPointer                 = yoPointer;
handles.yoPointerHotSpot          = [2 2];
handles.ymPointer                 = ymPointer;
handles.ymPointerHotSpot          = [2 2];
handles.zoomIconUp                 = zI1;
handles.zoomIconDown               = zI2;
handles.curPointer                = 'arrow';
handles.curPointerData.CData      = zoomPointer;
handles.curPointerData.HotSpot    = [1 1];
handles.state                     = 'normal';
handles.bgcolor1                  = bgcolor1;
handles.bgcolor2                  = bgcolor2;
handles.bgcolor3                  = bgcolor3;
handles.bgcolor4                  = bgcolor4;
handles.imAxRatio                 = imAxRatio;
handles.I = []; handles.map = []; handles.savedVars = struct; handles.isPanning = false; handles.curTitle = ''; handles.CurrentPointAxes = []; handles.CurrentPointFig = []; handles.timer = timer('Name', 'BtnUpTimer',
... '
'StartDelay', 0.2, ...
'TimerFcn',
{@btnUpTimerFcn, im});

guidata(im, handles);

set(im, 'handlevisibility', 'callback');

if nargin == 1 && ischar(fname)
  switch lower(fname)
  case '-sample'
    fname = createSampleImageFcn;
    if ~isempty(fname)
      loadImageFcn(handles.LoadImageBtn, [], fname);
      try
        pause(0.5);
        delete(fname);
      catch
        errordlg(lasterr);
      end
    end
    otherwise
    filename = which(fname);
    if ~isempty(filename)
      loadImageFcn(handles.LoadImageBtn, [], filename);
    else
      errordlg(sprintf('%s\nnot found.', fname));
      return;
    end
  end
  set(im, 'visible', 'on');
function variableManipulationFcn(varargin)
% This callback is called when one of the three buttons under
% the variable
% listbox is pressed.

obj = varargin{1};

handles = guidata(obj);

vars = fieldnames(handles.savedVars);
if isempty(vars)
    return;
end
listboxVal = get(handles.VariableList, 'value');
varName = vars(listboxVal);
switch get(obj, 'Tag')
    case 'SaveAs' % save the variable as a .mat file
        [fname, pname] = uiputfile(...
            '*.mat', 'MAT files (*.mat)'; ...
            '*.txt', 'TXT files (*.txt)'; ...
            sprintf('Save ''%s'' as:', varName), ...
            varName);
        if ~isequal(fname, 0) && ~isequal(pname, 0)
            saveDatFcn(pname, fname, handles.savedVars.(varName))
        end
    case 'Rename' % rename the variable in the base workspace
        answer = inputdlg({sprintf('Rename ''%s'' as:', varName)}, ...
            'Rename...', 1, {varName});
        if ~isempty(answer) && strcmp(answer{1})
            if ~(evalin('base', sprintf('exist('''%s''', ''var'')'), answer{1})) & ...
                isempty(strmatch(answer{1}, vars, 'exact'))
                newVarNames = vars;
                newVarNames{listboxVal} = answer{1};
                for id = 1:length(vars)

```
tmp.(newVarNames{id}) = handles.savedVars.(vars{id});
end
handles.savedVars = tmp;
showAllVarsFcn(handles);
set(handles.VariableList, 'value', listboxVal);
assignin('base', answer{1},
handles.savedVars.(answer{1}));
evalin('base', sprintf('clear %s;', varName));
else
errordlg('That name is already in use.');
return;
end
end

case 'Delete' % delete the variable from the base workspace
btn = questdlg(sprintf('Delete ''%s'' from the workspace?',
varName), ... 
'Delete from workspace...', 'Yes', 'No', 'No');
switch btn
  case 'Yes'
    handles.savedVars = rmfield(handles.savedVars, varName);
    showAllVarsFcn(handles);
    evalin('base', sprintf('clear %s;', varName));
  case 'No'
end
end

guidata(obj, handles);

%---------------------------------------------------------------
% saveDatFcn
%---------------------------------------------------------------
function saveDatFcn(pname, fname, var)
% this function saves the variable to file

[p, fname, ext] = fileparts(fname);
switch lower(ext)
  case '.mat'
    eval(sprintf('%s = var;', fname));
save(fullfile(pname, [fname, ext]), fname, '-v6');

    case '.txt'
        eval(sprintf('%s = var;', fname));
        save(fullfile(pname, [fname, ext]), fname, '-ascii', '-double', '-tabs');
    otherwise
        errordlg('Unknown extension.');
    end

%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
% selectVariableFcn
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------

function selectVariableFcn(varargin)
% this callback is called when a variable is selected from the variable
% listbox. When the variable is clicked, it will plot the data in the
% Preview Box above. If the variable is double-clicked, it will be opened
% in the Array Editor. It also ensures that the variable in the base
% workspace is the same copy as the variable stored in the GRABIT
% workspace. This means that if you change the contents of a variable in
% the base workspace (via other functions), then it will allow you to
% update the variable in the GRABIT workspace.

obj = varargin{1};

handles = guidata(obj);

sType = get(handles.GrabitGUI, 'SelectionType');
switch sType
    case {'normal', 'open'} % single or double click
        vars = fieldnames(handles.savedVars);
        if isempty(vars)
            return; vars
        end
end

obj = varargin{1};

handles = guidata(obj);

sType = get(handles.GrabitGUI, 'SelectionType');
switch sType
    case {'normal', 'open'} % single or double click
        vars = fieldnames(handles.savedVars);
        if isempty(vars)
            return; vars
        end
end
listVal = get(obj, 'value');
varName = vars{listVal};

% check to see if the stored var is the same as the var in the base workspace
if evalin('base', sprintf('exist('%s', ''var'')', varName))

% the copy in the base workspace must be a n-by-2 DOUBLE array
if strcmp(class(evalin('base', varName)), 'double') && ...
    ndims(evalin('base', varName)) == 2 && ...
    size(evalin('base', varName), 2) == 2
if ~isequal(evalin('base', varName), handles.savedVars.(varName))
% may have been modified in the base workspace
    btn = questdlg(sprintf('''%s'' may have been modified in the base workspace (e.g. Array Editor).\nUpdate the variable?', varName), ...
    'Modified Variable', 'Update Base Workspace', 'Update Grabit', 'Neither', 'Update Grabit');

    switch btn
    case 'Update Base Workspace'
        assignin('base', varName, handles.savedVars.(varName));
    case 'Update Grabit'
        handles.savedVars.(varName) = evalin('base', varName);
        showAllVarsFcn(handles);
        set(obj, 'value', listVal);
    end
end
else
    btn = questdlg(sprintf('''%s'' in the base workspace is different from the one in Grabit.\nUpdate the base workspace variable?', varName), ...
    'Wrong Variable', 'Update Base Workspace', 'Leave Untouched', 'Update Base Workspace');

    switch btn
case 'Update Base Workspace'
    assignin('base', varName, handles.savedVars.(varName));
end
end

else % the variable does not exist in base workspace, so make a copy
    assignin('base', varName, handles.savedVars.(varName));
end

switch sType
    case 'normal' % single click
        axes(handles.PreviewAxis);
        set(handles.PreviewLine, ...
             'xdata', handles.savedVars.(varName)(:, 1), ...
             'ydata', handles.savedVars.(varName)(:, 2));
        axis auto;

    case 'open' % double click
        try
            openvar(varName);
        catch
            errordlg(lasterr);
        end
end
end

guidata(obj, handles);

%---------------------------------------------------------------
%---------------------------------------------------------------
% loadImageFcn
% this function loads an image file

[obj, filename] = splitvar(varargin([1, 3]));
handles = guidata(obj);

if isempty(filename)
    [fname, pathname] = uigetfile(...
        {'*.bmp;*.jpg;*.jpeg;*.tif;*.tiff;*.gif;*.png', ...
        'Image Files (*.bmp, *.jpg, *.jpeg, *.tif, *.tiff, 
        *.gif, *.png); 
        '*.bpm', 'BPM files (*.bpm)'; 
        '*.jpg;*.jpeg', 'JPG files (*.jpg, *.jpeg)';
        '*.tif;*.tiff', 'TIFF files (*.tif, *.tiff)';
        '*.gif', 'GIF files (*.gif)';
        '*.png', 'PNG files (*.png)';
        '*.*', 'All files (*.*)'}, 'Select an image file');
    if ischar(fname)
        filename = fullfile(pathname, fname);
    else
        return;
    end
end

set(handles.ImageFileLoc, 'string', filename);

try
    %warning off;
    [A, map] = imread(filename);
    %warning on;
    catch
        errordlg(lasterr);
        return;
end

if ndims(A) == 3  %some TIFF files have wrong size matrices.
    if size(A, 3)>3
        A = A(:, :, 1:3);
    elseif size(A, 3)<3
        errordlg('This is a weird image file...possibly a bad TIFF file...');
        return;
    end
end

% Need this so that image shows up when not called by a CALLBACK set(0, 'ShowHiddenHandles', 'on');

cla(handles.PreviewAxis);
handles.PreviewLine = line(NaN, NaN, ...
if isempty(map)
    imageInfo = imfinfo(filename);
    if strcmpi(imageInfo.ColorType, 'grayscale')
        colormap(gray(2^imageInfo.BitDepth));
    end
else
    colormap(map);
end
iH = image(A);
set(iH, 'HitTest', 'off', 'EraseMode', 'normal');
set(handles.ImageAxis, 'xtick', [], 'ytick', []);
axis equal;

set(handles.GrabitGUI, 'nextplot', NP);
set(handles.ImageAxis, ...
    'drawmode' , 'fast', ...
    'tag' , 'ImageAxis', ...
    'callback', @winBtnDownFcn);
set(get(handles.ImageAxis, 'title'), ...
    'string' , '', ...
    'fontunits' , 'pixels', ...
    'fontsize' , 24, ...
    'color' , 'red');
handles.ImageLine = line(NaN, NaN, ...
    'color' , 'red', ...
    'linestyle' , 'none', ...
    'marker' , '.', ...
    'tag' , 'ImageLine', ...
    'hitTest' , 'off');
handles.CalibPtsH(:,1) = line(repmat(NaN, 2, 4), repmat(NaN, 2, 4));
handles.CalibPtsH(:,2) = line(repmat(NaN, 2, 4), repmat(NaN, 2, 4));
set(handles.CalibPtsH(:,1), ...
    {'marker', 'color'}, {'o', 'r', 'o', 'b', 'o', 'b'}, ...
    'markersize', 10, ...
'hittest'    , 'off');
set(handles.CalibPtsH(:,2), ...  
    {'marker','color'}, {'+', 'r'; 'x', 'r'; '+' , 'b'; 'x', 'b'}, ...  
    'markersize'    , 20, ...  
    'hittest'       , 'off');

% initialize image data
handles.I    = A;  
handles.map  = map;  
handles.CalibVals  = [];  
handles.CalibPts  = [NaN, NaN, NaN, NaN];  
handles.CalibPtsIm  = repmat(NaN, 4, 2);  
handles.ImLimits  = [get(handles.ImageAxis, 'xlim'); ...  
                    get(handles.ImageAxis, 'ylim')];

set(handles.CalibrateImageBtn, ...)  
    'string', 'Calibrate', ...  
    'enable', 'inactive', ...  
    'value' , 0);  
set(handles.GrabPointsBtn, ...)  
    'enable', 'off', ...  
    'value' , 0);  
set(handles.ZoomStateBtn, ...)  
    'enable', 'inactive', ...  
    'value' , 0);  
zoom off;

set(0, 'ShowHiddenHandles', 'off');

guidata(obj, handles);

%--------------------------------------------------------------------------
%---------------------------------------------
%---------------------------------------------
% calibrateImageFcn
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

function calibrateImageFcn varargin
% this function performs calibration of the image by prompting the user to  
% select 4 points on the image as reference points.

    obj = varargin{1};
handles = guidata(obj);

if isempty(handles.I)
    set(obj, 'enable', 'off');
else
    switch get(obj, 'value')
    case 0 % start calibration
        set(obj, 'value', 1, 'backgroundcolor', handles.bgcolor4);
        handles.curPointer = 'custom';
        handles.curPointerData.CData = handles.xoPointer;
        handles.curPointerData.HotSpot = handles.xoPointerHotSpot;
        set(handles.GrabitGUI, ...
            'PointerShapeCData', handles.curPointerData.CData,
            ...
            'PointerShapeHotSpot', handles.curPointerData.HotSpot, ...
            'WindowButtonMotionFcn', {@pointerFcn, handles, handles.curPointer});
        set([handles.LoadImageBtn, ...
            handles.SaveAs, ...
            handles.Rename, ...
            handles.Delete], ...
            'enable', 'off');
        set(handles.CalibPtsH, 'xdata', NaN, 'ydata', NaN);
        handles.CalibVals = [];
        handles.CalibPts = [NaN, NaN, NaN, NaN];
        handles.CalibPtsIm = repmat(NaN, 4, 2);
        set([handles.hXoValue, handles.hXmValue, ...
            handles.hYoValue, handles.hYmValue], ...
            'string', ' NaN');
        handles.curTitle = 'Click on the ORIGIN of x-axis';
        set(get(handles.ImageAxis, 'title'), ...
            'string', handles.curTitle);
        % change state to CALIBRATION
        handles.state = 'calibration';
    case 1 % stop (prematurely) calibration
        set(obj, ...
            'value', 0, ...
            'enable', 'off');
        handles.CalibPts = [NaN, NaN, NaN, NaN];
        handles.CalibPtsIm = repmat(NaN, 4, 2);
'backgroundcolor', handlesbgcolor3, ...
'string' ..., 'Calibrate');
handles.curTitle = '';
set(get(handles.ImageAxis, 'title'), 'string', '');

handles.curPointer = 'arrow';
set(handles.GrabitGUI, ...
'WindowButtonMotionFcn', {@pointerFcn, handles, handles.curPointer});

set([handles.LoadImageBtn, ...
    handles.SaveAs, ...
    handles.Rename, ...
    handles.Delete], ...
'enable', 'on');

% calibration was prematurely stopped, so reset all values
set(handles.CalibPtsH, 'xdata', NaN, 'ydata', NaN);
handles.CalibVals = [];
handles.CalibPts = [NaN, NaN, NaN, NaN];
handles.CalibPtsIm = repmat(NaN, 4, 2);
set([handles.hXoValue, handles.hXmValue, ...
    handles.hYoValue, handles.hYmValue], ...
'string', ' NaN');

set(handles.GrabPointsBtn, 'enable', 'off');
set(handles.CoordinateEdit, 'visible', 'off');

% change state to NORMAL
handles.state = 'normal';
end
end

guidata(obj, handles);

%---------------------------------------------------------------------------------------------
%---------------------------------------------------------------------------------------------
% grabPointsFcn
%---------------------------------------------------------------------------------------------
%---------------------------------------------------------------------------------------------
function grabPointsFcn(varargin)
% this function is used to extract data points by prompting the user to
% select points on the image.

obj = varargin{1};

handles = guidata(obj);

switch get(obj, 'value')
  case 0  % initiate point grabbing
    calib = handles.CalibVals;
    
    axes(handles.ImageAxis);
    set(handles.ImageAxis, ...% 'xlim', handles.ImLimits(1,:), ...% 'ylim', handles.ImLimits(2,:));
    set(handles.ImageLine, 'xdata', NaN, 'ydata', NaN);
    handles.curTitle = {'Grab points by clicking on data points.', ...% '<BACKSPACE> or <DEL> to delete previous point. <ENTER> to finish.'};
    set(get(handles.ImageAxis, 'title'), ...
        'string', handles.curTitle);

    set(handles.PreviewLine, 'xdata', NaN, 'ydata', NaN);
    set(handles.PreviewAxis, ...
        'xlim', [min([calib.Xo calib.Xm]) max([calib.Xo calib.Xm])], ...
        'ylim', [min([calib.Yo calib.Ym]) max([calib.Yo calib.Ym])]);

    handles.ImDat   = [];
    handles.TrueDat = [];

    handles.curPointer          = 'crosshair';
    set(handles.GrabitGUI, ...
        'WindowButtonMotionFcn', {@pointerFcn, handles, handles.curPointer});

    set(obj, ...
        'value' , 1, ...
        'string' , 'Grabbing Points (0)', ...
        'backgroundcolor' , handles.bgcolor4);
    set([handles.CalibrateImageBtn, ...
        handles.LoadImageBtn, ...
        handles.SaveAs, ... handles.Rename, ...
handles.Delete], ...
'enable', 'off');

% change state to GRAB
handles.state = 'grab';

case 1 % finish point grabbing
set(obj, ...
'value' , 0, ...
'string' , 'Grab Points', ...
'backgroundcolor' , handles.bgcolor1);

handles.curTitle = '';
set(get(handles.ImageAxis, 'title'), 'string', '');
handles.curPointer = 'arrow';
set(handles.GrabitGUI, 'WindowButtonMotionFcn',
{@pointerFcn, handles, handles.curPointer});
set(handles.CalibrateImageBtn, 'enable', 'inactive');
set([handles.LoadImageBtn, ...
handles.SaveAs, ...
handles.Rename, ...
handles.Delete], ...'
'enable', 'on');
if ~isempty(handles.TrueDat) % some points were grabbed
varNames = fieldnames(handles.savedVars);
varNames{end + 1} = findNextVarNameFcn(varNames);
handles.savedVars.(varNames{end}) = handles.TrueDat;
assignin('base', varNames{end},
handles.savedVars.(varNames{end}));
showAllVarsFcn(handles);
end

% change to NORMAL state
handles.state = 'normal';
end

guidata(obj, handles);

%---------------------------------------------------------------

% showAllVarsFcn
% this function shows all data set variables that exist in the GRABIT workspace in the variable listbox.

varNames = fieldnames(handles.savedVars);

if isempty(varNames)
  listboxStr = {''};
else
  for id = 1:length(varNames)
    [m, n] = size(handles.savedVars.(varNames{id}));
    listboxStr{id} = sprintf('%s [%dx%d]', varNames{id}, m, n);
  end
end

set(handles.VariableList, 'string', listboxStr, 'value', length(listboxStr));

% this helper function determines the next available variable name by checking the existing variable names in the base workspace and GRABIT workspace.

wsVarNames = evalin('base', 'who');
vars = unique([wsVarNames(:); varNames(:)]);
idx = 1;
while 1
  if isempty(strmatch(sprintf('Data%03d', idx), vars, 'exact'))
    newVarName = sprintf('Data%03d', idx);
    return;
  end
  idx = idx + 1;
end
else
    idx = idx + 1;
end}
end

function zoomBtnFcn(varargin)
% this function toggles the zoom state
obj = varargin{1};
handles = guidata(obj);

switch get(obj, 'value')
    case 0
        set(obj, ...
            'value', 1, ...
            'backgroundcolor', handles.bgcolor4, ...
            'cdata', handles.zoomIconDown);
        udata.titlestring = get(get(handles.ImageAxis, 'Title'), ...
            'string');
        udata.btnstate = get([handles.LoadImageBtn, ...
            handles.CalibrateImageBtn, ...
            handles.GrabPointsBtn, ...
            handles.VariableList, ...
            handles.SaveAs, ...
            handles.Rename, ...
            handles.Delete], 'enable');
        udata.imgstate = get(handles.ImageAxis, 'ButtonDownFcn');
        udata.handlevisibility = get(handles.ImageAxis, ...
            'handlevisibility');
        set(obj, 'userdata', udata);
        set(get(handles.ImageAxis, 'Title'), 'string', 'Zoom ON');
        set([handles.LoadImageBtn, ...
            handles.CalibrateImageBtn, ...
            handles.GrabPointsBtn, ...
            handles.VariableList, ...}
handles.SaveAs, ...
handles.Rename, ...
handles.Delete], 'enable', 'off');
set(handles.ImageAxis, 'ButtonDownFcn', '');
zoom('on');
handles.curPointerData.CData = get(handles.GrabitGUI, 'PointerShapeCData');
handles.curPointerData.HotSpot = get(handles.GrabitGUI, 'PointerShapeHotSpot');
set(handles.GrabitGUI, ...
'PointerShapeCData', handles.zoomPointer, ...
'PointerShapeHotSpot', handles.zoomPointerHotSpot, ...
'WindowButtonMotionFcn', {@pointerFcn, handles, 'custom'}, ...
'keypressfcn', ';'); % this prevents switching to command window

% this seems necessary in some versions of Matlab
set(handles.ImageAxis, 'handlevisibility', 'on');

case 1
set(obj, ...
'value', 0, ...
'backgroundcolor', handles.bgcolor3, ...
'cdata', handles.zoomIconUp);
zoom('off');

%-----------------------------------------------------------
% If zoom created a compact view window, expand it to fill the whole axes.
%-----------------------------------------------------------

xl = get(handles.ImageAxis, 'xlim'); xrng = diff(xl);
yl = get(handles.ImageAxis, 'ylim'); yrng = diff(yl);
if abs(yrng/xrng - handles.imAxRatio) > .01 % wrong axes ratio
    if yrng/xrng > handles.imAxRatio
        xrng = yrng / handles.imAxRatio;
        xl = mean(xl) + [-0.5, 0.5] * xrng;
    else
        yrng = xrng * handles.imAxRatio;
        yl = mean(yl) + [-0.5, 0.5] * yrng;
    end
set(handles.ImageAxis, 'xlim', xl, 'ylim', yl);
end
udata = get(obj, 'userdata');
set(get(handles.ImageAxis, 'Title'), 'string', udata.titlestring);
set(handles.ImageAxis, 'handlevisibility', udata.handlevisibility);
set(handles.LoadImageBtn, ...
    handles.CalibrateImageBtn, ...
    handles.GrabPointsBtn, ...
    handles.VariableList, ...
    handles.SaveAs, ...
    handles.Rename, ...
    handles.Delete], {'enable'}, udata.btnstate);
set(handles.ImageAxis, 'ButtonDownFcn', udata.imgstate);
set(handles.GrabitGUI, ...
    'PointerShapeCData'    , handles.curPointerData.CData,
    ...
    'PointerShapeHotSpot'  , handles.curPointerData.HotSpot,
    ...
    'WindowButtonMotionFcn', {@pointerFcn, handles, handles.curPointer}, ...
    'keypressfcn'          , @keyPressFcn);
end
guida(obj, handles);
yl = get(handles.ImageAxis, 'ylim');
if pt(1,1) > xl(1) && pt(1,1) < xl(2) && pt(1,2) > yl(1) &&
    pt(1,2) < yl(2)
    set(handles.GrabitGUI, 'pointer', ptr);
else
    set(handles.GrabitGUI, 'pointer', 'arrow');
end

function figResizeFcn(varargin)
% this function makes sure the axis fills the whole axes extent

obj = varargin{1};
handles = guidata(obj);

axis(handles.ImageAxis, 'equal');
handles.imAxRatio = diff(get(handles.ImageAxis, 'ylim')) / ...
    diff(get(handles.ImageAxis, 'xlim'));
handles.ImLimits = [get(handles.ImageAxis, 'xlim'); ...
    get(handles.ImageAxis, 'ylim')];
guidata(obj, handles);

function keyPressFcn(varargin)
% this is for the keyboard shortcuts. During 'grab' mode,
<backspace>
% deletes the last point clicked and <return> ends the 'grab' mode.

obj = varargin{1};

handles = guidata(obj);

if ~isempty(handles.I)
    k = lower(get(obj, 'CurrentKey'));

    switch k
    case 'a'
        % zoom in
        xl = get(handles.ImageAxis, 'xlim'); xrng = diff(xl);
        yl = get(handles.ImageAxis, 'ylim'); yrng = diff(yl);

        % prevent zooming in too much.
        % set the limit to 64x zoom.
        if xrng >= size(handles.I, 2)/64*2
            % animate zoom
            for id = 0:0.2:1
                set(handles.ImageAxis, ...
                    'xlim', xl + id * xrng / 4 * [1, -1], ...
                    'ylim', yl + id * yrng / 4 * [1, -1]);
                drawnow;
            end
        end
    end

    case 'z'
        % zoom out
        xl = get(handles.ImageAxis, 'xlim'); xrng = diff(xl);
        yl = get(handles.ImageAxis, 'ylim'); yrng = diff(yl);

        % animate zoom
        for id = 0:0.2:1
            set(handles.ImageAxis, ...
                'xlim', xl + id * xrng / 2 * [-1, 1], ...
                'ylim', yl + id * yrng / 2 * [-1, 1]);
            drawnow;
        end
    end

    case 'space'
        % reset view
        resetViewFcn(handles.ResetViewBtn);
    end

    case {'backspace', 'delete'}

        switch handles.state
        case 'grab'
            if ~handles.isPanning

            end
        end
    end
end
if isempty(handles.ImDat)
    return;
else
    handles.ImDat(end, :) = [];
    handles.TrueDat(end, :) = [];
end

set(handles.PreviewLine, ...
    'xdata', handles.TrueDat(:, 1), ...
    'ydata', handles.TrueDat(:, 2));
set(handles.ImageLine, ...
    'xdata', handles.ImDat(:, 1), ...
    'ydata', handles.ImDat(:, 2));

set(handles.GrabPointsBtn, 'string',
    sprintf('Grabbing Points (%d)', size(handles.ImDat, 1)));

guidata(obj, handles);
end
end
case {'return', 'enter'}
    switch handles.state
        case 'grab'
            grabPointsFcn(handles.GrabPointsBtn);
        end
    end
end

function resetViewFcn(varargin)
% this function resets the view

obj = varargin{1};

end

handles = guidata(obj);

if ~isempty(handles.I)
    xl = get(handles.ImageAxis, 'xlim');
    yl = get(handles.ImageAxis, 'ylim');
    xd = (handles.ImLimits(1, :) - xl) / 10;
    yd = (handles.ImLimits(2, :) - yl) / 10;
% animate zoom
    for id = 0:10
        set(handles.ImageAxis, ...
            'xlim', xl + id * xd, ...
            'ylim', yl + id * yd);
        drawnow;
    end
end

% take focus away
loseFocusFcn(handles)

function loseFocusFcn(handles)
% attempt to take focus away by setting the ENABLE property to off and then
% back to the original setting
settings = get([handles.LoadImageBtn, ...  
    handles.ResetViewBtn, ...  
    handles.VariableList, ...  
    handles.SaveAs, ...  
    handles.Rename, ...  
    handles.Delete], ...
'enable');
set([handles.LoadImageBtn, ...  
    handles.ResetViewBtn, ...  
    handles.VariableList, ...  
    handles.SaveAs, ...  
    handles.Rename, ...
handles.Delete, ...

'enable', 'off');
drawnow;
set({[handles.LoadImageBtn, ...
    handles.ResetViewBtn, ...
    handles.VariableList, ...
    handles.SaveAs, ...
    handles.Rename, ...
    handles.Delete], ...
{'enable'}, settings);

%---------------------------------------------------------------
--
%---------------------------------------------------------------
--
function winBtnDownFcn
% this function is called when the mouse click initiates

obj = varargin{1};
handles = guidata(obj);

if strcmpi(get(handles.timer, 'Running'), 'on') || isempty(handles.I)
    return;
end

handles.CurrentPointAxes = get(handles.ImageAxis, 'CurrentPoint');
handles.CurrentPointFig = get(handles.GrabitGUI, 'CurrentPoint');

switch get(handles.GrabitGUI, 'SelectionType')
    case 'normal'
        set(handles.CoordinateEdit, 'visible', 'off');
        id = find(isnan(handles.CalibPts));
        if ~isempty(id)
            set(handles.CalibPtsH(id(1),:), 'xdata', NaN, 'ydata', NaN);
        end
end
% first call winBtnMotionPauseFcn to prevent immediate
click-n-drag
set(handles.GrabitGUI, ...
    'WindowButtonMotionFcn', ...
    @winBtnMotionPauseFcn, handles, handles.CurrentPointAxes(1,1:2), clock));

case 'alt'
    set(handles.CoordinateEdit, 'visible', 'off');
    id = find(isnan(handles.CalibPts));
    if ~isempty(id)
        set(handles.CalibPtsH(id(1),:), 'xdata', NaN, 'ydata', NaN);
    end
    xl = get(handles.ImageAxis, 'XLim');midX = (xl(1)+xl(2))/2;
    yl = get(handles.ImageAxis, 'YLim');midY = (yl(1)+yl(2))/2;
    figPos = get(handles.GrabitGUI, 'Position');
    handles.curTitle = get(get(handles.ImageAxis, 'title'),
    'string');
    set(handles.GrabitGUI, ...
        'Pointer', 'custom', ...
        'PointerShapeCData' , handles.zoomInOutPointer, ...
        'PointerShapeHotSpot' , handles.zoomInOutPointerHotSpot, ...
        'WindowButtonMotionFcn' , @zoomMotionFcn, handles, ...
        get(handles.GrabitGUI,
        'CurrentPoint'), ...
        figPos(4), ...
        size(handles.I, 2), ...
        midX, ...
        midY, ...
        diff(xl)/2, ...
        diff(yl)/2));
    set(get(handles.ImageAxis, 'title'), 'string',
    'Zooming...');
end
guidata(obj, handles);
function zoomMotionFcn(varargin)
% this performs the click-n-drag zooming function. The pointer location
% relative to the initial point determines the amount of zoom (in or out).

[obj, handles, initPt, figHt, horizPx, midX, ...
  midY, rngXhalf, rngYhalf] = splitvar(varargin([1, 3:end]));

pt = get(obj, 'CurrentPoint');

% get relative pointer location (y-coord only).
% power law allows for the inverse to work:
%   C^(x) * C^(-x) = 1
% Choose C to get "appropriate" zoom factor
r = 30 ^ ((initPt(2) - pt(2)) / figHt);

% make sure it doesn't zoom in too much.
% the limit is based on the size of the original image.
% set limit to 64x zoom.
if r < horizPx/64/rngXhalf/2 % stop zoom
    set(get(handles.ImageAxis, 'title'), 'string', 'Max Zoom Reached');
    set(obj, ...
      'Pointer', 'arrow', ...
      'WindowButtonMotionFcn', '');
else
    set(handles.ImageAxis, ...
      'XLim', [midX - r * rngXhalf, midX + r * rngXhalf], ...
      'YLim', [midY - r * rngYhalf, midY + r * rngYhalf]);
end

%---------------------------------------------------------------------
%---------------------------------------------------------------------
% winBtnMotionPauseFcn
%---------------------------------------------------------------------
%---------------------------------------------------------------------
function winBtnMotionPauseFcn(varargin)
% this prevents click-n-drag from happening for X seconds. This is useful
% because users may move the mouse as they are clicking.

[object, handles, xy, c] = splitvar(varargin([1, 3:end]));

if etime(clock, c) > .15 % waits .15 seconds before dragging occurs
    set(object, ...
        'Pointer', 'custom', ...
        'PointerShapeCData', handles.closedHandPointer, ...
        'PointerShapeHotSpot', handles.closedHandPointerHotSpot);
    set(object, 'WindowButtonMotionFcn', {@winBtnMotionFcn, handles.ImageAxis, xy});
    handles.curTitle = get(get(handles.ImageAxis, 'title'), 'string');
    set(get(handles.ImageAxis, 'title'), 'string', 'Panning...');
    handles.isPanning = true;
    guidata(object, handles);
    end

function winBtnMotionFcn(varargin)
% this function is called when click-n-drag (panning) is happening

[axH, xy] = splitvar(varargin(3:4));

pt = get(axH, 'CurrentPoint');
set(axH, ...
    'xlim', get(axH, 'xlim') + (xy(1) - pt(1,1)), ...
    'ylim', get(axH, 'ylim') + (xy(2) - pt(1,2)));
function winBtnUpFcn(varargin)
% this is called when the mouse button is released. It initiates
% the button
% down timer object (see btnUpTimerFcn). This timer waits for
% 0.2 seconds
% before any action is taken. This is in order to detect a
double click. If
% a double click is detected, the single click action is NOT
% executed.

obj = varargin{1};

handles = guidata(obj);
if ~isempty(handles.I) % there is an image displayed
    switch get(obj, 'SelectionType')
        case 'normal'
            if strcmpi(get(handles.timer, 'Running'), 'off')
                % start the timer which waits some time to see if
double-clicking
                % occurs
                start(handles.timer);
            end

            set(obj, ...
                'Pointer', handles.curPointer, ...
                'PointerShapeCData', handles.curPointerData.CData, ...
                'PointerShapeHotSpot', handles.curPointerData.HotSpot, ...
                'WindowButtonMotionFcn', @pointerFcn, handles, ...
                handles.curPointer);

            set(get(handles.ImageAxis, 'title'), 'string', ...
                handles.curTitle);
        end

    case 'alt'
        set(obj, ...

'Pointer', handles.curPointer, ...
'PointerShapeCData', handles.curPointerData.CData, ...
'PointerShapeHotSpot', handles.curPointerData.HotSpot, ...
'WindowButtonMotionFcn', {@pointerFcn, handles, handles.curPointer});

set(get(handles.ImageAxis, 'title'), 'string', handles.curTitle);
end
end

%-------------------------------------------------------------------
% btnUpTimerFcn
%-------------------------------------------------------------------

function btnUpTimerFcn(varargin)
figH = varargin{3};
handles = guidata(figH);
switch get(handles.GrabitGUI, 'SelectionType')
case 'open' % double click
  % this will center the view

  % get current units
  un = get([0, handles.GrabitGUI, handles.ImageAxis], 'units');

  set([0, handles.GrabitGUI, handles.ImageAxis], 'units', 'pixels');
  pt = get(0, 'PointerLocation');
  figPos = get(handles.GrabitGUI, 'Position');
  pt2 = pt - figPos(1:2);
  axPos = get(handles.ImageAxis, 'position');

  % check to see if pointer is inside the image axes
  if pt2(1) > axPos(1) && pt2(1) < axPos(1)+axPos(3) && ...
    pt2(2) > axPos(2) && pt2(2) < axPos(2)+axPos(4)
xl = get(handles.ImageAxis, 'xlim'); xrng = diff(xl);
yl = get(handles.ImageAxis, 'ylim'); yrng = diff(yl);
x  = (pt2(1)-axPos(1))/axPos(3)*xrng+xl(1);
y  = (axPos(2)+axPos(4)-pt2(2))/axPos(4)*yrng+yl(1);

% animate (slide) to the new location. this may give a
better
% visual perception of the view change

% determine how fast to animate, depending on the location
of the
% pointer
interval = ceil(norm((figPos(1:2)+axPos(1:2)+axPos(3:4)/2)
- pt)/30);

if interval
    ld = (([x, y] - [xrng, yrng]/2) - [xl(1),
yl(1)])/interval;
pd = ((figPos(1:2)+axPos(1:2)+axPos(3:4)/2) -
pt)/interval;
else
    % interval == 0, meaning it's the same point
    ld = [0, 0];
pd = [0, 0];
end

for id = 0:interval
    set(handles.ImageAxis, ...
        'xlim', xl + id * ld(1), ...
        'ylim', yl + id * ld(2));

    % center the pointer location
    set(0, ...
        'PointerLocation', pt + id * pd);
drawnow;
end

% reset UNITS
set([0, handles.GrabitGUI, handles.ImageAxis], {'units'},
un);

case 'normal'  % single click

switch handles.state
    case 'grab'
if ~handles.isPanning && ~isempty(handles.CurrentPointAxes)
    calib = handles.CalibVals;
    
    X = handles.CurrentPointAxes(1, 1);
    Y = handles.CurrentPointAxes(1, 2);
    
    % the point on the image
    handles.ImDat(end + 1, :) = [X, Y];
    
    % the true data point coordinates
    handles.TrueDat(end + 1, :) = ([calib.Xo; calib.Yo] + ...
        [calib.e1 calib.e2] \ [X - calib.Xxo; Y - calib.Yyo])';

    set(handles.PreviewLine, ...
        'xdata', handles.TrueDat(:, 1), ...
        'ydata', handles.TrueDat(:, 2));

    set(handles.ImageLine, ...
        'xdata', handles.ImDat(:, 1), ...
        'ydata', handles.ImDat(:, 2));

    set(handles.GrabPointsBtn, 'string', sprintf('Grabbing Points (%d)', size(handles.ImDat, 1)));

end

case 'calibration'
    if ~handles.isPanning && ~isempty(handles.CurrentPointAxes)
        id = find(isnan(handles.CalibPts));
        pt = handles.CurrentPointAxes;
        set(handles.CalibPtsH(id(1),:), 'xdata', pt(1,1), ...
            'ydata', pt(1,2));
        handles.CalibPtsIm(id(1),:) = pt(1,1:2);
        set(handles.CoordinateEdit, ...
            'units', 'pixels', ...
            'position', [handles.CurrentPointFig+5, 5], 100,
            25], ...
            'visible', 'on', ...
            'string', 'Enter Value');
    end
end
end

handles.CurrentPointAxes = [];
handles.CurrentPointFig = []; 
handles.isPanning = false;

guidata(handles.GrabitGUI, handles);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--------------
--------------
--------------
--------------

% coordinateEditFcn
%--------------
--------------
--------------
--------------

function coordinateEditFcn(varargin)
% this function is called when a value is entered in the edit box for
% calibration values

obj = varargin{1};

handles = guidata(obj);

strs = {'Click on the MAXIMUM value of x-axis', 'Click on the ORIGIN of y-axis', 'Click on the MAXIMUM value of y-axis'};
ptrs = {'xmPointer', 'yoPointer', 'ymPointer'};
hs = {'hXoValue', 'hXmValue', 'hYoValue', 'hYmValue'};

val = str2double(get(obj, 'string'));
if ~isnan(val)
    id = find(isnan(handles.CalibPts));
    if ~isempty(id)
        handles.CalibPts(id(1)) = val;
        set(handles.(hs{id(1)}), 'string', sprintf(' %g', val));
        set(obj, 'visible', 'off');
        handles.curTitle = strs{id(1)};
        set(get(handles.ImageAxis, 'title'), 'string', strs{id(1)});
        if id(1) == 4
            handles = calculateCalibrationFcn(handles);
        else
            handles.curPointerData.CData = handles.(ptrs{id(1)});
            handles.curPointerData.HotSpot = handles.([ptrs{id(1)} 'HotSpot']);
        end
    end
end
'PointerShapeCData' , handles.curPointerData.CData,
...
'PointerShapeHotSpot' , handles.curPointerData.HotSpot);
else
    set(obj, 'string', '');
else
    set(obj, 'string', '');
end

guidata(handles.GrabitGUI, handles);

%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
% calculateCalibrationFcn
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------

function handles = calculateCalibrationFcn(handles)
% this function calculates the calibration values

Xxo = handles.CalibPtsIm(1,1);
Yxo = handles.CalibPtsIm(1,2);
Xxm = handles.CalibPtsIm(2,1);
Yxm = handles.CalibPtsIm(2,2);
Xyo = handles.CalibPtsIm(3,1);
Yyo = handles.CalibPtsIm(3,2);
Xym = handles.CalibPtsIm(4,1);
Yym = handles.CalibPtsIm(4,2);

v1 = [Xxm - Xxo; Yxm - Yxo];
v2 = [Xym - Xyo; Yym - Yyo];

Xo = handles.CalibPts(1);
Xm = handles.CalibPts(2);
Yo = handles.CalibPts(3);
Ym = handles.CalibPts(4);

Xlength = Xm - Xo;
Ylength = Ym - Yo;

% the basis vectors in the MATLAB domain
\[ e_1 = \frac{v_1}{X_{\text{length}}}; \]
\[ e_2 = \frac{v_2}{Y_{\text{length}}}; \]

% rearrage axes
\[ C = [e_1 \ e_2] \ \ [X_{yo} - X_{xo}; Y_{yo} - Y_{xo}]; \]
\[ \text{blahX} = [X_{xo}; Y_{xo}] + C(2) \times e_2; X_{xo} = \text{blahX}(1); Y_{xo} = \text{blahX}(2); \]
\[ \text{blahY} = [X_{yo}; Y_{yo}] - C(1) \times e_1; X_{yo} = \text{blahY}(1); Y_{yo} = \text{blahY}(2); \]

\[ \text{calib.Xo} = X_{o}; \]
\[ \text{calib.Xm} = X_{m}; \]
\[ \text{calib.Yo} = Y_{o}; \]
\[ \text{calib.Ym} = Y_{m}; \]
\[ \text{calib.e1} = e_1; \]
\[ \text{calib.e2} = e_2; \]
\[ \text{calib.Xxo} = X_{xo}; \]
\[ \text{calib.Yyo} = Y_{yo}; \]

\text{handles.CalibVals} = \text{calib};
\set{\text{handles.GrabPointsBtn}, ...}
\set{\text{'enable'}, \text{'inactive'}};
\set{\text{handles.CalibrateImageBtn}, ...}
\set{\text{'value'}, 0, ...}
\set{\text{'bgcolorcolor'}, \text{handles.bgcolor3}, ...}
\set{\text{'enable'}, \text{'inactive'}, ...}
\set{\text{'string'}, \text{'Re-Calibrate'}};

\text{handles.state} = \text{'normal'};

\text{handles.curPointer} = \text{'arrow'};
\set{\text{handles.GrabitGUI}, ...}
\set{\text{'WindowButtonMotionFcn'}, \{@pointerFcn, \text{handles, handles.curPointer}});

\set{\text{handles.LoadImageBtn}, ...}
\set{\text{handles.SaveAs}, ...}
\set{\text{handles.Rename}, ...}
\set{\text{handles.Delete}, ...}
\set{\text{'enable'}, \text{'on'}};

%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
% % splitvar
%---------------------------------------------------------------
%---------------------------------------------------------------
%---------------------------------------------------------------
%------------------------------------------------------------------------function varargout = splitvar(varargout)
% this function splits input arguments into individual
variables.
%------------------------------------------------------------------------%------------------------------------------------------------------------% createSampleImageFcn
%------------------------------------------------------------------------%------------------------------------------------------------------------function fname = createSampleImageFcn
% this function creates a temporary image file in the temp
directory by
% loading in a sample binary image data.
str = [
'4749463839611002CA01F700000000008000000080008080000000808000800
08080C0'
'C0C0C0DCC0A6CAF0402000602000802000A02000C02000E0200000400020400
0404000'
'604000804000A04000C04000E04000006000206000406000606000806000A06
000C060'
'00E06000008000208000408000608000808000A08000C08000E0800000A0002
0A00040'
'A00060A00080A000A0A000C0A000E0A00000C00020C00040C00060C00080C00
0A0C000'
'C0C000E0C00000E00020E00040E00060E00080E000A0E000C0E000E0E000000
0402000'
'40400040600040800040A00040C00040E000400020402020404020406020408
02040A0'
'2040C02040E02040004040204040404040604040804040A04040C04040E0404
0006040'
211


\'6840E8323D69E8660C0B5952098A55B256E8400BE1AAC4B8A8B502A96851E92C6B5CED\'

\'2753CEDE9170B31DBB862A57BECE755CB78848F4DEC7C0BE16362759C348F7A66758\'

\'C6B64478B1B51EC491B3BD9AF4496B094C5AC562C9B59CE72659A55EB6C68A5F92205\'

\'D255B4A7FD89B97C9754D4B6D6273503A16B655BA17ACED6B6B7C56D6E75BB5BDEF6D6\'

\'B7BF056E70854BA78000003B\'

];

% convert hex to dec
s = hex2dec(reshape(deblank(reshape(str\', 1, [])), 2, []));

fname = [tempname '.gif'];
fid = fopen(fname, 'w');
if fid > 0
    fwrite(fid, s);
    fclose(fid);
else
    fname = '\';
    errordlg('Error trying to create sample image.');
end
Appendix E - Derivation of source/sink strength ratio estimate

To obtain the sink spacing, the source is presumed to be placed at the origin (center of the porous canopy), then a sink is placed between the source and recovered wake velocity.

Here in this section, the terms disk, cylinder and tube are defined.

*Disk:* an infinitely thin disk placed perpendicular to the incoming flow.

*Cylinder:* a disk that has thickness of length $D$ in the directions perpendicular and horizontal to the flow.

*Tube:* the domain where the fluid and object is present.

In Figure 1, the blue arrows indicate the freestream velocity from left to right, the red ellipse represents the Rankine oval that is created as a result of the source and sink pair. The red points along the $y=0$ represents the stagnation points. Figure 2 gives a more detailed representation of the streamlines for a similar flow with incoming velocity from right to left.

*Figure 206: Rankine oval [2].
The centerline velocity for the potential flow solution of source and equal sink strength is well known according to introductory fluid mechanics texts.

![Figure 207: Coordinate placement of source and sink](image)

**Relationship between Actuator Disk Theory and Porous Flow**

Actuator disk theory utilizes the concept of representing the flow around a propeller. In this approach, the propeller is assumed to be stationary and the flow is steady.

However, the flow around a propeller creates a velocity deficit. At the disk, there is a change in the magnitude of the pressure on both sides.

This flow behavior is similar in porous flow. The flow through a porous cylinder causes a drop in velocity which is due to a decrease in the momentum of the wind that flows through it.

The porous body also increases the turbulence intensity of the wind as it exits the porous cylinder. As the wind enters the porous object, it will absorb some of the momentum in the air and later
release this air which shows up as a recirculation zone and the increased turbulence intensity is after this zone.

Using actuator disk theory (or momentum theory) we can represent the porous cylinder as a disk and the flow domain as a streamtube.

(1) Is the inlet where the induced velocity (freestream) enters the streamtube
(2) Is a point in front of the disk (windward of disk)
(3) Is a point behind the disk (leeward)
(4) Is the outlet, where the airflow exits

If we consider the mass flow rate across each point in the flow domain, under conservation of mass

\[ \dot{m} = \rho U_1 A_1 = \rho U_2 A_2 = \rho U_3 A_3 = \rho U_4 A_4 \]  \hspace{1cm} (E.1)

It is worth mentioning that since the porous cylinder is considered as a disk

\[ U_2 = U_3 \]  \hspace{1cm} (E.2)

Thus,

\[ \dot{m} = \rho U_1 A_1 = \rho U_2 A_2 = \rho U_4 A_4 \]  \hspace{1cm} (E.3)

To account for the pressure change across the disk, the conservation of momentum is applied via the Bernoulli’s equation across position (1) to (2)

\[ p_1 + \frac{1}{2} \rho U_1^2 = p_2 + \frac{1}{2} \rho U_2^2 \]

\[ p_2 = p_1 + \frac{1}{2} \rho U_1^2 - \frac{1}{2} \rho U_2^2 \]  \hspace{1cm} (E.4)

And across (3) to (4)
\[ p_3 + \frac{1}{2} \rho U_3^2 = p_4 + \frac{1}{2} \rho U_4^2 \]

\[ p_3 = p_4 + \frac{1}{2} \rho U_4^2 - \frac{1}{2} \rho U_3^2 \]  \hspace{1cm} (E.5)

Axial induction factor (or interference factor, \( a \)) is the fractional decrease in wind velocity between the freestream and the energy extraction device (turbine or tree).

Describing the induction factor at the disk,

\[ a = \frac{U_1 - U_2}{U_1} \]  \hspace{1cm} (E.6)

\[ U_2 = -U_1 a + U_1 \]

\[ U_2 = U_1(1 - a) \]  \hspace{1cm} (E.7)

At the wake position, \( (4) \)

\[ a = \frac{U_3 - U_4}{U_1} = \frac{U_1(1-a) - U_4}{U_1} \]  \hspace{1cm} (E.8)

\[ U_1 a = U_1(1 - a) - U_4 \]

\[ U_4 = U_1(1 - 2a) \]  \hspace{1cm} (E.9)

Also, from the conservation of linear momentum, the forces acting on the disk, \( F \), is as a result of the pressure difference exerted on the disk area, \( A_2 \).

\[ F = (p_2 - p_3)A_2 \]  \hspace{1cm} (E.10)

The force on the disk results in a drop in the freestream velocity across the domain. This change in momentum of the air across the domain generates the thrust in the air as it exits the disk, provided the pressure at the inlet and outlet of the streamtube are the same, i.e.
\[ p_1 = p_4 \quad \text{(E.11)} \]

\[ F = \dot{m}U_4 - \dot{m}U_1 \quad \text{(E.12)} \]

Thus,

\[ F = \dot{m}(U_4 - U_1) = (p_2 - p_3)A_2 \quad \text{(E.13)} \]

Combing equations (E.1), (E.10) and (E.13), we have

\[
\left[ \left( p_1 + \frac{1}{2} \rho U_1^2 - \frac{1}{2} \rho U_2^2 \right) - \left( p_4 + \frac{1}{2} \rho U_4^2 - \frac{1}{2} \rho U_3^2 \right) \right] A_2 = (U_4 - U_1)(\rho U_2A_2)
\]

\[
\left[ p_1 + \frac{1}{2} \rho U_1^2 - \frac{1}{2} \rho U_2^2 - p_1 + \frac{1}{2} \rho U_4^2 - \frac{1}{2} \rho U_2^2 \right] A_2 = (U_4 - U_1)(\rho U_2A_2)
\]

\[
\left( \frac{1}{2} \rho U_1^2 - \frac{1}{2} \rho U_4^2 \right) A_2 = \rho U_2A_2(U_4 - U_1)
\]

Dividing both sides by \( \rho A_2 \), we have

\[
\frac{1}{2} \left( U_1^2 - U_4^2 \right) = U_2(U_4 - U_1)
\]

Using the algebraic relationship, \( a^2 - b^2 = (a + b)(a - b) \)

\[
\frac{1}{2} \left( U_4 - U_1 \right) \left( U_4 + U_1 \right) = U_2(U_4 - U_1)
\]

Divide both sides by \( U_4 - U_1 \)

\[
U_2 = \frac{(U_4 + U_1)}{2} \quad \text{(E.14)}
\]
The previous equations have two unknowns and a free parameter $s_s$. This means that the value used for $s_s$ would change based on the source/sink strength ratios, i.e. it is deterministic.

To aid in estimation, an initial $s_s$ is initially guessed and then the simultaneous equation that follow can be solved.

Let us assume that, $s_s = 2D$, $r_u = 3D$ and $r_w = 10D$

Equation (E.18) and (E.19) becomes

$$U_2 = U_1 - \frac{m_{so}}{2\pi(3D)} + \frac{m_{si}}{2\pi(3D + 2D)}$$

$$U_4 = U_1 + \frac{m_{so}}{2\pi(10D)} - \frac{m_{si}}{2\pi(10D - 2D)}$$

$$U_2 = U_1 - \frac{m_{so}}{6\pi D} + \frac{m_{si}}{10\pi D}$$ \hspace{1cm} (E.20)

$$U_4 = U_1 + \frac{m_{so}}{20\pi D} - \frac{m_{si}}{16\pi D}$$ \hspace{1cm} (E.21)

$U_4$ can be obtained from the simulated velocity field. Using the simulated case at porosity (inertial resistance) $= 15 \text{ m}^{-1}$,

$$U_4 = 0.6U_1$$ \hspace{1cm} (E.22)

From equation (16),

$$U_2 = \frac{U_1 + 0.6U_1}{2} = 0.8U_1$$ \hspace{1cm} (E.23)

Substituting $U_4$ and $U_2$ into equation (22) and (23), we have

$$0.8U_1 = U_1 - \frac{m_{so}}{6\pi D} + \frac{m_{si}}{10\pi D}$$
\[
0.6U_1 = U_1 + \frac{m_{so}}{20\pi D} - \frac{m_{si}}{16\pi D}
\]

Simplifying the above equations and multiplying both sides by \(\pi D\)

\[-0.2\pi D U_1 = -\frac{m_{so}}{6} + \frac{m_{si}}{10} \quad (E.24)\]

\[-0.4\pi D U_1 = \frac{m_{so}}{20} - \frac{m_{si}}{16} \quad (E.25)\]

Multiplying both sides by 10 in equation (E.24) and by 20 in equation (E.25)

\[-6.283D U_1 = -\frac{10m_{so}}{6} + m_{si} \quad (E.26)\]

\[-25.133D U_1 = m_{so} - \frac{20m_{si}}{16} \quad (E.27)\]

Make \(m_{si}\) subject of the formula in equation (26)

\[m_{si} = 1.667m_{so} - 6.283D U_1 \quad (E.28)\]

Substitute into equation (E.27)

\[-25.133D U_1 = m_{so} - 1.25(1.667m_{so} - 6.283D U_1)\]

\[-25.133D U_1 - 7.854D U_1 = m_{so} - 2.084m_{so}\]

\[m_{so} = \frac{-32.987D U_1}{-1.084} = 30.4308D U_1 \quad (E.29)\]

Substitute value into equation (E.28)

\[m_{si} = 1.667(30.4308D U_1) - 6.283D U_1\]

\[m_{si} = 50.728D U_1 - 6.283D U_1 = 44.445D U_1\]

Ratio of source and sink strength for sink spacing of 2D at \(c_i = 15\)m\(^{-1}\)
\[ \frac{m_{sl}}{m_{so}} = 1.46 \]  \hspace{1cm} \text{(E.30)}

Using the preceding formulation for the potential flow estimation, any spacing distance chosen must be compared with the residual sum of squares error, total sum of squares error, R-squared and difference between the simulated porous velocity and the potential flow wake velocity.

\[ SS_{res} = \sum_i (y_i - f_i)^2 = \sum_i e^2 \]  \hspace{1cm} \text{(E.31)}

\[ SS_{tot} = \sum_i (y_i - f_i)^2 = \sum_i e^2 \]  \hspace{1cm} \text{(E.32)}

\[ R^2 \equiv 1 - \frac{SS_{res}}{SS_{tot}} \]  \hspace{1cm} \text{(E.33)}

\[ U_{4_{sim}} - U_{4_{pflow}} \]  \hspace{1cm} \text{(E.34)}
Appendix F – Tree classification in Minnesota

Table 11: Tree classification in Minnesota based on canopy shape at maturity

<table>
<thead>
<tr>
<th>S/N</th>
<th>Tree Name</th>
<th>Latin Name</th>
<th>Height (up to #ft)</th>
<th>Height (m)</th>
<th>Canopy shape</th>
<th>Tree description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Eastern White Pine</td>
<td>Pinus strobus</td>
<td>100</td>
<td>30</td>
<td>Pyramidal, oval</td>
<td>Needles grow in bundles of 5. Cone is up to 8 in. (20 cm) long.</td>
</tr>
<tr>
<td>2</td>
<td>Red Pine</td>
<td>Pinus resinosa</td>
<td>80</td>
<td>24</td>
<td>Irregular, oval, upright</td>
<td>Flexible needles are up to 6 in. (15 cm) long and grow in bundles of 2. Common in sandy soils. MN state tree</td>
</tr>
<tr>
<td>3</td>
<td>Jack Pine</td>
<td>Pinus banksiana</td>
<td>70</td>
<td>21</td>
<td>Open, irregular</td>
<td>Needles are in bundles of 2 and are widely forked. Curved cones grow laterally along branchlets.</td>
</tr>
<tr>
<td>4</td>
<td>Black Spruce</td>
<td>Picea mariana</td>
<td>75</td>
<td>23</td>
<td>Narrow, pyramidal, upright</td>
<td>Small to medium-sized tree has 4-sided needles that are about 0.5 in. (1.3 cm) long.</td>
</tr>
<tr>
<td>5</td>
<td>White Spruce</td>
<td>Picea glauca</td>
<td>75</td>
<td>23</td>
<td>Pyramidal, upright</td>
<td>4-sided, stiff blue-green needles curve upward along branchlets.</td>
</tr>
<tr>
<td>6</td>
<td>Tamarack</td>
<td>Larix laricina</td>
<td>80</td>
<td>24</td>
<td>Pyramidal</td>
<td>Needles grow in tufts. Stalkless cones grow upright. One of the only conifers to shed its needles in winter.</td>
</tr>
<tr>
<td>S/N</td>
<td>Tree Name</td>
<td>Latin Name</td>
<td>Height (up to #ft)</td>
<td>Height (m)</td>
<td>Canopy shape</td>
<td>Tree description</td>
</tr>
<tr>
<td>-----</td>
<td>------------------</td>
<td>------------------</td>
<td>--------------------</td>
<td>------------</td>
<td>------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>7</td>
<td>Eastern Hemlock</td>
<td><em>Tsuga canadensis</em></td>
<td>70</td>
<td>21</td>
<td>Pyramidal</td>
<td>Flat needles grow from 2 sides of twigs, parallel to the ground. Tip of tree usually drops.</td>
</tr>
<tr>
<td>8</td>
<td>Balsam Fir</td>
<td><em>Abies balsamea</em></td>
<td>60</td>
<td>18</td>
<td>Narrow, symmetrical, spire-shaped, dense crown</td>
<td>Flattened needles grow around branchlets in 2 rows. Purplish cones grow upright.</td>
</tr>
<tr>
<td>9</td>
<td>Northern Whitecedar</td>
<td><em>Thuga occidentalis</em></td>
<td>70</td>
<td>21</td>
<td>Compact, conical to pyramidal form</td>
<td>Drooping branchlets are covered with scale-like leaves. Wood is aromatic</td>
</tr>
<tr>
<td>10</td>
<td>Eastern Redcedar</td>
<td><em>Juniperus virginiana</em></td>
<td>60</td>
<td>18</td>
<td>Columnal, oval, pyramidal</td>
<td>4-sided branchlets are covered with overlapping, scale-like leaves. Fruit is a blue berry.</td>
</tr>
<tr>
<td>11</td>
<td>Bigtooth Aspen</td>
<td><em>Populus grandidentata</em></td>
<td>60</td>
<td>18</td>
<td>Pyramidal crown to spreading canopy</td>
<td>Leaves have large, blunt teeth along the edges. Flowers bloom in a long cluster</td>
</tr>
<tr>
<td>12</td>
<td>Balsam Poplar</td>
<td><em>Populus balsamifera</em></td>
<td>80</td>
<td>24</td>
<td>Narrow, somewhat columnar. Open with a few stout ascending branches</td>
<td>Drooping flower clusters are succeeded by oval capsules containing cottony seeds</td>
</tr>
<tr>
<td>S/N</td>
<td>Tree Name</td>
<td>Latin Name</td>
<td>Height (up to #ft)</td>
<td>Height (m)</td>
<td>Canopy shape</td>
<td>Tree description</td>
</tr>
<tr>
<td>-----</td>
<td>----------------</td>
<td>---------------------</td>
<td>-------------------</td>
<td>------------</td>
<td>--------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>13</td>
<td>Paper Birch</td>
<td>Betula papyrifera</td>
<td>70</td>
<td>21</td>
<td>Narrowly oval, open with branches ascending</td>
<td>White to red-brown bark peels off trunk in thin sheets.</td>
</tr>
<tr>
<td>14</td>
<td>Silver Maple</td>
<td>Acer saccharinum</td>
<td>80</td>
<td>24</td>
<td>Vase</td>
<td>Note short trunk and spreading crown. 5-lobed leaves are silvery beneath</td>
</tr>
<tr>
<td>15</td>
<td>Yellow Birch</td>
<td>Betula alleghaniensis</td>
<td>100</td>
<td>30</td>
<td>Round</td>
<td>Bark is red to yellowish and peels off in strips. Cone-like oval fruit grows erect on branchlets.</td>
</tr>
<tr>
<td>16</td>
<td>Eastern Cottonwood</td>
<td>Populus deltoides</td>
<td>100</td>
<td>30</td>
<td>Irregular, pyramidal, round</td>
<td>Leaves are up to 7 in. (18 cm) long. Flowers are succeeded by capsules containing seeds with cottony 'tails'.</td>
</tr>
<tr>
<td>17</td>
<td>Mountain Maple</td>
<td>Acer spicatum</td>
<td>25</td>
<td>7.6</td>
<td>Irregular crown of small upright branches</td>
<td>Leaves usually have 3 lobes. Winged seed pairs are red in summer, brown in autumn.</td>
</tr>
<tr>
<td>18</td>
<td>Red Maple</td>
<td>Acer rubrum</td>
<td>90</td>
<td>27</td>
<td>Oval, round upright</td>
<td>Leaves have 3-5 lobes and turn scarlet in autumn. Flowers are succeeded by red, winged seed pairs.</td>
</tr>
<tr>
<td>19</td>
<td>Boxelder</td>
<td>Acer negundo</td>
<td>60</td>
<td>18</td>
<td>Round and dense canopy</td>
<td>Leaves have 3-7 leaflets. Seeds are encased in paired keys</td>
</tr>
<tr>
<td>S/N</td>
<td>Tree Name</td>
<td>Latin Name</td>
<td>Height (up to #ft)</td>
<td>Height (m)</td>
<td>Canopy shape</td>
<td>Tree description</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------</td>
<td>-------------------</td>
<td>--------------------</td>
<td>------------</td>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>20</td>
<td>Sugar Maple</td>
<td><em>Acer saccharum</em></td>
<td>100</td>
<td>30</td>
<td>Oval round dense canopy</td>
<td>Leaves have 5 coarsely toothed lobes. Fruit is a winged seed pair. Tree sap is the source of maple syrup.</td>
</tr>
<tr>
<td>21</td>
<td>Northern Red Oak</td>
<td><em>Quercus rubra</em></td>
<td>90</td>
<td>27</td>
<td>Round dense canopy</td>
<td>Large tree has a rounded crown. Leaves have 7-11 spiny lobes.</td>
</tr>
<tr>
<td>22</td>
<td>White Oak</td>
<td><em>Quercus alba</em></td>
<td>100</td>
<td>30</td>
<td>Round, pyramidal with moderate density</td>
<td>Leaves have 5-9 rounded lobes. Acorn has shallow, scaly cup.</td>
</tr>
<tr>
<td>23</td>
<td>Bur Oak</td>
<td><em>Quercus macrocarpa</em></td>
<td>80</td>
<td>24</td>
<td>Round, spreading with dense canopy</td>
<td>Leaves have 5-9 lobes and are widest above the middle. The acorn cup is fringed.</td>
</tr>
<tr>
<td>24</td>
<td>Red Alder</td>
<td><em>Alnus spp.</em></td>
<td>100</td>
<td>30</td>
<td>Broad cone</td>
<td>Dark green leaves have toothed edges. Woody fruits are cone-like.</td>
</tr>
<tr>
<td>25</td>
<td>American Basswood</td>
<td><em>Tilia americana</em></td>
<td>100</td>
<td>30</td>
<td>Oval, pyramidal with dense canopy</td>
<td>Leaves are heart-shaped. Flowers and nutlets hang from narrow leafy bracts. Often multi-trunked.</td>
</tr>
<tr>
<td>26</td>
<td>American Elm</td>
<td><em>Ulmus americana</em></td>
<td>100</td>
<td>30</td>
<td>Vase with moderately dense canopy</td>
<td>Note vase-shaped profile. Leaves are toothed. Fruits have a papery collar and are notched at the tip.</td>
</tr>
<tr>
<td>S/N</td>
<td>Tree Name</td>
<td>Latin Name</td>
<td>Height (up to #ft)</td>
<td>Height (m)</td>
<td>Canopy shape</td>
<td>Tree description</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>-----------</td>
<td>--------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>27</td>
<td>Sumac</td>
<td><em>Rhus spp.</em></td>
<td>35</td>
<td>10.5</td>
<td>Upright, erect, round with moderately dense canopy</td>
<td>Long leaves have 10-30 leaflets and turn red in autumn. White flowers are succeeded by a cluster of red fruits.</td>
</tr>
<tr>
<td>28</td>
<td>Black Cherry</td>
<td><em>Prunus serotina</em></td>
<td>80</td>
<td>24</td>
<td>Oval with moderately dense canopy</td>
<td>Aromatic bark and leaves smell cherry-like. Dark berries have an oval stone inside.</td>
</tr>
<tr>
<td>29</td>
<td>Eastern Hophornbeam</td>
<td><em>Ostrya virginiana</em></td>
<td>50</td>
<td>15</td>
<td>Oval, round with moderately dense canopy</td>
<td>Trunk has sinewy, muscle-like bark. Hop-like fruits are hanging, cone-like clusters.</td>
</tr>
<tr>
<td>30</td>
<td>Black Ash</td>
<td><em>Fraxinus nigra</em></td>
<td>50</td>
<td>15</td>
<td>Variable, round, oval to oblong-ovate, densely branched</td>
<td>Leaves have 7-11 leaflets. Fruit is a winged seed.</td>
</tr>
<tr>
<td>31</td>
<td>American Mountain-ash</td>
<td><em>Sorbus americana</em></td>
<td>30</td>
<td>9</td>
<td>Slightly pyramidal, upright with a rounded crown</td>
<td>Leaves have 13-17 leaflets. Red fruits occur in dense clusters.</td>
</tr>
</tbody>
</table>
Appendix G - Experimental Validation method

To validate the results obtained in the numerical simulation performed in this thesis, a validation experiment was conducted in an open circuit wind tunnel using a 3D printed tree. A major source of error in the calibration measurement is the presence of the open slot in the top and bottom of the test section.

Measures were taken to reduce the error that arose due to this by covering the bottom openings with tape and the top with paper and tape.

The reason a tape and paper were used for the top was due to limited resources and the need to move the Pitot tube traversing mechanism to record pressure measurements.

Calibration Procedure

Calibration is an essential process in the execution of an experiment. It is to be planned with attention to the resources and constraints of the measurement method. [49, 103] has presented important guidelines that have been approached with ASTM/ASME guidelines.

The calibration of the open suction wind tunnel was performed by measuring the pressure differential along the vertical and horizontal sections of an undisturbed flow.

1. Measurements of the indoor and outdoor weather conditions
2. Check of wind tunnel connections
3. Seal gaps and holes in the test section,
4. Level the incline tube manometer.
5. Offset manometer zero by 0.1-0.5 in. $H_2O$
6. Set fan frequency to 5.9 Hz.
7. Measure the dynamic pressure of the still air using the Pitot Tube (average for 60 seconds)
8. Move probe position upward along the same x-axis, move from y = 0 to y = 12 in. Increments of 0.5 in.

The calibration was performed at three different fan speeds to obtain the velocity profile. The results of the calibration procedure are given in the next chapter.

**Experimental results**

To validate the numerical results obtained in this study, the experiment conducted in the wind tunnel around a 3D printed tree focused more on the geometric similarity rather than dynamic similarity of naturally occurring pine (*Pinus Strobus*) trees. The results herein show the calibration results of the velocity profile, the velocity profile around the tree (windward and leeward).

**Calibration Results**

The velocity profile measured at four locations along the test section (x = 0, 3, 6, 9 and 12 inches) is observed and reported.

Atmospheric conditions such as wet and dry bulb temperature, barometric pressure and current indoor and outdoor weather conditions were noted before the start of the experiment.

The calibration curves for the wind tunnel are provided below at three fan frequency (5.9 Hz, 14.4 Hz and 21.9 Hz).

In total, the calibration experiment was run 5 times, twice at the 5.9 Hz and 14.4 Hz frequency and once at the 21.9 Hz frequency.

To show the trend in the results of the velocity calibration below, an offset of 0.005 in. $H_2O$ was added to the measured value every 3 in (0.0762 m) distance from the inlet. In other words, at 3 in.
(0.0762 m) from the inlet, 0.005 in. H₂O is added, at 6 inches from the inlet; 0.010 in H₂O, at 9 in (0.2286 m); 0.015 in. H₂O, at 12 in. (0.3048 m); 0.020 in. H₂O.

During the calibration experiment, dynamic pressure readings were observed, measured and averaged during a 60 s interval.

![Figure 208: Calibration curve along test section at fan speed = 5.9 Hz](image-url)
Figure 209: Calibration curve along test section at fan speed = 14.4 Hz

Figure 210: Calibration curve along test section at fan speed = 21.9 Hz
The calibration curve at 5.9 Hz corresponds to nominal velocity of 2.5 m/s. It shows that the velocity profile is uniform and the presence of gaps at the top of the test section has no effect on the velocity profile. The calibration curves at 14.4 and 21.9 Hz corresponds to 7.5 and 12.5 m/s (nominal) respectively. At these fan speeds, the suction of velocity causes the drop in dynamic pressure at the top to be intensified.

From the calibration curves, it is evident that the flow is generally uniform in the test section. Beyond the calibration, the top-third of the test section was neglected in the final measurement procedure.

A sample of the raw data for the measurements is given below at 21.9 Hz. The rest of the data is provided in Appendix A (Experimental Runs).

In the raw data presented, the gray area represents areas in which the pressure readings were deemed unreliable due to the opening of the test section that caused additional suction at higher fan frequencies. The green highlighted portion indicates the areas where pressure measurements were taken, and the red area indicates the presence of the 3D printed tree and the inability to take measurements due to measurement device limitations.

**Figure 211: Raw data of measurements in the wind tunnel at 21.9 Hz**
Measurements of velocity around tree

The results from the experiment show that the flow eventually regains its velocity magnitude leeward of the 3D printed tree towards the top.

The distance of the wake can be compared with the work in [72] also show a gradual reattachment of the lower layers of the velocity compared to the top layers. The difference between the velocity magnitude for the experiment and the numerical simulation results is between 20 and 40 percent for the different tree height measurements.

Based on the sensitivity limitations in an inclined manometer and the Pitot tube inlet diameter, the velocity profile is generally linear with variations 3 inches to the open traverse slot. The presence of the open traverse slot gives rise to a pressure drop of 0.005 in. H₂O.

The diameter used to normalize the experimental results was the measured diameter at the height of the probe. The plots in figure 212 show the calculated streamwise velocity as a function of the normalized diameter and dynamic pressure, the diameter used to normalize the result is the average of the height at which the measurements were taken; 1 in. (0.0254 m). The incoming velocity at each fan speed and height calculated in the wind tunnel was used to normalize the velocity.
Figure 212: The velocity profile at each of the three heights measured from the bottom of the test section at 4 in., 4.5 in. and 5 in. (a) The measured velocity profile at 2.5 m/s (b) The measured velocity profile at 7.5 m/s (c) The measured velocity profile at 10 m/s (d) The measured velocity profile at 12.5 m/s
There is a 23 percent difference between the experimental results and the simulation in this study, numerical simulation (1500 m$^{-1}$) in [67] and experimental work done in [72]. This is indicative of the errors identified earlier in the experimental procedure and the inability to measure turbulent intensity in the procedure.

The velocity rapidly returns to close to freestream velocity, 0.78$u_0$ compared to literature. This observed behavior is as a result of the 2-D flow assumptions made in this study.
Figure 214: Comparison of streamwise experimental velocity with [72] for solid case.

The experimental results of one of the porous cases from [72] was compared with the experimental results. There is a lower drop in velocity in my experimental runs in this study at probe height of 5 in. (0.127 m) and 4 in. (0.1016 m).