Design and Development of a New Model for Predicting the Mechanical Properties of Three-Dimensional Braided Composite Materials

by

Daniel Ryan Aldrich

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering
University of Alberta

Abstract

Three-dimensional (3D) braided composite materials have been used in the transportation, aviation, and military applications because of their many beneficial attributes. Included in these attributes is their increased through thickness properties that make them an idea replacement for composites where structural integrity is required. This increase to the through thickness properties arises from the additional interlocking yarns in the through thickness direction. However, due the complexities introduced by the additional yarns and undulations in the through thickness direction, modelling, predicting the properties, and tailoring the properties becomes quite difficult.

The goal of this thesis is to present an investigation on three-dimensional braids produced from a rotary three-dimensional braider. Additionally, to develop a geometric and finite element models using sub-modelling. And to create a model for predicting the axial and transverse tensile elastic moduli of the 3D braids.

The development of an in-house rotary three-dimensional braiding machine is presented. The machine is commissioned, and braids are produced to demonstrate the geometric similarities between the geometric models produced with the same braiding parameters.

Two geometric models are developed, a mathematical model as well as a computer aided design model. These models provide a visual representation of the braided structure and allow further development of a FEA analysis to determine the braid properties. A finite element model is developed, as well as a global-local model. The models are used to predict the axial and transverse modulus for the braid, and the axial and transverse stiffness for the sub-unit cells, respectively. Finally, a method of combining the individual sub-unit cell stiffnesses is developed to predict the axial and transverse moduli. This method provides a rapid way of calculating the moduli of the braided structure, as well as a method for predicting the properties of braids with a larger cross-sectional area.

To analyse the data collected during this study, a design of experiments (DoE) is used. Further, the DoE allowed for the determination of the significance of the factors on the axial and transverse moduli. It is found that the cross-section braided structure has a significant effect on the each of these moduli. Further, these properties are attributed to the number of each of the sub-unit cells present in the braided structure. It is then shown that the as the braided structure increases in size, the moduli approached the stiffness value of the middle sub-unit cell due the larger rate at which the number of middle sub-unit cells grow compared to the others.

Finally, the study compares the results of the new method for determining 3D braided properties to the well established FGM model. It is shown that the new method's values for predicting the elastic properties are more accurate when compared to the FGM model.

Preface

This thesis is an original work by Daniel Ryan Aldrich. No part of this thesis has been previously published.

“We knew the world would not be the same. A few people laughed, a few people cried, most people were silent. I remembered the line from the Hindu scripture, the Bhagavad-Gita; Vishnu is trying to persuade the Prince that he should do his duty and, to impress him, takes on his multi-armed form and says, “Now I am become Death, the destroyer of worlds.” I suppose we all thought that, one way or another.”

-Robert Oppenheimer, *“The Decision to Drop the Bomb”*

To my loving parents, Anna & Dan, and sister, Amanda, without their love and support I would not have achieved as much as I have.

Acknowledgements

I would like to start by thanking my supervisor Dr. Cagri Ayranci. Dr. Ayranci, I would like to thank you for the incredible opportunities that you were able to provide to me. Throughout my degree, Dr. Ayranci was able to help and guide me in my studies, and ultimately was able to assist me achieving my goals.

Additionally, I would like to thank my defence examination committee: Dr. Hossein Rouhani, Dr. Jason Carey, and Dr. Ahmed Qureshi for their recommendations and comments, time, and commitment.

Next, I would like to thank my family. My loving parents, Anna and Dan Aldrich, and my sister, Amanda Arya, who were always there for me and pushed me to do my best. Their love and guidance has carried me so far.

Finally, I would also like to thank my best friend, Beste Avcı, for her support, kindness, and general silliness; you always kept me smiling.

Contents

List of Tables

List of Figures

List of Symbols

Latin

A_i	Cross-sectional area, where i specifies the area of body i.
d	Yarn diameter.
E_{ii}	Young's modulus, where ii specifies the direction.
G_{ij}	Shear modulus, where ij specifies the direction.
k_{eff}, k_i	Effective and individual spring constants.
K^*, K_f, K_m	Bulk modulus for the composite, fibres, and matrix.
n_c, n_e, n_m	Number of sub-unit cells for the corners, edges, and middle.
n	Number of cams in the y-direction
m	Number of cams in the z-direction
r	Yarn radius.
SF	Spacing factor.
V_i, V_f, V_m	Volume fraction of the individual constituent, fibre, and matrix.
z	Braiding pitch.

Greek

ε	Strain.
θ_{int}, θ_{ext}, θ_{ave}	Braid angle for the internal, external, and average.
κ	Yarn packing factor.
ν_{ij}	Poisson's ratio, where ij specifies the direction.
ρ	Density.
σ	Stress.

Abbreviations

2D Two-Dimensional

3D Three-Dimensional

CAD Computer-Aided Design

CLPT Classical Laminate Plate Theory

DOE Design of Experiments

FE Finite Element

FEA Finite Element Analysis

FGM Fabric Geometry Model

FRPC Fibre Reinforced Polymer Composite

GPIO General Purpose Input and Output

GUI Graphical User Interface

PMC Polymer Matrix Composite

PWM Pulse Width Modulation

RVE Representative Volume Element

TCM Textile Based Composite Material

UofA University of Alberta

Glossary of Terms

Anisotropic A material not having the same properties in every direction.

Axial Yarns A yarn whose angle is 0° with respect to the longitudinal axis of the braid.

B-Spline A spline formed using minimal support for the degree of the spline.

Bobbin A barrel for holding yarns, this may include flanges or a taper.

Braid A textile material created from three or more yarns that are interlaced with no two yarns twisted around each other.

Braid Angle The acute angle of the yarns measured from the axis of the formed braid.

Braid Cycle Refers to the steps required to complete a cycle of the braiding process.

Bulk Modulus A volumetric modulus based on the pressure developed from the application of a volumetric strain.

Cam See Horn Gear.

Carrier A device used to carry and support the thread and bobbin assembly.

Denier A form of linear density measured as the mass, in grams, per 9000 metres of yarn.

Density The mass per unit volume usually expressed in grams per cubic centimetre.

Exterior Angle The acute angle of the exterior yarns measured from the axis of the formed braid.

Fibre/Filament The smallest continuous unit of matter that defines the basic element of textile materials.

Fibre Volume Fraction Ratio of the area of fibres to the total area of the composite.

Horn Gear A notched disk used to advance the carriers to their new positions.

Impregnate The process of adding the matrix material into the composite.

Interior Angle See Braid Angle.

Linear Density the mass per unit length of yarn, in grams per centimetre.

Longitudinal Referring to the directions parallel to the braid axis.

Pitch Length The length measured along the braid axis to form a unit cell.

Stitching The process of joining layers of fabric by passing a thread through their thickness.

Surface Angle The acute angle of the yarns on the surface measured from the axis of the formed braid.

Sub-Unit Cell Additional, smaller element that can be patterned to recreate a full composite material.

Tex The mass per length of yarn, in grams per 10000 centimetres.

Transverse Referring to the directions perpendicular to the braid axis.

Triaxial Braid A type of two-dimensional braided structure that has additional axial yarns.

Unit Cell The smallest repeating element of a composite material.

Volume Fraction The volumetric proportion of the of one constituent of the composite.

Yarn Collection of fibres/filaments.

Yarn Packing Fraction Ratio of the area of fibres to the area of a yarn.

Chapter 1

Introduction

Due to their high specific stiffness and strength, the use of composite materials is quickly increasing throughout the world [1]. The annual market share, for carbon fibre composites alone, occupies an annual share of over 13.23 billion in U.S. dollars worldwide [2]. This annual share is a result of composite materials being increasingly used in a number of different fields including transportation, aircraft and aerospace, structural applications, and military [3]. In the aerospace industry, Boeing is increasing the use of composite materials in their aircrafts, for example, the 787 Dreamliner is composed of 50% advanced composite materials [4].

Composite materials offer numerous advantages over traditional materials. The main advantages are lower weight, high corrosion resistance, high fatigue strength, increased stiffness, and increased strength, as well as tailorable properties [5]. Because of their tailorability, composite materials can be designed for a specific application. This tailorability is achieved by adjusting the design and manufacturing parameters, such as, fibre angle, fibre volume fraction, materials, *etc.* On the other hand, composites face a number of challenges including high cost due materials used, UV sensitivities, high environmental impacts, brittle failure, and complex fabrication processes.

Composite materials can be viewed under two broad categories, short fibre and long fibre. Short-fibre composites have lower general properties; however, they are less costly to produce. Long-fibre composites are used when higher loads are expected and are therefore more common; some examples of long fibre composites include laminates, filament winding, braiding, *etc.*

Among these long-fibre composites, braided composites have better through-thickness properties when compared to some conventional composites, such as laminates, due to the additional interlocking of the yarns [5]. Braided composites can be further sub-categorized as two-dimensional (2D) and three-dimensional (3D) braided composites. While the properties of the 2D and 3D braided composites can be tailored by adjusting the materials, braid angle, inclusion of axial fibres, *etc.*; two-dimensional braided structures are still limited by the properties in the through-thickness direction. By including additional interlocking yarns in the through-thickness direction, a three-dimensional (3D) braided composite is created and further increases the through-thickness properties. For this thesis, the design and modelling of three-dimensional braided composites are considered.

1.1 Motivation

Three-dimensional braided composite structures are composed of yarns of fibres braided to form a preform with a solid cross-sectional area [6]. The solid cross-sectional area is created by the inclusion of additional interlocking yarns in the through-thickness direction, which results in increased through thickness properties for 3D braided composites [6, 7]. Though 3D braided composites have been developed to help prevent problems of delamination and to increase the through-thickness properties, there still exist a number of complications that limit the widespread use of these composites. The additional interlocking of the yarns and the resulting undulations further complicates the geometry of the braid and leads to complications in the development of geometric and finite element models needed to predict the properties of the structures.

Properties of three-dimensionally braided composites need to be accurately determined to allow the increase in the use of these composites. Because production of

3D braided composites is much slower compared to 2D braided composites, trial-and-error strategies to design these composites will not be sufficient. The need for an accurate method and model to predict the composite's properties is required. In literature, these models often utilize "unit cells" that are composed of yarns that follow straight paths, or that would require massive iteration to tailor the properties [7–18]. Most current models make oversimplifications to the geometry, such as assuming the yarns travel straight paths in the unit cell and do not account for the undulations of the yarns. Additionally, these models are not greatly suited for use in tailoring applications due to the requirement to rerun the model for each iteration of the tailoring process.

Consequently, it is necessary to have a reliable machine to produce braided structures, along with an advanced model and method, to predict the properties. These lead to the objectives of this thesis. The complications in the modelling 3D braids, and the need for a reliable and accurate method to predict the properties of the 3D braids resulted in the motivation behind this study.

1.2 Thesis Objectives

The first objective of this thesis is to develop a computer-controlled machine that is capable of reliably producing 3D braids. The machine is required to produce a 3D braid using a rotary style movement, as well as have the ability for further adaptability in the future. This machine is used to help further develop an understanding of the braid geometry.

The second objective is the creation of a geometric, computer-aided design (CAD), and finite element (FE) model. To create the geometric model, the method relies on the emulation of the braiding machine to determine the position of the yarns. The geometric model is used to create a computer-aided design model, as well as a finite element model. The finite element model can then be analysed with global-local FE analysis to determine the properties of the sub-unit cells.

The final objective of this thesis is to form a new method and model for determining the tensile axial and transverse Young's modulus of 3D braided composite structures. This new method describes an efficient way to determine the tensile axial and transverse Young's modulus of different sized rectangular (square) 3D braids. The proposed method relies on the use of a finite element model that is analysed by its unit and sub-unit cells to create a method for predicting the properties of larger braided structures.

1.3 Thesis Outline

This thesis is organised into seven chapters, and appendices. A background to composite materials, and the progression from 2D composites materials to the development of 3D braided composites is provided in Chapter 2. Current models used for analysis of composites and 3D braided composites are summarized in Chapter 3. The design of an in-house 3D braiding machine is examined in Chapter 4, as well as a general analysis of the geometric structure. The geometry is analysed with open source image analysis software. The mathematics and programming behind the geometric and CAD models used to develop the final FE model are investigated in Chapter 5. The models presented in Chapter 5 are based on the modelling technique, machine emulation. Further, the efficient automation of the creation of the models is explored. The method used to determine the Young's modulus in the axial and transverse directions is outlined in Chapter 6. The application of global-local FEA on sub-unit cells is shown to provide the stiffness of the sub-unit cells, and the application of a spring model is used to determine the tensile properties of the full model as well as the larger models. The results are compared to the well-known fabric geometry model by Dr. Frank Ko, and the analysis of the full cross-sectional area model is given. Finally, he presented and future work is summarized in Chapter 7.

References

[1] D. Gay, S. V. Hoa, and S. W. Tsai, *Composite Materials: Design and Applications*. CRC Press, 2002, ISBN: 9781420031683. DOI: 10.1201/9781420031683.

[2] M. Sauer, M. Kühnel, and E. Witten, "Composites market report (2017)," AVK & CCeV, Tech. Rep., 2017.

[3] D. D. L. Chung, "Applications of composite materials," in *Composite Materials: Functional Materials for Modern Technologies*. London: Springer London, 2003, pp. 1–13, ISBN: 978-1-4471-3732-0. DOI: 10.1007/978-1-4471-3732-0_1. [Online]. Available: https://doi.org/10.1007/978-1-4471-3732-0_1.

[4] J. Hale, "Boeing 787 from the ground up," *Areo Quarterly*, no. 4, 2006.

[5] A. K. Kaw, *Mechanics of Composite Materials*. Taylor & Francis Inc, Oct. 1, 2005, 490 pp., ISBN: 0-8493-1343-0.

[6] R. T. Brown, "Through-the-thickness braided composites for aircraft applications," in *FAA, Ninth DOD(NASA)FAA Conference on Fibrous Composites in Structural Design*, vol. 3, 1992, pp. 1231–1247.

[7] F. Ko, "Tensile strength and modulus of a three-dimensional braid composite," in *Composite Materials (Seventh Conference)*, 1984, pp. 392 –403.

[8] F. Ko and C. Pastore, "Structure and properties of an integrated 3-d fabric for structural composites," in *Recent Advances in Composites in the United States and Japan*, ASTM International, 1985, pp. 428–439. DOI: 10.1520/stp32805s.

[9] J.-M. Yang, C.-L. Ma, and T.-W. Chou, "Fiber inclination model of three-dimensional textile structural composites," *Journal of Composite Materials*, vol. 20, no. 5, pp. 472–484, 1986. DOI: 10.1177/002199838602000505.

[10] G.-W. Du and F. Ko, "Unit cell geometry of 3-d braided structures," *Journal of Reinforced Plastics and Composites*, vol. 12, no. 7, pp. 752 –768, 1993. DOI: 10.1177/073168449301200702.

[11] L. Chen, X. Tao, and C. Choy, "Mechanical analysis of 3-d braided composites by the finite multiphase element method," *Composites Science and Technology*, vol. 59, no. 16, pp. 2383–2391, 1999. DOI: 10.1016/s0266-3538(99)00087-1.

[12] T. Zeng, L. zhi Wu, and L. cheng Guo, "Mechanical analysis of 3d braided composites: A finite element model," *Composite Structures*, vol. 64, no. 3-4, pp. 399–404, 2004. DOI: 10.1016/j.compstruct.2003.09.041.

[13] N. Tolosana, S. Lomov, and A. Miravete, "Development of a geometrical model for a 3d braiding unit call based on machine emulation," 2007.

[14] N. Tolosana, M. Carrera, R. G. de Villoria, L. Castejon, and A. Miravete, "Numerical analysis of three-dimensional braided composite by means of geometrical modeling based on machine emulation," *Mechanicas of Advanced Materials and Structures*, vol. 19, pp. 207 –215, 2012, ISSN: 1537-6494. DOI: 10.1080/15376494.2011.578784.

[15] C. Zhang, X. Xu, and K. Chen, “Application of three unit-cells models on mechanical analysis of 3d five-directional and full five-directional braided composites,” *Applied Composite Materials*, vol. 20, no. 5, pp. 803–825, 2012. DOI: 10.1007/s10443-012-9309-0.

[16] H. Y. Yang, X. G. Zhou, J. S. Yu, and Z. Luo, “Entity simulation of rectangular preform for 3d braided composite based on ansys,” *Advanced Materials Research*, vol. 800, pp. 336 –340, 2013, ISSN: 1662-8985. DOI: 10.4028/www.scientific.net/amr.800.336.

[17] C. Zhang and X. Xu, “Finite element analysis of 3d braided composites based on three unit-cells models,” *Composite Structures*, vol. 98, pp. 130–142, 2013. DOI: 10.1016/j.compstruct.2012.11.003.

[18] H. Ahn and W.-R. Yu, “Mechanical analysis of 3d braided and woven composites using fiber-based continuum analysis,” *Composite Structures*, vol. 160, pp. 1105–1118, 2017. DOI: 10.1016/j.compstruct.2016.11.003.

Chapter 2

Background

2.1 Composite Materials

Composite materials are materials that are composed of two or more constituents with different distinct physical properties to create an effective material with characteristics different from each of the individual constituents [5]. The constituents that form the composite remain separated and distinguishable (on the macro-scale) from each other, unlike the combining of metals to form an alloy. The formation of composites usually comprises of two components: the reinforcement, and the matrix [5]. The reinforcement is embedded within the matrix, and can take the form of fibres, particles or flakes; while the matrix is usually a continuous, homogeneous, isotropic material [5].

Traditional structural materials cannot always meet modern demands or requirements. Composite materials offer high strength to weight ratios (specific strength) and high stiffness to weight ratios (specific stiffness), as well as tailorability, and ease of manufacture [19]. These applications include use in the aeronautics/aerospace, transportation, structural, and military applications [4, 19]. Additionally, while composite materials are able to provide high specific stiffness and specific strength, they are also able to provide the benefits of improved fatigue and impact resistance, thermal conductivity, and corrosion resistance [5, 19]. These properties facilitate the use of composite structures in aeroplanes, satellites, consumer products, as well as military gear.

Composites are often distinguished by the material of the matrix (metal, ceramic, or polymer matrix composites) and the type of reinforcement (fibre, particle, or flake reinforced composites) present. In the case of fibre-reinforced composites, they can be first classified as either discontinuous or continuous. Discontinuous (short) fibre composites are a more versatile material in terms of manufacturability. Because these fibres are short, they allow composites to be formed though spray lay-up, extrusion, and even injection moulding. Whereas, continuous (long) fibre composites are more limited in the means of production. However, continuous fibre composites are advantageous due to the general increase in the composite's mechanical properties. Because of the higher properties and practical applications of long fibre composites, this thesis will focus on these. Additionally, of the long fibre composites, they can be further classified as being a textile-base composite if the fibres are woven, knitted, or braided. The use of textile-based composites allows for a higher production rate due to the relative ease of producing textiles.

2.2 Textile-Based Composite Materials

Textile-based composite materials (TCM) are composites that consists of a textile reinforcement embedded in a polymer matrix material. Textile based composites utilise fibres that are interlocked, usually through braiding, knitting, or weaving; however, some forms of TCM are non-woven, where the fibres do not interlock but are supported by the matrix or fused to each other, such as in the case of unidirectional composite tapes and fibre mats. The different types of TCM are summarized in Figure 2.1.

Due to the interlocking fibres in some TCM, a significant increase in the structural integrity of the composite structures can be observed in the off axis direction with regards to strength and stiffness.[19]. The increase in structural integrity makes these materials ideal replacements for structural materials where the specific strength

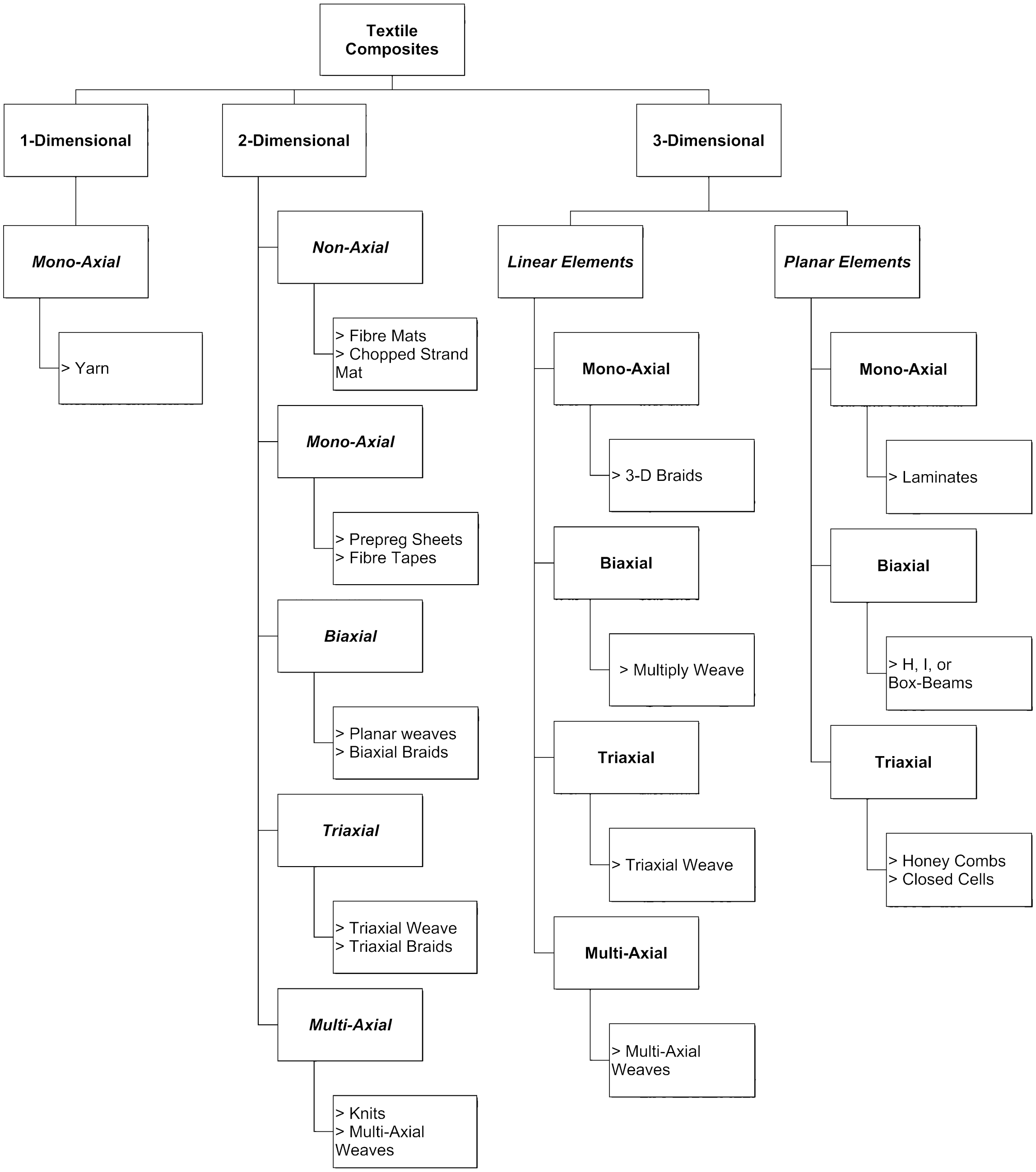


Figure 2.1: Examples of the different types of textile based composites, sorted by their main categories. Adapted from reference [20]

of the material is important. To further increase the structural properties or to tailor the material properties for specific applications, the development of various two-dimensional (2D) composite materials have been introduced. Of the many different types of 2D composites, 2D braided composites (tubular braided composites) and woven composites (lamina based composites) are the most common.

Braided and woven composite structures both consist of interlocking yarns, however, they have been differentiated by some researchers based on the direction that the yarns travel [21]. Florentine *et al.* has referred to structures that consist of thick walled structures that produce a net shape as a woven material [21]. However, Dr. Frank Ko has mentioned that the difference between weaving and braiding to be the direction that the fibres travel [21]. In a braided structure, the yarns travel/undulate along a single axis known as the braiding axis, where as woven composites the yarns travel along a set of orthogonal directions [21]. For this study, a focus on braid composites, under the definition that Dr. Frank Ko introduced, is made.

These braided structures still have drawbacks, where the properties in the through-thickness direction are limited. The reduced properties in the through-thickness direction are usually due to the limited material in this direction. To mediate this problem, the inclusion of additional interlocking yarns in the through-thickness direction has been shown to increase these properties [8]. These new structures, known as 3D braided composites, are explored in this thesis.

2.3 Three-Dimensional Braided Composites

The production of three-dimensional (3D) braided composites is achieved with two main types of 3D braiders, in the literature. The first design (Track and Column) was originally developed by Florentine, with the second design (Rotary) originally developed by Smith (Square Braiding),later adapted by Tsuzuki to increase the number of yarns [22, 23]. These were followed by an adaptation by Ko to accommodate a hexagonal close packed arrangement (Hexagonal), and finally a further adaptation by Ko to increase the number of yarns further with the inclusion of a switching mechanism (Advanced Hexagonal) [24]. Further detail on the design of these braiders is explored in Chapter 3.

Many researchers have investigated the tensile strength and modulus of these three-dimensional braids. It was found that, though there is a lowering of the tensile properties in the axial direction, there is a significant increase in the properties in the through-thickness direction [7, 8, 25, 26]. This increase in the properties in the through-thickness direction led to the development of these structures for use in aircrafts [6, 27]. The increase in the properties allowed for higher structural integrity and made it possible to develop structures already with the required shape [6].

Due to these advancement in the aircraft industry, there was a need to be able to predict the properties of the final braided structure before the timely manufacturing process took place. This led researchers to start analysing the structures in a similar way to 2D braid composite structures.

By braking the structure down in to unit cells, a representation of the full structure was created. These unit cell structures allowed for the analysis of a much smaller representative volume, rather than needing to analyse the full structure [10].

Further effort was taken to try to analyse the smaller repeating structures found within the unit cell. These structures, known as sub-unit cells, have been analysed with multiphase finite element analysis to better predict the properties of these structures. [11].

Methods for 3D braiding have been altered by some researchers in the attempt to increase the number of active yarns and thereby increase the fibre volume fraction for these structures [24].

The "five-directional" braided composites, presented by Zhang *et al.*, include additional axial fibres that are shown to increase the tensile and bending stiffnesses [15]. The axial fibres act similar to the those found in 2D tri-axial braided composite structures.

In an effort to better predict the mechanical properties of 3D braided composites researchers, such as Gao *et al.*, have further developed existing models into computer programs [28]. These computer programs are capable of assisting in the determining the braid parameters required to tailor the properties of 3D braided composites [28].

2.4 Conclusions

In this chapter, a brief background on composite materials, textile-based composite materials, and the development of three-dimensional braided composites is presented. Composite materials are shown to have desirable properties over traditional materials, including their high specific strength and specific stiffness, and their ability to be tailored to the application. Though these composite materials have numerous benefits, they do still pose some complications. Where these complications exist, the development of textile-based composites aims to correct. However, it is noted that even the textile-based composites still have drawbacks mainly the through-thickness properties. This leads to the introduction of 3D braided composite materials. In the next chapter, various machines and models that have been developed for 3D braided composites are outlined.

References

[4] J. Hale, “Boeing 787 from the ground up,” *Areo Quarterly*, no. 4, 2006.

[5] A. K. Kaw, *Mechanics of Composite Materials*. Taylor & Francis Inc, Oct. 1, 2005, 490 pp., ISBN: 0-8493-1343-0.

[6] R. T. Brown, “Through-the-thickness braided composites for aircraft applications,” in *FAA, Ninth DOD(NASA)FAA Conference on Fibrous Composites in Structural Design*, vol. 3, 1992, pp. 1231–1247.

[7] F. Ko, “Tensile strength and modulus of a three-dimensional braid composite,” in *Composite Materials (Seventh Conference)*, 1984, pp. 392 –403.

[8] F. Ko and C. Pastore, “Structure and properties of an integrated 3-d fabric for structural composites,” in *Recent Advances in Composites in the United States and Japan*, ASTM International, 1985, pp. 428–439. DOI: 10.1520/stp32805s.

[10] G.-W. Du and F. Ko, “Unit cell geometry of 3-d braided structures,” *Journal of Reinforced Plastics and Composites*, vol. 12, no. 7, pp. 752 –768, 1993. DOI: 10.1177/073168449301200702.

[11] L. Chen, X. Tao, and C. Choy, “Mechanical analysis of 3-d braided composites by the finite multiphase element method,” *Composites Science and Technology*, vol. 59, no. 16, pp. 2383–2391, 1999. DOI: 10.1016/s0266-3538(99)00087-1.

[15] C. Zhang, X. Xu, and K. Chen, “Application of three unit-cells models on mechanical analysis of 3d five-directional and full five-directional braided composites,” *Applied Composite Materials*, vol. 20, no. 5, pp. 803–825, 2012. DOI: 10.1007/s10443-012-9309-0.

[19] D. D. L. Chung, *Composite Materials: Science and Applications*, Second Edition. New York: Springer, 2010, ISBN: 978-1-84882-830-8.

[20] F. K. Ko, “Three-dimensional fabrics for composites,” *Textile Structural Composites*, pp. 129–171, 1989.

[21] R. A. Florentine, “Magnaweave process - from and fundamentals to applications,” *Textile Research Journal*, pp. 620 –623, 1981.

[22] M. F. Smith, “Apparatus and method for automated braiding of square rope and rope product produced thereby,” U.S. Patent US4803909A, Feb. 1989.

[23] M Tsuzuki, M Kimbara, K Fukuta, and A Machii, “Three-dimensional fabric woven by interlacing threads with rotor driven carriers,” U.S. Patent US5067525, Nov. 1991.

[24] F. Schreiber, F. Ko, H. Yang, E. Amalric, and T. Gries, “Novel three-dimensional braiding approach and its products,” in *ICCM-17: 17th International Conference on Composite Materials*, 2009.

[25] D. Whyte, “Structure and properties of 3-d braid reinforced composites,” PhD thesis, Drexel University, United States, Jan. 1986.

[26] C.-L. Ma, J.-M. Yang, and T.-W. Chou, "Elastic stiffness of three-dimensional braided textile structural composites," in *Composite Materials: Testing and Design (Seventh Conference)*, ASTM International, 1986, pp. 404–404–18. DOI: 10.1520/STP35360S.

[27] M. B. Dow and H. B. Dexter, "Development of stitched, braided and woven composite structures in the act program and at langley research center," 1997.

[28] Y. T. Gao, F. K. Ko, and H. Hu, "Integrated design for manufacturing of braided preforms for advanced composites part ii: 3d braiding.," *Applied Composite Materials*, vol. 20, no. 6, pp. 1065 –1075, 2013, ISSN: 0929189X. DOI: 10.1007/s10443-012-9304-5.

Chapter 3

Literature Review

3.1 Introduction

In this chapter, a discussion on the development of three-dimensional braiding, and the design of three-dimensional braiding machines is provided. Additionally, the development of current well-established models for predicting the properties of these three-dimensional braids is established. The progression from unit cell representational volume elements (RVE) to sub-unit cell RVE is provided. Finally, the need for a new model for predicting the properties of three- dimensional braided structures is presented.

3.2 3D Braider Designs

There are two main types of three-dimensional (3D) braiders that have been developed and researched: Track and Column, and Rotary. Track and column braiders utilize a grid of carriers that are moved around on a baseplate usually assisted by actuators. These actuators move the entire column or row of carriers. Rotary braiders utilize a series of horn-gears or cams to move the carriers around a baseplate. The cams can be linked together and driven by a motor or each individual cam can be driven by an independent motor for more advanced braided structures. Track and Column, and Rotary designs both have advantages and drawbacks.

3.2.1 Track and Column

There are two main set-ups for track and column braiders, rectangular and circular [29]. For the rectangular set-up, the carriers are arranged in orthogonal rows and columns [21, 30–32]; whereas the carriers are arranged in concentric rings and azimuthal lines for a circular set-up [10, 29, 33, 34]. Within track and column 3D braiding, there exists many different desired patterns that machines are created to perform. For track and column braiders, the more common patterns are the two-step [35] and four-step [36] processes. Florentine has created and patented a four-step track and column 3D braiding machine as well as a process for producing 3D braids using both rectangular and annular track and column designs [21, 30].

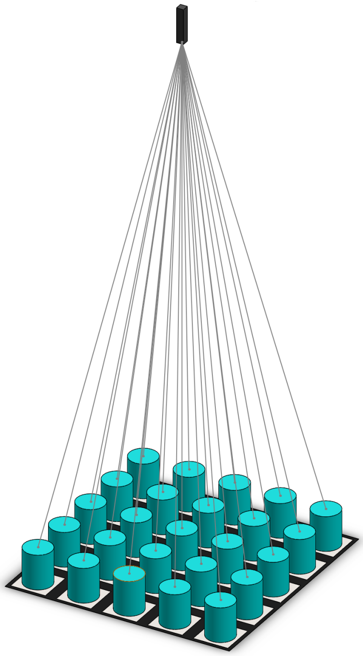


Figure 3.1: Example track and column 3D Braider.

3.2.2 Rotary

There are many different types of rotary braiders that have been developed. The first type of rotary design, known as the square braiding machine, was developed by Michael Smith [22]. The square braiding machine is similar to later machines produced by Tsuzuki *et al.*, however, it uses a maypole style movement that is truly continuous [22]. The second type of rotary design is the four position rotary 3D braider. Tsuzuki *et al.* adapted the square braiding machine to fill all of the locations on the cams and created a Cartesian grid of yarns [23]. This greatly increased the number of yarns used to produce a braid, but required a discontinuous (stepwise) movement [23]. To further increase the number of yarns in a braided structure, Dr. Frank Ko development the hexagonal rotary 3D braider based on the hexagonal close packing [28, 37]. Additional research has allowed a further increase in the number of yarns in a 3D braided structure [24]. Schreiber *et al.* included switching mechanisms between the adjacent cams of the hexagonal braider allowing for more complex braided structures, as well as the higher fibre volume fractions [24].

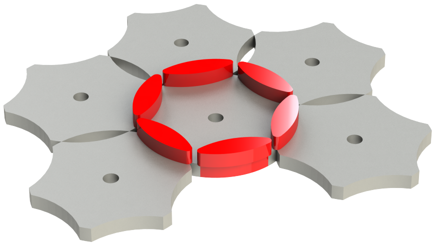


Figure 3.2: Example Rotary 3D braider.

3.3 3D Braided Unit Cell Geometry

3.3.1 Unit Cells

Unit cells, example shown in **??**, are the smallest structure within the full composite structure that can be repeated to recreate the full structure. Unit cells are beneficial because they allow the simplification of the full structure of the composite, while retaining the properties from the full structure. Because of this great simplification, many researchers have developed models and have investigated the properties of these unit cells [38–40]. The use of unit cells have proven invaluable as they are able to simplify complex models into a single repeating element. Many models developed for 3D braids, often over simplify the geometry of the unit cell by assuming that yarns travel in straight lines, and that there is limited interaction between the constituents. Because of these assumptions and oversimplifications, it leads to the development of models that are inaccurate [7, 8, 10, 11, 41].

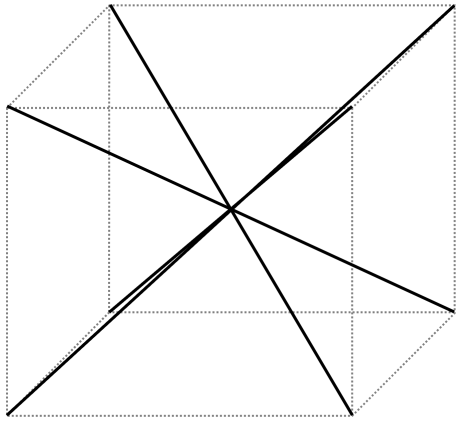


Figure 3.3: Example of a basic unit cell

3.3.2 Sub-Unit Cells

Sub-unit cells are an extension of unit cells, however, instead of a single structure being used to represent the full composite, the sub-unit cells can be patterned and rotated to represent the unit cell. Sub-unit cells for 3D braids were first noted by Dr. Ko, and analysed by many researchers [11, 15, 17]. The sub-unit cells have not been extensively studied due to their inherent complexity, however it has been noted that sub-unit cell posses differing properties from each other. This thesis aims to provide a method and model to analyse the sub-unit cells and predict the properties of the full-unit cell. Further, an investigation into the ability to extrapolate the properties to predict the properties of larger braided structures is discussed.

3.3.3 Geometric Models

Geometric models are used to help visualize and increase understanding of the shape of composite materials. Geometric models use mathematical and computational geometry to describe the geometric shape, including the creation of models through computer aided design (CAD). Many geometric models have been developed to help understand the geometry and properties of 2D composite structures [41–45]. The geometric models for 2D composite structures, often focus on the construction of the representative unit cell geometries for the structure rather than the structure as a whole. And due to the complex nature of the interactions found within these structures often simplifications are made. More notably, models often reduce the paths of yarns in the unit cell to straight lines and disregard the interactions of yarns allowing them to co-occupy the same space. However, more recently 2D models for braided composites have started to account for these limitations [42, 45].

Current geometric models for 3D braided composites are often developed from models that were developed for 2D braided composites [38]. Many researchers have developed these geometric models further for use with FE modelling and these are discussed further in Section 3.4.4. Sheng *et al.* were able to create a geometric model based on the projection of the yarn paths and the crimp function used to describe the yarns [46]. The model was able to show good agreement among results shown in other literature [46].

Machine Emulation

Machine emulation is a process for determining the structure of composite materials. By emulating the machine used to create the structure, the path of the yarns can be predicted and used to create a geometric model [13]. Though this process has been used in three-dimensional braided composites, no research has shown its implementation in two-dimensional composites.

For machine emulation, often code is created to track the locations of the individual yarns, as well as emulate the braiding process. The relative locations of yarns during the braiding processes are usually stored within a matrix [47]. These relative locations can be converted into the absolute locations for generating the geometric model. This creates a discrete path for the individual yarns of the braided composite [13, 47]. Further, it has been shown that the geometric models obtained through machine emulation can be exported into FEA software to determine the elastic properties of the composite's unit cell [14].

3.4 3D Braided Composite Models

Many researchers have extensively studied three-dimensional braided composites. In this section, a brief chronological review of these studies is provided, followed by a breakdown of the research based on various modelling methods that are used. There have been a number of different models, where some are derived from 2D models and, as such, some of the models presented include references to the 2D models.

Three dimensional braiders have been developed by several different researchers as a means to provide braided composites with better through thickness properties. Additionally, they allow for the formation of braided composites with a specified or varying cross-sectional area.

Ko *et al.* developed a geometric model based on the rule of mixtures and cosine averaging to predict the elastic properties of 3D braids based on the hexagonal close packing rotary 3D braider. This model assumes that the yarns of the braided structure follow idealized straight paths, and do not interact with adjacent yarns. The results from this method were compared to experimental results and it was found that the results were within 20.3% of the experimental results. Ko *et al.* also noted that the 3D braided composite materials had a lower longitudinal strength and modulus when compared to unidirectional composites, however, possessed a higher longitudinal strength and modulus when compared to woven laminates[7].

Ko *et al.* continued their research on 3D braided composite materials, showing that they can be used to develop 3D braided structures with a specified cross-sectional area. By further developing the previous model, it was adapted and applied to these new 3D braided structures. Using the properties of the yarns, the properties of the preform, as well as the composite are able to be predicted. It was found by Ko *et al.* that the model was able to predict the tensile elastic properties within 5 to 25% error. [8]

The model developed by Ko *et al.* was further developed by Whyte *et al.* to create a method of predicting the properties of a composite based on its unit cell geometry. This method now known as the fabric geometry model (FGM) is largely based on the volume averaging methods used for two dimensional composites. The FGM method, like the previous two methods, assumes that the yarns are ideal and travel straight paths through the unit cell; additionally, the yarns are assumed not to interact with each other [25].

Yang *et al.* developed a predictive model to determine the elastic properties of 3D braided composite structures. This model assumes that the yarns that compose the braid are treated as straight laminae sheets inclined to the braid angle of the yarns. The determination of the properties from this model is based on CLPT, and further assumes that the yarns all pass thought the centre of the unit cell without interacting with each other. It was noted by Yang *et al.* that the predictions are in reasonable agreement with experiments performed by other researchers, however, are unable to provide more than an approximation for the fibre volume fraction due the model allowing the fibres to co-occupy the same space [9].

Mo *et al.* provided a strain energy-based method for determining the elastic properties of the braided composites. Similar to the previous model Mo *et al.* utilize a unit cell that consists of idealized straight yarns [26].

Du *et al.* developed a geometric model that like previous models assumes that the yarn travel along straight paths within the unit cell. However, this model does not allow the yarns to pass though each other and occupy the same space. By not allowing the yarns to pass though each other, a more accurate fibre volume fraction can be determined. Furthermore, the maximum allowable fibre volume fraction was determined as the point when the yarns begin to jam. Du *et al.* were able to compare their predicted results with experimental data; this showed that the experimental data was closer to their theoretical limit, due to the tensions required to form the braiding structures [10].

Wang *et al.* provided a method for mapping the geometric structure of one braid to another allowing the original braid to undergo a shape change [41].

Chen *et al.* provided a finite element (FE) method that divides the typical unit cell model into several smaller sub-unit cell. They found that the braided structure can be divided into three types of sub-unit cells: the interior, surface, and corner cells. By reducing the unit cell into three smaller repeating sub-unit cells, the number of elements required in the FE analysis are drastically reduced, as well as the required amount of memory to run the simulation is reduced. Chen *et al.* found that the results fell within 2.2 to 12.2% of the experimental results [11].

Kostar *et al.* showed how different 3D structures can be created utilizing 3D braiding, including: composites that have a varying cross-sectional area [32].

Zeng *et al.* expanded on Chen's method to determine the structural properties of 3D braided composites [11], to include a method of determining the properties of damaged composites [12].

Zhang *et al.* provided a method using Visual-C++ to automatically create CAD models, given specified braiding parameters [48].

Tolosana *et al.* developed a geometric model, for the four-step (track and column) braiding, that is based on the movements of the machine to predict the final braided structure. By tracking the locations of the yarns at the machine, during its operation, the position of the yarns in the braid can be predicted. Tolosana *et al.* found that the developed geometric model follows closely to the geometry of the actual braided specimen [13].

Schreiber *et al.* developed and constructed a new type of 3D braider based on the original design built by Dr. Ko. The new design increases the number of yarns by including a switching mechanism between each of the adjacent horn gears. This increase in the number of yarns has shown to increase the fibre volume fraction of 3D braided structures, and the introduction of a switching mechanism allows for more complex braided structure to be developed [24].

Tolosana *et al.* developed a new method for processing the previously created CAD models [13]. The new method imports the CAD model of the yarns into FE software where the yarns are compressed together from the sides. After compression brings all the fibres closer together they are embedded in a matrix before virtual testing is performed on the samples. Tolosana were able to predict the longitudinal compression modulus and found that the results are within 4.2% of the experimental results [14].

Zhang *et al.* like Chen [11] and Zeng [12] divided the braided structure into sub-unit cells and were able to test them to determine that the sub-unit cells possess differing properties from one another [15].

Deng *et al.* were able to use the MATLAB®programming language to implement machine emulation presented by Tolosana *et al.* [13]. They found that they were able to create a geometric representation of the braided structure [47].

Gao *et al.* created a software solution with a GUI to assist in the process of tailoring the properties of 3D braided structures. The software implements the fabric geometry model, presented by Dr. Ko [8, 25], to predict the elastic properties of the braided structure [28].

Yang *et al.* used the process of machine emulation [13] to automatically create an APDL file for importing the model into commercially available FEA software for analysis [16].

Zhang *et al.* extended on their previous work [15] and were able to further confirm their findings that the sub-unit cells have differing properties. Furthermore, they were able to use their FEA results to predict the properties of the resulting braided structure and compare them to experimental results where they found that they were within 3.46 to 11.17%. Finally, Zhang *et al.* were able to see the effects that braid angle and fibre volume fraction have on the elastic properties[17].

Paul *et al.* utilized FEA to determine the effects of braid angle and fibre volume fraction on the elastic properties of the braided structure. The found that the braiding angle has a large effect on the longitudinal modulus, however, it only has a small effect on the transverse modulus [49]

Smith *et al.* found that 3D braided composites can be used to create a dental archwire that has tailorable properties. Additionally, they found that the polymer based archwire exhibits properties in a similar range to those found in current dental archwires made from shape memory alloys [50].

Because of the discovered properties and benefits of 3D braided composites, the development of models to predict the properties are necessary to further progress the use of such composite materials. Many models have been developed to help predict the properties of these braided structures. Many of these models are advantageous, however, they also have many disadvantages making them less suitable for predicting the properties of these structures and for assisting in the tailorability of the braids. In the following sub-sections, an overview of the different types of models, their advantages and disadvantages are provided.

3.4.1 Rule of Mixtures & Classical Laminate Plate Theory

The rule of mixtures (RoM) and classical laminate plate theory (CLPT) have been used extensively in composite materials to help predict the properties of braided and laminated composites [51, 52]. They have been shown to provide methods to help predict the mechanical properties of many composite materials.

The RoM for determining the longitudinal elastic modulus and major Poisson's ratio uses a weighted mean of the mechanical properties for each of the constituents that make up the composite material. Alternatively, the RoM uses the inverse weighted sum of inverses to determine the transverse elastic modulus and the shear modulus. The basic assumptions for the rule of mixtures are that,

1. Fibers are distributed evenly throughout the matrix,
2. Loads are applied parallel or normal to the fibre direction,
3. Fibres and matrix are perfectly bonded,
4. Matrix is free of voids,
5. Lamina is initially in an unstressed state, and
6. Fibre and matrix act linear elastically.

The method for determining the properties of a composite with the RoM is summarized in Equations (3.1) - (3.4).

$$E_1 = \sum_i V_i E_{1,i} \tag{3.1}$$

$$E_2 = \left(\sum_i \frac{V_i}{E_{2,i}} \right)^{-1} \tag{3.2}$$

$$\nu_{12} = \sum_i V_i \nu_i \tag{3.3}$$

$$G_{12} = \left(\sum_i \frac{V_i}{G_i} \right)^{-1} \tag{3.4}$$

Classical laminate plate theory is a method for predicting the relationship between the mechanical properties of laminated composite materials. Classical laminate plate theory is an extension of the Kirchkoff-Love theory of plates has been readily used to determine the mechanical properties of laminated composite materials, as well as being adapted for use on other composite structures [53]. The basic assumptions of CLPT are,

1. Every lamina is homogeneous and orthotropic,
2. A line straight and perpendicular to the middle surface remains straight and perpendicular to the middle surface during deformation,
3. The laminate is thin and is loaded only plane stress,
4. Displacements are continuous and small throughout the laminate, and
5. No slip occurs between the lamina interfaces.

The CLPT theory is a method for creating the extensional stiffness, A, the bending stiffness, D, and the extensional-bending coupling, B, matrices that are combined and allows for the determination of the effective material properties of the composite. These equations for CLPT have been summarized in Equations (3.5) - (3.7) and are described in [5]. Many researchers have utilized the CLPT as a base to build their own model for determining the mechanical properties of various composites. Though these methods are widely used, more accurate methods for determining the mechanical properties are continuously being developed from these models, as well as from various other methods.

$$[Q] = \begin{bmatrix} \dfrac{E_{11}}{1-\nu_{12}\nu_{21}} & \dfrac{E_{11}\nu_{21}}{1-\nu_{12}\nu_{21}} & 0 \\ \dfrac{E_{22}\nu_{21}}{1-\nu_{12}\nu_{21}} & \dfrac{E_{11}}{1-\nu_{12}\nu_{21}} & 0 \\ 0 & 0 & G_{12} \end{bmatrix} \tag{3.5}$$

$$\begin{aligned} A_{ij} &= \sum_{k=1}^{N} \left(\bar{Q}_{ij}\right)_k (z_k - z_{k-1}) \\ B_{ij} &= \frac{1}{2}\sum_{k=1}^{N} \left(\bar{Q}_{ij}\right)_k \left(z_k^2 - z_{k-1}^2\right) \\ D_{ij} &= \frac{1}{3}\sum_{k=1}^{N} \left(\bar{Q}_{ij}\right)_k \left(z_k^3 - z_{k-1}^3\right) \end{aligned} \tag{3.6}$$

$$\begin{aligned}\begin{bmatrix}N_x\\N_y\\N_z\end{bmatrix} &= \begin{bmatrix}A_{11} & A_{12} & A_{16}\\A_{12} & A_{22} & A_{26}\\A_{16} & A_{26} & A_{66}\end{bmatrix}\begin{bmatrix}\epsilon_x^0\\\epsilon_y^0\\\gamma_{xy}^0\end{bmatrix} + \begin{bmatrix}B_{11} & B_{12} & B_{16}\\B_{12} & B_{22} & B_{26}\\B_{16} & B_{26} & B_{66}\end{bmatrix}\begin{bmatrix}\kappa_x\\\kappa_y\\\kappa_{xy}\end{bmatrix}\\ \begin{bmatrix}M_x\\M_y\\M_z\end{bmatrix} &= \begin{bmatrix}B_{11} & B_{12} & B_{16}\\B_{12} & B_{22} & B_{26}\\B_{16} & B_{26} & B_{66}\end{bmatrix}\begin{bmatrix}\epsilon_x^0\\\epsilon_y^0\\\gamma_{xy}^0\end{bmatrix} + \begin{bmatrix}D_{11} & D_{12} & D_{16}\\D_{12} & D_{22} & D_{26}\\D_{16} & D_{26} & D_{66}\end{bmatrix}\begin{bmatrix}\kappa_x\\\kappa_y\\\kappa_{xy}\end{bmatrix}\end{aligned} \tag{3.7}$$

There are also models that utilize a modified form of the CLPT; the most notable model started with Ishikawa and Chou. Ishikawa and Chou developed models based on the CLPT to help predict the properties of woven and later satin woven composites. Ishikawa and Chou created three models for predicting the properties of woven composites: the mosaic model, the fibre undulation model, and the bridge model. Of these models the more notable are the mosaic model and the fibre undulation model. The mosaic model reduces a woven composite to an assembly of cross-ply laminates to which CLPT is applied [54–56]. This reduces the complex undulations of the weave to a simple stack of laminae. The fibre undulation model is an extension of the mosaic model that further considers the effects of the undulations and continuity of the fibres [54, 57]. This model adapted the CLPT to take in account the undulating regions of a woven composite, by assuming they follow a sinusoidal path [58].

3.4.2 Volume Averaging Methods

Volume averaging methods (VAM) involve a superpositioning of the individual constituents of the composite based on the volume fraction of the constituent. There are two main methods to superpose the properties together: stiffness averaging and compliance averaging. Stiffness averaging (see Equation (3.8)) utilizes the stiffness matrix where as compliance averaging (see Equation (3.9)) utilizes the compliance matrix.

$$[C_{eff}] = \sum_i k_i\,[C_i] \tag{3.8}$$

$$[S_{eff}] = \sum_i k_i\,[S_i] \tag{3.9}$$

Fabric Geometry Model

The fabric geometry model (FGM) is a model used to predict the elastic properties of composite materials. The model utilizes analysis of the unit cells of the composite to develop a combined stiffness matrix for the composite [8, 20, 25, 59]. Though the model pre-dates the volume averaging methods, the fabric geometry model can be seen as a simplified form of the volume averaging method; with the fabric geometry model not taking in account the undulations of the fibres. FGM has been used with both stiffness averaging and compliance averaging methods for iso-strain and iso-stress conditioning, respectively. FGM provides a simplified and fast method for determining the properties of the braided composites and can even be further applied to other composite structures. However, it has been shown be able to predict the properties to within 5 to 25%, this may be attributed to the assumption that the yarns travel in straight lines and the undulation of yarns are not considered [8, 20, 25, 59]. Further, the FGM method fails to account for the interactions between the yarns and can require a large number of iterations for tailoring properties.

3.4.3 Fiber Inclination Model

The fiber inclination model (FIM) is a combination of CLPT and the VAM. It is a direct extension the fibre undulation model presented by Ishikawa and Chou [60]; whereby the method was adapted for three-dimensional braided structures. It considers each of the different fibre directions as a unidirectional angle lamina set at an incline. It utilizes the equations presented for the CLPT but invokes a VAM to determine the stiffness matrix used to determine the extensional stiffness (A), the bending-extension coupling matrix (B), and the bending stiffness (D) matrices [9, 60]. The fibre inclination model, like the FGM model, does not consider the undulation or the interaction of the yarns. Additionally., for tailoring of properties an iterative process is required.

3.4.4 Finite Element Models

Finite element (FE) models are often implemented to provide accurate representations of composite performance [61, 62]. These models are often based on geometric models; by creating a geometric model, the geometry can be accurately recreated. From there, the model can be simplified to aid in the creation of the finite elements and the solving of the system.

It is shown by Yang *et al.* that geometric models can be efficiently created for export into commercial FE software [16]. For the four-step (track and column) braiding process, Yang *et al.* were able to create appropriate entities in the ANSYS FE software, utilizing code written in Matlab® [16]. The FE model was created to show the structure of the yarn geometry and allowed for the studying of the yarn properties [16]. However, no analysis was conducted.

Axial Young's modulus is shown to decrease as the braid angle increased, this is consistent with Chen *et al.* [11, 12, 49]. It has also been shown that the properties of braided composites can be controlled by adjusting the fibre volume fraction [49]. These changes in the composite's structure, show the importance of tailorability in three-dimensional braided composites.

Other researchers have investigated the smaller sub-unit cells originally noted by Dr. Frank Ko. The sub-unit cells are a collection of repeating geometries in the braided composite, that unlike a regular unit cell do not represent the full braided structure. Instead, sub-unit cells are combined by patterning, translating, and rotating to form the completed braided structure. Modelling of the sub-unit cells was performed by Zhang *et al.* and it was shown that the individual sub-unit cells have differing properties [15, 17]. Zhang *et al.* were able to use this modelling to predict the longitudinal modulus to within 3.5 to 11.2% the experimental results.

Similar to sub-modelling the finite multiphase element method (FMEM), uses a coarse global mesh to determine the overall effects then uses a finer local mesh to determine the effects in the unit cells [11]. However, the finite multiphase element model reduces the FE model into a regular gridded structure that consists of three types of finite elements: fibre, matrix, and mixed [11, 12]. Using these FMEM methods, Chen *et al.* were able to produce results within 2.2 to 12.2% of experimental results.

Many of the model presented here are able to predict the properties of 3D braids, or show properties present in the geometric structure of the braids. However, most of the widely used models over simplify the braided structure by assuming the yarns travel straight paths and the yarns do not interact with each other. These over simplifications lead to inaccuracies in the prediction of the elastic properties. Further, for the advancement of 3D braided composite structures, there must exist an efficient method for tailoring the properties of the braided structure.

3.5 Conclusions

In this chapter, an overview of the current models and methods used in predicting the properties of 3D braided composite structures is provided. Three-dimensional braided composites have been model using geometric, analytical, as well as finite element models. The use of unit cells is introduced as the base for most models and the topic of sub-unit cells is addressed. The progression of 3D braided models is developed, and the current drawbacks are noted. Finally, the need for a new model to accurately predict the properties of 3D braids is made.

References

[5] A. K. Kaw, *Mechanics of Composite Materials*. Taylor & Francis Inc, Oct. 1, 2005, 490 pp., ISBN: 0-8493-1343-0.

[7] F. Ko, “Tensile strength and modulus of a three-dimensional braid composite,” in *Composite Materials (Seventh Conference)*, 1984, pp. 392 –403.

[8] F. Ko and C. Pastore, “Structure and properties of an integrated 3-d fabric for structural composites,” in *Recent Advances in Composites in the United States and Japan*, ASTM International, 1985, pp. 428–439. DOI: 10.1520/stp32805s.

[9] J.-M. Yang, C.-L. Ma, and T.-W. Chou, “Fiber inclination model of three-dimensional textile structural composites,” *Journal of Composite Materials*, vol. 20, no. 5, pp. 472–484, 1986. DOI: 10.1177/002199838602000505.

[10] G.-W. Du and F. Ko, “Unit cell geometry of 3-d braided structures,” *Journal of Reinforced Plastics and Composites*, vol. 12, no. 7, pp. 752 –768, 1993. DOI: 10.1177/073168449301200702.

[11] L. Chen, X. Tao, and C. Choy, “Mechanical analysis of 3-d braided composites by the finite multiphase element method,” *Composites Science and Technology*, vol. 59, no. 16, pp. 2383–2391, 1999. DOI: 10.1016/s0266-3538(99)00087-1.

[12] T. Zeng, L. zhi Wu, and L. cheng Guo, “Mechanical analysis of 3d braided composites: A finite element model,” *Composite Structures*, vol. 64, no. 3-4, pp. 399–404, 2004. DOI: 10.1016/j.compstruct.2003.09.041.

[13] N. Tolosana, S. Lomov, and A. Miravete, “Development of a geometrical model for a 3d braiding unit call based on machine emulation,” 2007.

[14] N. Tolosana, M. Carrera, R. G. de Villoria, L. Castejon, and A. Miravete, “Numerical analysis of three-dimensional braided composite by means of geometrical modeling based on machine emulation,” *Mechanicas of Advanced Materials and Structures*, vol. 19, pp. 207 –215, 2012, ISSN: 1537-6494. DOI: 10.1080/15376494.2011.578784.

[15] C. Zhang, X. Xu, and K. Chen, “Application of three unit-cells models on mechanical analysis of 3d five-directional and full five-directional braided composites,” *Applied Composite Materials*, vol. 20, no. 5, pp. 803–825, 2012. DOI: 10.1007/s10443-012-9309-0.

[16] H. Y. Yang, X. G. Zhou, J. S. Yu, and Z. Luo, “Entity simulation of rectangular preform for 3d braided composite based on ansys,” *Advanced Materials Research*, vol. 800, pp. 336 –340, 2013, ISSN: 1662-8985. DOI: 10.4028/www.scientific.net/amr.800.336.

[17] C. Zhang and X. Xu, “Finite element analysis of 3d braided composites based on three unit-cells models,” *Composite Structures*, vol. 98, pp. 130–142, 2013. DOI: 10.1016/j.compstruct.2012.11.003.

[20] F. K. Ko, “Three-dimensional fabrics for composites,” *Textile Structural Composites*, pp. 129–171, 1989.

[21] R. A. Florentine, “Magnaweave process - from and fundamentals to applications,” *Textile Research Journal*, pp. 620 –623, 1981.

[22] M. F. Smith, “Apparatus and method for automated braiding of square rope and rope product produced thereby,” U.S. Patent US4803909A, Feb. 1989.

[23] M Tsuzuki, M Kimbara, K Fukuta, and A Machii, “Three-dimensional fabric woven by interlacing threads with rotor driven carriers,” U.S. Patent US5067525, Nov. 1991.

[24] F. Schreiber, F. Ko, H. Yang, E. Amalric, and T. Gries, “Novel three-dimensional braiding approach and its products,” in *ICCM-17: 17th International Conference on Composite Materials*, 2009.

[25] D. Whyte, “Structure and properties of 3-d braid reinforced composites,” PhD thesis, Drexel University, United States, Jan. 1986.

[26] C.-L. Ma, J.-M. Yang, and T.-W. Chou, “Elastic stiffness of three-dimensional braided textile structural composites,” in *Composite Materials: Testing and Design (Seventh Conference)*, ASTM International, 1986, pp. 404–404–18. DOI: 10.1520/STP35360S.

[28] Y. T. Gao, F. K. Ko, and H. Hu, “Integrated design for manufacturing of braided preforms for advanced composites part ii: 3d braiding.,” *Applied Composite Materials*, vol. 20, no. 6, pp. 1065 –1075, 2013, ISSN: 0929189X. DOI: 10.1007/s10443-012-9304-5.

[29] A. Bogdanovich and D. Mungalov, “Recent advancements in manufacturing 3-d braided preforms and composites,” *Composite Systems - Macrocomposites, Microcomposites, Nanocomposites*, pp. 61 –72, 2003.

[30] R. A. Florentine, “Apparatus for weaving a three-dimensional article,” U.S. Patent US4312261A, Jan. 1982.

[31] R. M. Roberts and W. A. Douglas, “3 dimensional braiding apparatus,” U.S. Patent US5337647A, Aug. 1994.

[32] T. Kostar and T.-W. Chou, “A methodology for cartesian braiding of three-dimensional shapes and special structures,” *Journal of Materials Science*, vol. 37, pp. 2811 –2814, 2002.

[33] R. T. Brown and E. D. Ratliff, “Method of sequenced braider motion for multi ply braiding apparatus,” U.S. Patent US4621560A, Nov. 1986.

[34] R. T. Brown, “Braiding apparatus,” U.S. Patent US4753150A, May 1988.

[35] P. Popper and R. McConnell, “A new 3-d braid for integrated parts manufacturing and improved delamination resistance - the 2-step process,” in *The 32nd International SAMPE symposium and exhibition*, 1987, pp. 92–103, ISBN: 0938994344.

[36] W. Li and A. Shiekh, “Effect of processes and processing parameters on 3-d braided preforms for composites,” *SAMPE Q.; (United States)*, vol. 19:4, Jul. 1988.

[37] F. K. Ko, F. Schreiber, H.-J. Yand, and T. Gries, “Recent advancements in three-dimensional braiding,” in *Proceedings of the 1st World Conference on 3D-Fabrics and their Applications*, Manchester, UK, 2008.

[38] S. Green, M. Matveev, A. Long, D. Ivanov, and S. Hallett, “Mechanical modelling of 3d woven composites considering realistic unit cell geometry,” *Composite Structures*, vol. 118, pp. 284 –293, 2014. DOI: 10.1016/j.compstruct.2014.07.005.

[39] C. Ayranci and J. P. Carey, “Predicting the longitudinal elastic modulus of braided tubular composites using a curved unit-cell geometry,” *Composites: Part B*, vol. 41, pp. 229 –235, 2010. DOI: 10.1016/j.compositesb.2009.10.006.

[40] C. Sun and R. Vaidya, “Prediction of composite properties from a representative volume element,” *Composites Science and Technology*, vol. 56, no. 2, pp. 171–179, 1996. DOI: 10.1016/0266-3538(95)00141-7.

[41] Y. Wang and A. S. D. Wang, “Geometric mapping of yarn structures due to shape change in 3-d braided composites,” *Composite Science and Technology*, vol. 54, no. 4, pp. 359 –370, 1995. DOI: 10.1016/0266-3538(95)00059-3.

[42] T. Alpyildiz, “3d geometrical modelling of tubular braids,” *Textile Research Journal*, vol. 82, no. 5, pp. 443–453, 2011. DOI: 10.1177/0040517511427969.

[43] Y. T. Gao, F. K. Ko, and H. Hu, “Integrated design for manufacturing of braided preforms for advanced composites part i: 2d braiding.,” *Applied Composite Materials*, vol. 20, no. 6, pp. 1007 –1023, 2013, ISSN: 0929189X. DOI: 10.1007/s10443-012-9303-6.

[44] S. C. Queka, A. M. Waasa, K. W. Shahwanb, and V. Agaramb, “Analysis of 2d triaxial flat braided textile composites,” *International Journal of Mechanical Sciences*, vol. 45, no. 6-7, pp. 1077 –1096, 2003. DOI: 10.1016/j.ijmecsci.2003.09.003.

[45] G. W. Melenka, B. K. Cheung, J. S. Schofield, M. R. Dawson, and J. P. Carey, “Evaluation and prediction of the tensile properties of continuous fiber-reinforced 3d printed structures,” *Composite Structures*, vol. 153, pp. 866–875, 2016, ISSN: 0263-8223. DOI: 10.1016/j.compstruct.2016.07.018.

[46] S. Z. Sheng and S. van Hoa, “Modeling of 3d angle interlock woven fabric composites,” *Journal of Thermoplastic Composite Materials*, vol. 16, no. 1, pp. 45–58, 2003. DOI: 10.1177/0892705703016001206.

[47] C. Deng, J. J. Jiang, and L. C. Fang, “Algorithm design of four-step three-dimensional braided composite structures in matlab environment,” in *Applied Mechanics and Materials*, ser. Machine design and manufacturing engineering; (ICMDME 2013), vol. 365, Trans Tech Publications, 2013, pp. 1144 –1147. DOI: 10.4028/www.scientific.net/amm.365-366.1144.

[48] M.-Z. Zhang and H.-J. Li, “Automatically generated geometric description of 3d braided rectangle preform,” *Computational Materials Science*, vol. 39, no. 4, pp. 836 –841, 2007. DOI: 10.1016/j.commatsci.2006.10.010.

[49] O. I. Paul, “Modeling and simulation of three dimensionally braided composite and mechanical properties analysis using finite element method (FEM),” *The International Journal of Engineering and Science (IJES)*, vol. 3, no. 3, pp. 1 –8, 2014, ISSN: 2319 - 1813.

[50] S. Smith, D. Romanyk, P. Major, and C. Ayranci, “An investigation on the preparation and mechanical properties of three-dimensional braided composite orthodontic archwires,” *Journal of International Oral Health*, vol. 8, no. 5, pp. 554–559, 2016.

[51] C. Ayranci and J. P. Carey, “Predicting the longitudinal elastic modulus of braided tubular composites using a curved unit-cell geometry,” *Composites Part B: Engineering*, vol. 41, no. 3, pp. 229–235, 2010. DOI: 10.1016/j.compositesb.2009.10.006.

[52] T. Ting, *Anisotropic Elasticity: Theory and Applications*, 4. New York: Oxford University Press, 1996, vol. 63, p. 1056. DOI: 10.1115/1.2787237.

[53] B. A. A. E. H. Love, “Xvi. the small free vibrations and deformation of a thin elastic shell,” *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 179, pp. 491–546, 1888, ISSN: 0264-3820. DOI: 10.1098/rsta.1888.0016. eprint: http://rsta.royalsocietypublishing.org/content/179/491.full.pdf. [Online]. Available: http://rsta.royalsocietypublishing.org/content/179/491.

[54] T. Ishikawa and T. W. Chou, “Stiffness and strength behaviour of woven fabric composites,” *Journal of Materials Science*, vol. 17, no. 11, pp. 3211–3220, 1982. DOI: 10.1007/bf01203485.

[55] T. Ishikawa, K. Koyama, and S. Kobayashi, “Elastic moduli of carbon-epoxy composites and carbon fibers,” *Journal of Composite Materials*, vol. 11, no. 3, pp. 332–344, 1977. DOI: 10.1177/002199837701100307.

[56] T. Ishikawa, “Anti-symmetric elastic properties of composite plates of satin weave cloth,” *Fibre Science and Technology*, vol. 15, no. 2, pp. 127–145, 1981. DOI: 10.1016/0015-0568(81)90066-x.

[57] T. Ishikawa and T.-W. Chou, “Elastic behavior of woven hybrid composites,” *Journal of Composite Materials*, vol. 16, no. 1, pp. 2–19, 1982. DOI: 10.1177/002199838201600101.

[58] W. Johnson, J. Masters, I. Raju, and J. Wang, “Classical laminate theory models for woven fabric composites,” *Journal of Composites Technology and Research*, vol. 16, no. 4, p. 289, 1994. DOI: 10.1520/ctr10589j.

[59] G. W. Du and F. K. Ko, “Geometric modeling of 3-d braidied preforms for composites,” in *Preceedings of 5th Textile Structural Composites Symposium*, Philadelphia, PA: Drexel University.

[60] L. Tong, A. Mouritz, and M. Bannister, *3D Fibre Reinforced Polymer Composites*. Elsevier Science, 2002, ISBN: 0-08-043938-1.

[61] C. E. Hage, R. Younès, Z. Aboura, M. Benzeggagh, and M. Zoaeter, "Analytical and numerical modeling of mechanical properties of orthogonal 3d CFRP," *Composites Science and Technology*, vol. 69, no. 1, pp. 111–116, 2009. DOI: 10.1016/j.compscitech.2007.10.048.

[62] S. Green, A. Long, B. E. Said, and S. Hallett, "Numerical modelling of 3d woven preform deformations," *Composite Structures*, vol. 108, pp. 747–756, 2014. DOI: 10.1016/j.compstruct.2013.10.015.

Chapter 4

Design of a Rotary Three-Dimensional Braiding Machine and Analysis of the Braided Structure

4.1 Introduction

In this chapter, the design, development and results of creating an in-house three-dimensional rotary braiding machine is discussed. Summaries for the various components that assemble to form the 3D braider, as well as the description of how the parts function is outlined. Finally, the production of 3D braided samples is described, and the analysis of the braided structures is provided.

4.2 Machine Development

For this work, an in-house three-dimensional rotary braider is designed and developed. This allows for further study of the geometric properties of the 3D braided structure of braids produced from a 3D rotary braider. Additionally, the development of an in-house machine allows for future work to be developed from the material presented.

For the research presented in this work, a set-up of three by three cams is used. This provides the means to analyse the properties of the smallest square braided structure that also possesses the three sub- unit cells discussed in Chapter 6. However, the machine is designed to accept up to 7 by 7 cams, meaning that the machine can easily be adapted to produce larger and more complex braided structures. A summary of the maximum specifications of the machine are shown in Table 4.1, while the full braider drawings are presented in Appendix A.2.

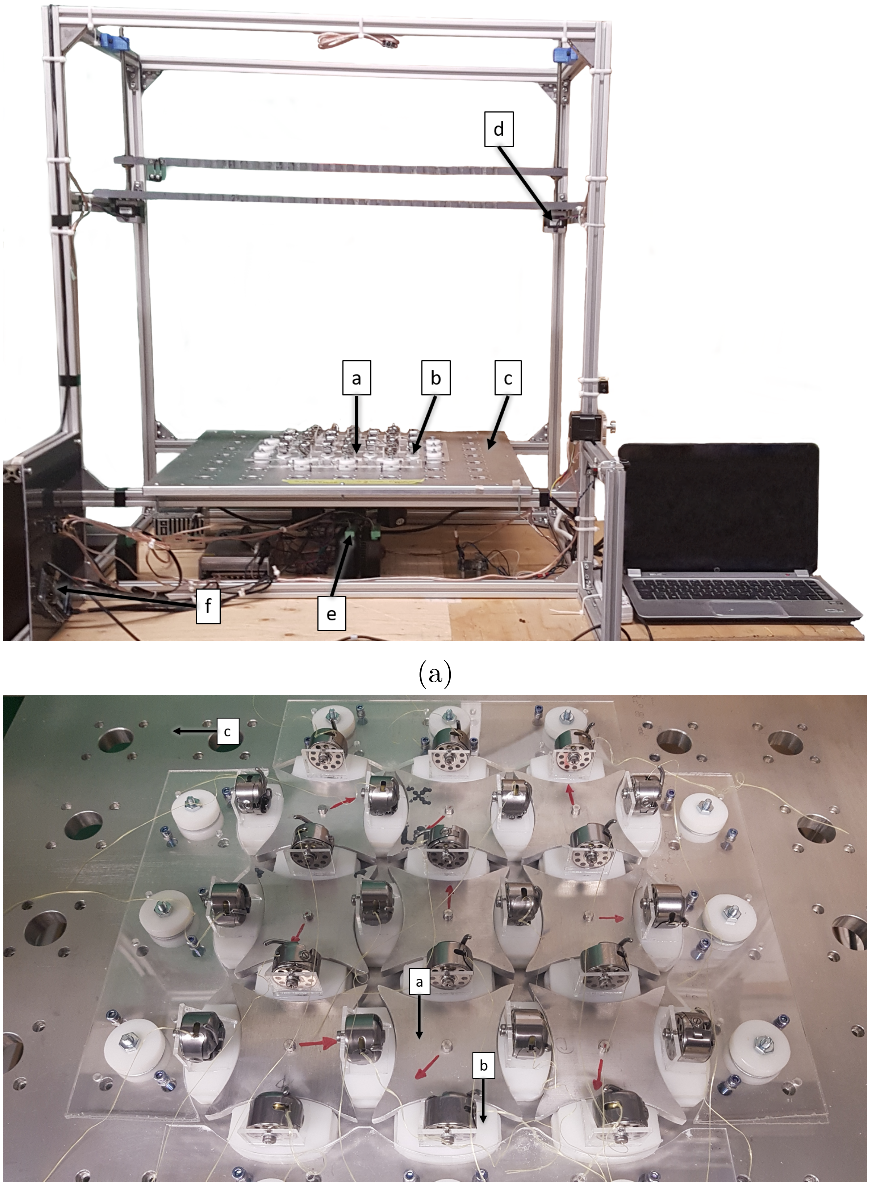
Table 4.1: Summary of the maximum machine specifications.

Property	**Value**	**Units**
Number (Cams)	7 by 7	-
Number (Yarns)	112	Total
Uptake	180	[mm]

The braiding machine is comprised of a number of different components that work together to form the 3D braided structures. The major components include: the cams and carriers, which move the yarns to their new positions above the base plate; the uptake mechanism with the zeroing mechanism, which keeps the braid forming consistently with the ability to quick reset itself; the geared motors and drivers that provide the electro-mechanical power to the system; and the control system, which acts as a brain for the machine. All of the machine components can be seen in Figure 4.1. In the following sections, an overview of the different assemblies that create the braider, as well as details on the individual parts that make the assemblies are provided.

4.2.1 Base Plate

The base plate serves as the level surface where the carriers run along, as well as the attachment point for the geared motors and cam system. The base plate is CNC cut from a two-foot square, 1/2" thick aluminium sheet. While the base plate is supported on two opposite ends, by the frame of the machine, the choice of 1/2" thick aluminium sheet means that the base plate is stiff enough such that deflection, due to bending, is negligible and does not affect the operation of the machine or formation of braids. A CNC is used to accurately create holes for the motor shaft and the holes for attaching the motor assemblies to the base plate. The motor assembly holes are created as a counterbore to ensure the head of the socket head screws lay beneath the surface that the carriers slide on. This ensures that the flange on the carriers can never run into the head of the screws.



(a)

(b)

Figure 4.1: Layout of the custom built three-dimensional braiding machine including: a) the cams, b) the carrier, c) the base plate, d) the uptake mechanism, e) the motors and drivers, and f) the control system.

4.2.2 Cams and Carrier

The cams and carriers are what drive the braiding process; the carriers are what carry the yarns while the cams push the carriers around the base plate. The cams, and carriers for this rotary 3D braider are based off designs presented by Ko, Pastore, and other researchers [7, 8, 63]. The cams are allowed to rotate in a clockwise or counter-clockwise motion that moves the carriers and their yarns around the cams forming the braid.

Before the full-scale production of the braider is started, a scaled down version is rapid prototyped using additive manufacturing as a proof of concept. Once the proof of concept, shown in Figure 4.2, is shown to successfully move the carriers with the cams, the design is adapted for subtractive manufacturing techniques. Using subtractive manufacturing, the time required to produce the parts is significantly reduced, however, more forethought is required to ensure the parts could be properly assembled. In the following sections, the design considerations for both the cams and carriers will be discussed.

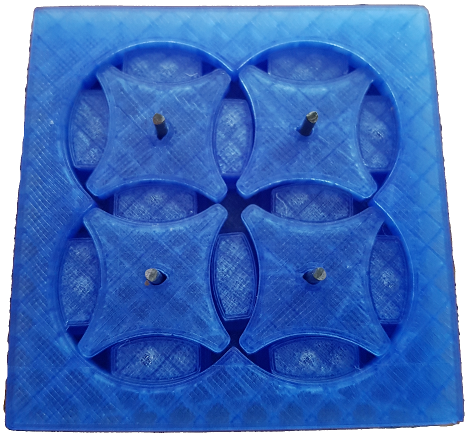
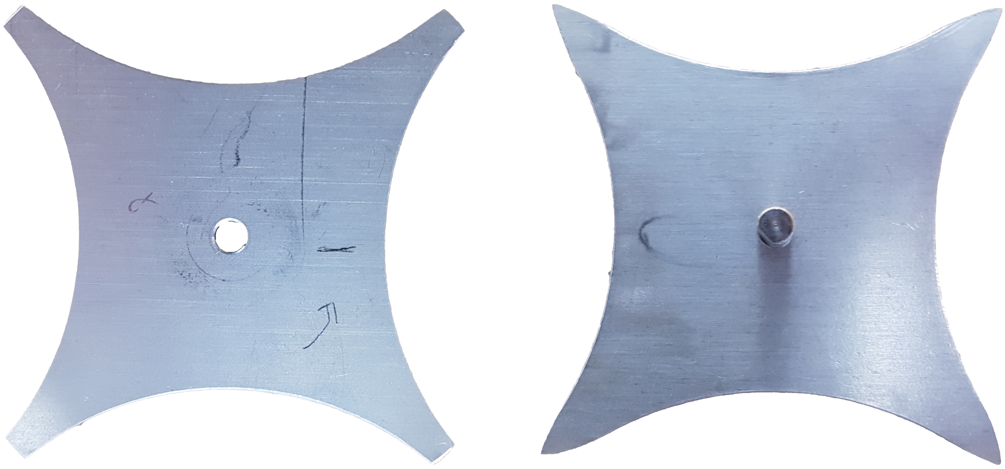


Figure 4.2: Prototype of the cam and carrier mechanisms.

Cams

The cams or horn gears are used to move the carriers around the base plate. The cams, which are driven by a geared stepper motor, are able to turn in 90° intervals that corresponds to the position of the adjacent cams. The cams are waterjet cut from a 1/4" aluminium sheet, with the hole to attach the shaft from the motor assembly reamed to create a press fit. To significantly decrease the chance of the system jamming, the corners of the cams are chamfered and smoothed. The result of the chamfering can be seen in Figure 4.3. This will allow the carriers that interact with the cams to automatically realign themselves as they are moved around.



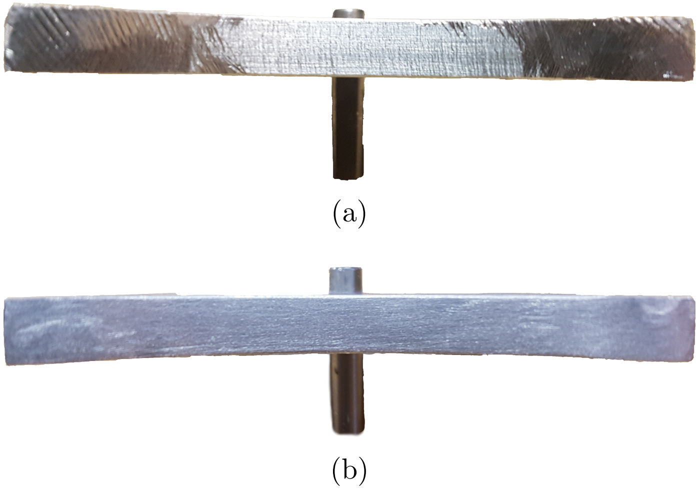
(a) (b)

Figure 4.3: Cams before (a) and after (b) chamfering.

Because the chamfering and waterjetting processes produce a very rough surface finish (see Figure 4.4a), further machining is required to finish the surfaces. The surfaces are finished to 150 microns, and the direction of the surface finishing is set to the same as the movement of the mechanism. The results of the surface finishing process are shown in Figure 4.4.

Carriers

The carriers are the mechanism that holds and carry the yarns that make the braid. The carriers are moved around the base plate by means of the cams and are able to control the tension in the yarns during the braiding process. In this section, the carriers are broken down into two sub-assemblies (the base of the carriers, and the bobbin and tensioning mechanism) and the functions of each of the sub-assemblies



(a)

(b)

Figure 4.4: Cams before (a) and after (b) surface finishing.

are highlighted.

The base of the carriers, shown in Figure 4.5, function to provide a low friction interface with the base plate to ensure that the carriers provide little resistance as they are pushed around the base plate. Additionally, the base of the carriers provides a flange that interfaces with the cams; this flange is designed to prevent the carriers from lifting up during the braiding process when the tensions become high in the yarns. The base of the carriers is created from three parts, two similar pieces and a rod to align all of the components.

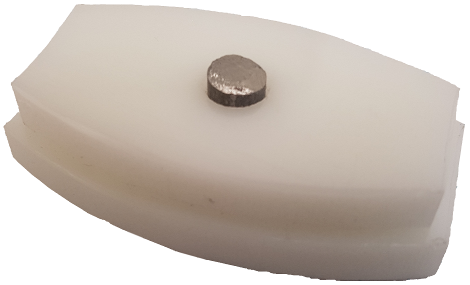


Figure 4.5: Base of Carrier.

The two similar pieces are waterjet cut from ultra high molecular weight polyethylene (UHMWPE). This material is chosen to provide the low friction interface between the cams and the carriers, and the base plate and the carriers. The low friction occurs due to the combination of UHMWPE's extremely low surface energy with most materials, and great wear performance, and self-lubricating behaviour [64, 65]. The high wear resistance, low coefficients of frication, and self-lubricating behaviour for UHMWPE makes it suitable for the continuous sliding that occurs while the carriers are moved around by the cams. To assemble the parts of the base together, a small rod is used for alignment while the surfaces are adhered together by a cyanoacrylate based adhesive. The alignment rod is left protruding from the top of the assembly to allow for the alignment of the bobbin and tensioning mechanism discussed next.

The second part of the carrier assembly is the bobbin and tensioning mechanism, shown in Figure 4.6. This assembly provides support for the bobbin, which houses the yarn, and provides the tension for the yarns, which assists in providing a consistent braided structure. The bobbin used to house the yarn is a standard sewing bobbin used in sewing machines. Sewing machine bobbins are chosen to hold the yarn due to the relatively small form, as well as the ready availability of the part. Additionally, the bobbins are acquired with their bobbin cases, which have a built-in adjustable tensioning mechanism removing the need for a separate tensioning mechanism.

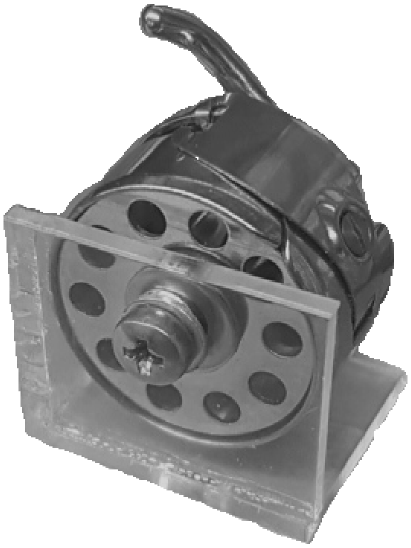


Figure 4.6: Tension mechanism.

To support the bobbin a mount is designed and cut from acrylic and also featured a hole on the bottom for attaching the assembly to the base of the carrier. The support for the bobbin assembly is precision cut from 1/16" acrylic sheet by a commercial laser cutter [66] and assembled using a solvent based cement (Weld-On 4 Acrylic Adhesive). To provide the tension in the yarns that is required to produce consistent braids, the leaf-spring mechanism on the bobbin case is utilized. This allows for easy adjustments to be made in the tension of the yarns.

Finally, the two sub-assemblies for the carrier are bonded together using the rod from the base again for alignment. The bonding is performed utilizing the same cyanoacrylate-based adhesive that is used to bond the two haves of the base together. Next, once the carrier is assembled and the yarns are loaded on to the bobbins, the yarns need to be attached to the uptake mechanism.

4.2.3 Uptake Mechanism

The uptake mechanism is what pulls the yarn to ensure the braid is forming in a consistent manner. The uptake mechanism is designed to progress the braid up during the braiding process, this ensures that a consistent braid is formed, and the formation of the braid occurs at a single point (the braid formation point). Additionally, the uptake mechanism is able to control the braid angle of the braided structure; this is preformed by adjusting the speed of the uptake mechanism. By increasing or decreasing the speed of the uptake mechanism, the braid angle can be reduced or increased, respectively. This leads to one of the ways in which tailorability is introduced into the braided structure.

The uptake mechanism is divided into six parts the base, the lead-screw linear actuators, the cross bar, the motor driver, the plain bearing supports, and the endstop feedback system, shown in Figure 4.7. Similar to the cam system, the uptake mechanism is driven by stepper motors that drive a lead-screw and thus also drives the crossbar up or down. The lead-screw linear actuators, full specifications are available in Appendix A.1, which are used to drive the uptake mechanism, are mounted to the base, which affixes the uptake mechanism to the frame of the braiding machine. The travelling nut of the lead-screw linear actuator is attached to the crossbar and causes the crossbar to travel up and down depending on the direction the lead screws are driven. To support the ends of the lead screws and to prevent unnecessary deflection, the ends are supported in by two plain bearings that are attached to top of the braiding machine's frame.

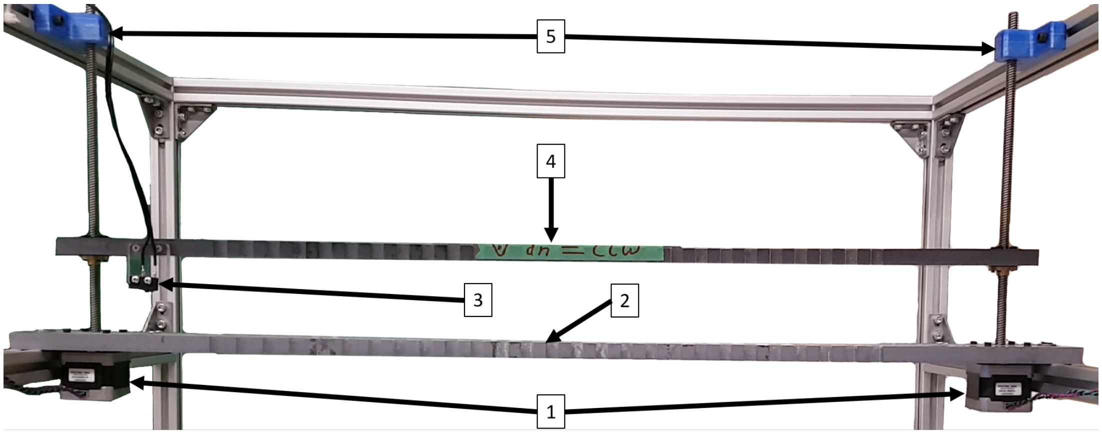


Figure 4.7: Braider uptake mechanism showing 1) the linear actuators, 2) the base of the uptake mechanism, 3) the end stop switch, 4) the cross bar, and 5) the plain bearing supports.

Zeroing Mechanism

A zeroing mechanism is designed to assist in the rapid resetting of the machine, which allows for faster production braided samples. By sensing if the crossbar has reached the lowest position, the machine is able to reset the linear actuator. Further, by understanding the zero point, the position of the crossbar can be tracked through the braiding process and can be moved to any position. The mechanism is able to sense when the machine has returned to its initial state by the use of an end stop (momentary switch), that becomes depressed when the mechanism reaches its lowest position. The microcontroller is able to poll the endstop and determine when the mechanism has reached its reset position and stop the motors.

4.2.4 Geared Motors and Driving Mechanism

The geared motors are what turn the cams on the base plate. Each motor assembly, shown in Figure 4.8, consists of a 200-step stepper motor and a 19.19:1 planetary gear reducer[67]. The geared motors are chosen because they use the compact NEMA 17 motor form factor while still providing enough torque to turn the cams. For complete specifications on the motors used, please refer to Appendix A.1, and for the complete electrical documentation please refer to Appendix A.3.

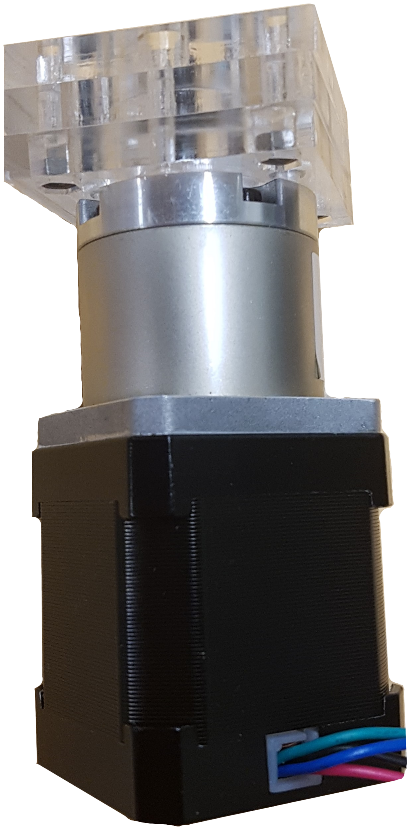
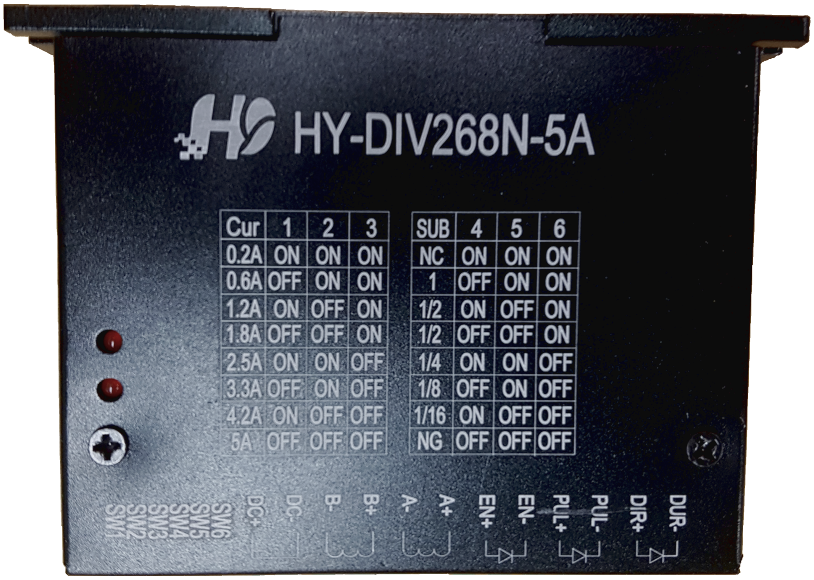


Figure 4.8: Geared stepper motors used to drive the cams.

The motors used to drive the cams are divided into two groups, those that drive the even cams and those that drive the odd cams, however, it is noted that provided more drivers, each motor could be independently driven, and more complex braiding patterns could be achieved. All of the motors of a specified group are wired in series to each other to make all of the motors of the same group step together. Additionally, this lowers the required amount of current to drive all of the motors and allows all the motors of a single group to be driven by a single motor driver, shown in Figure 4.9.



Cur	1	2	3
0.2A	ON	ON	ON
0.6A	OFF	ON	ON
1.2A	ON	OFF	ON
1.8A	OFF	OFF	ON
2.5A	ON	ON	OFF
3.3A	OFF	ON	OFF
4.2A	ON	OFF	OFF
5A	OFF	OFF	OFF

SUB	4	5	6
NC	ON	ON	ON
1	OFF	ON	ON
1/2	ON	OFF	ON
1/2	OFF	OFF	ON
1/4	ON	ON	OFF
1/8	OFF	ON	OFF
1/16	ON	OFF	OFF
NG	OFF	OFF	OFF

Figure 4.9: Current limiting stepper driver based on the TB6600 chip.

The motor drivers are a HY-DIV268N-5A; these motor drivers are based on the Toshiba TB6600HG PWM chopper-type bipolar stepping motor driver integrated circuit (IC) and are also able to isolate the braiders control circuitry from the high voltage used to power the motors. Because the drivers are based on the TB6600 IC, this allows the motors to be run at a higher voltage than the motor is rated for. This is possible due to the PWM chopper-type current control that monitors the current draw from the motor and when it reaches above a specified threshold the power to the motor is momentarily interrupted and the cycle is allowed to repeat. By running the motors at a higher voltage, the step response of the motor is shortened allowing for faster stepping. And by using the chopper-type driver, the motor is protected from overheating and becoming damaged from the over-voltage.

Motor Adaptor

For adaptability, the motor mounting holes in the base plate are made to the NEMA 17 motor specifications. However, due to the inclusion of a motor with a planetary gear reduction, the mounting holes of the motor do not match those used for the base plate. To remedy this, a motor adaptor is designed from three pieces of 1/8" laser cut acrylic. The three pieces work together to securely fasten the motors to the base plate, while making sure the heads of the screws remain below the surface and nuts are captive. The inclusion of the adaptor, additionally, provides additional space require for the motor coupling that is required to change the shaft size to that of a NEMA 17 stepper motor. The assembled motor adaptor can be seen attached to the motor in Figure 4.8.

4.2.5 Control System

The control system is primary broken into two components: the master system that determines what tasks need to be executed and when they will be executed, and the slave system that executes the tasks from the master system. The braiding machine can be controlled from any Windows® computer that will act as the master system; the computer runs an application designed and written in MATLAB® [68] that sends serial commands to a connected microcontroller.

An open source, commercially available microcontroller board, Arduino: Mega [69], is chosen to interface with the master system and send the logic signals to the various drivers and systems that drive the braiding machine. The Arduino microcontroller is chosen to control the braider, because it has open-source hardware and software. Because of this, the programming of the microcontroller is straight forward. Additionally, the Arduino: Mega offers many extra GPIO pins that can be used to expand the machine later.

To control the braiding machine, commands are sent from the controlling computer to the microcontroller using the attached USB Type-B connector. Commands are sent in an eight-byte chunk that are interoperated by the microcontroller and translated to the commands/functions to be executed, see Figure 4.10 for an overview of this process. While the microcontroller is executing the commands, both the microcontroller and the controlling computer wait for commands to finish to ensure that the command buffer does not overflow. The firmware used on the microcontroller is available in full in Appendix B.2.

4.3 Braid Production

Braid production is started by entering the settings/parameters on a computer, and continues with the computer sending appropriate commands to a microcontroller. The production of the braids is assisted by a computer program written in MATLAB®. The program provides an easy to use interface that allows the user to control the different aspects of the braiding process. Once started, the program is able to communicate to the connected braiding machine by the use of serial communication. The protocol for communicating with the braiding machine is custom made to allow better control over the necessary parameters.

The serial protocol utilizes eight byte commands that are able to control and create different environments on the microcontroller. The different commands are all built using the same structure. The first byte chooses the library, the second selects the function or control sequence, and the remaining six bytes set various options and provide the data necessary to run the functions or control sequences. To ensure the control system and microcontroller are synchronized, all variables are stored on both systems using an object or class structure and are updated together. Because the computer knows the state of all the objects and variables on the microcontroller, this allows the computer to display any available information at any time. And further allows the easy implementation of software monitoring of specific machine parameters.

The production of the braid is used to determine the geometric properties of the braided structure. In Figure 4.11 a braid is shown being formed by the 3D rotary braiding machine. In the following subsection, the measurements of the external braid angle and pitch of the braid are determined.

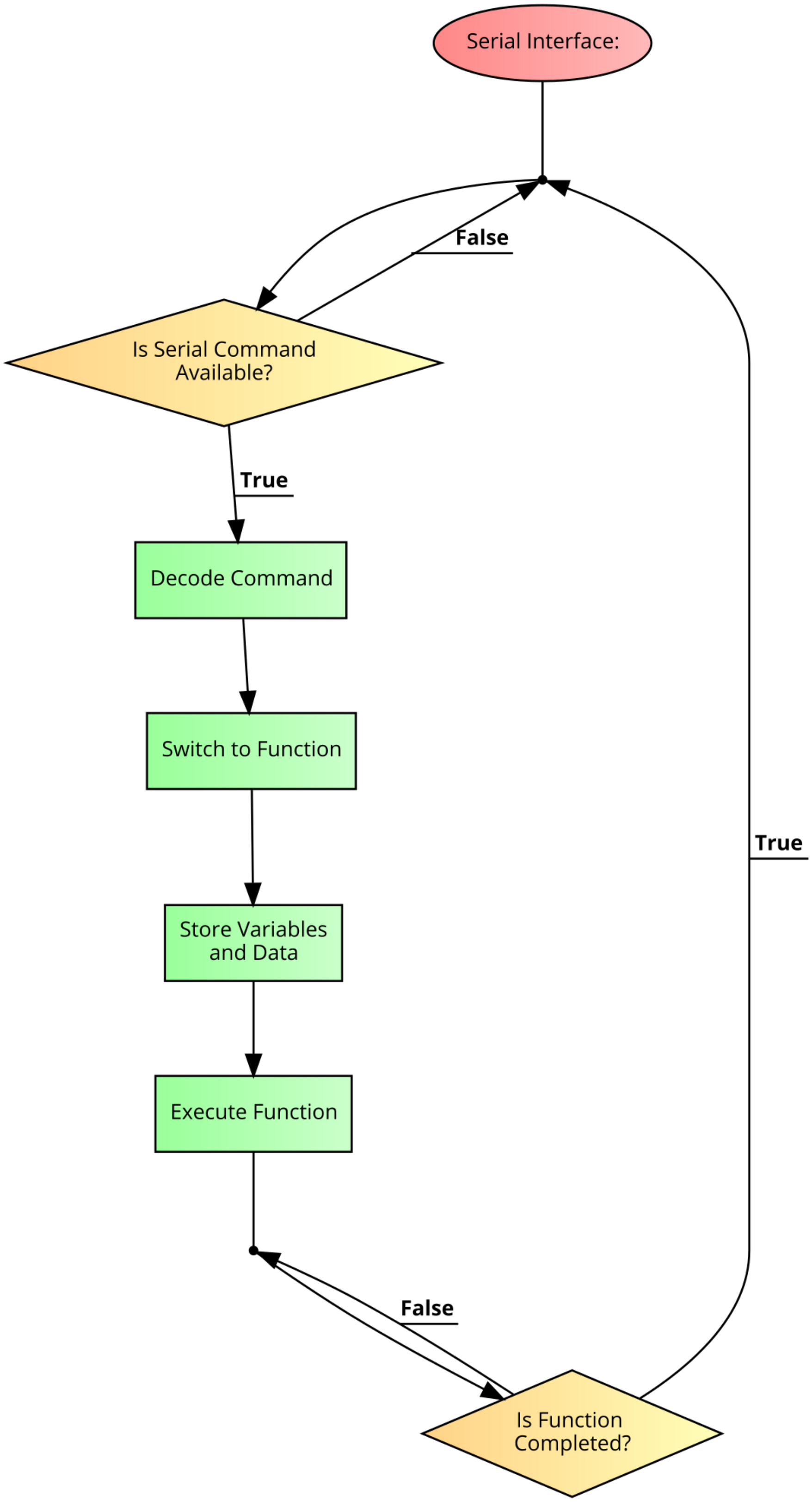


Figure 4.10: Flow diagram of the Arduino firmware.

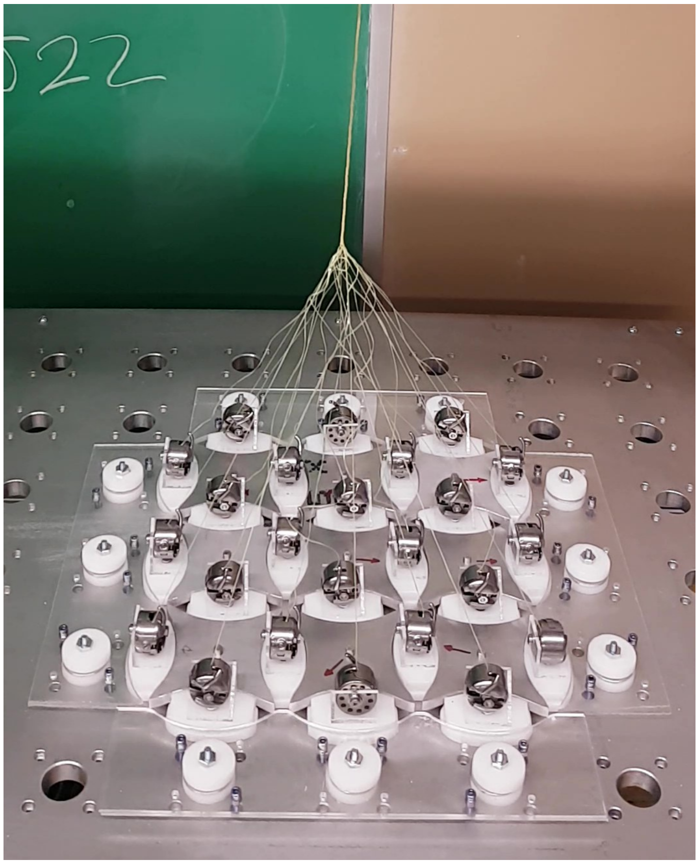


Figure 4.11: Produced braid before removal from the machine.

4.3.1 Measured Geometry

The rotary three-dimensional braider is used to create a sample three by three non-impregnated specimen from 200-denier Aramid fibres (Kevlar® from Fiber-Line®, USA). These specimen's geometric properties are later analysed. For the specimen, the external braid angle is measured, as well as the pitch of the braid.

To measure the external braid angle and the pitch of the braid, the freely available image measurement software, ImageJ [70], is utilized. To determine each of the parameters, five measurements are taken along the length of the braid. Of these parameters, the pitch length of the braid is chosen to generate a geometric model and is further discussed in Chapter 5.

External Braid Angle

To measure the external braid angle, a picture of the specimen is taken with a reference background. The background shows a millimetre grid, and provides a reference for the measurements taken. For the measurement of the external braid angle reference lines are drawn from the base of a yarn being measured to the same point of the next yarn. This reference line represents the instantaneous braid axis. Next a line is drawn from the base of the yarn parallel along the length of it. This creates the two lines required to determine the external braid angle; the five measurement points are shown in Figure 4.12.

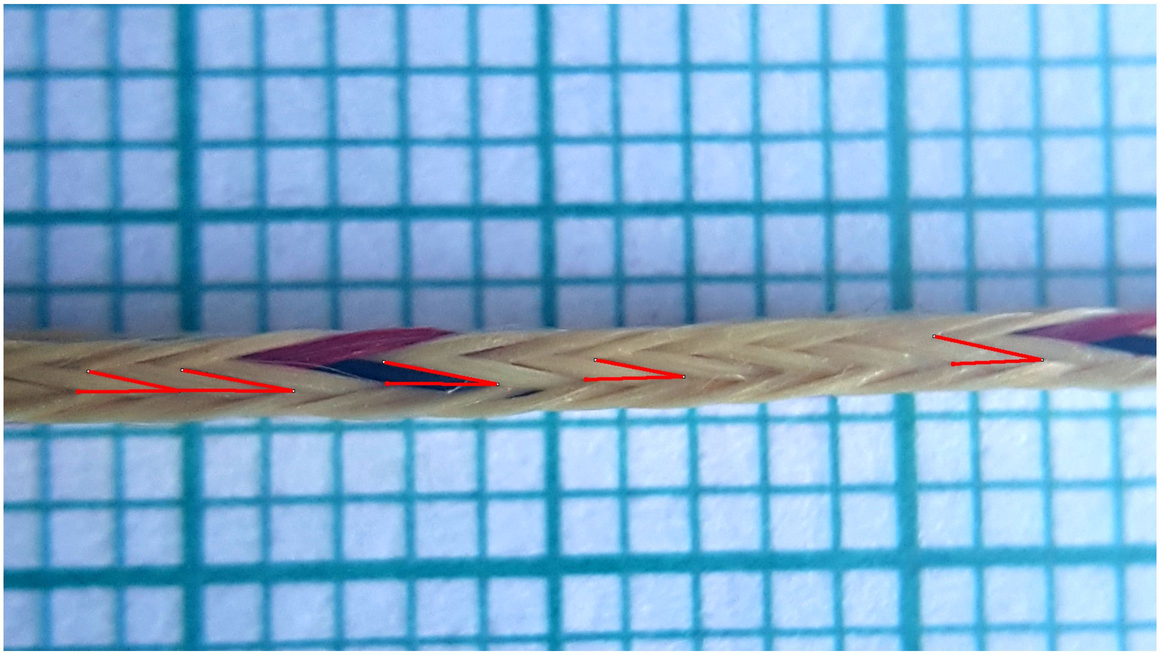


Figure 4.12: Measurement of the external braid angle.

From these points the, software is able to determine the average external braid angle to be 12.19 ± 1.87°. A full summary of each of the individual measurements is shown in Table 4.2.

Table 4.2: Summary of the measurements for the external braid angle.

Measurement	**Angle**
1	10.763
2	12.624
3	10.412
4	12.152
5	15.018
Average	12.194

Braid Pitch

To measure the braid pitch, a similar procedure to the external braid angle is implemented. A picture of the specimen is taken with a millimetre grid in the background to provide a reference for the measurements. For the measurement of the braid pitch, the scale is first set in the software using the grid to specify the length for one millimetre. This is done by drawing lines over the grid and using the lines as known distances for converting the number of pixels to real world measurements. Next the measurement lines are drawn from the base of a yarn extending forward to the base of the next yarn. These lines are shown in Figure 4.13 with the results summarized in Table 4.3.

From these points the, software is able to determine the average braid pitch to be 1.43 ± 0.07 mm. A full summary of each of the individual measurements is shown in Table 4.3.

Table 4.3: Summary of the measurements for the braid pitch.

Measurement	**Length**
1	1.420
2	1.379
3	1.446
4	1.367
5	1.546
Average	1.432

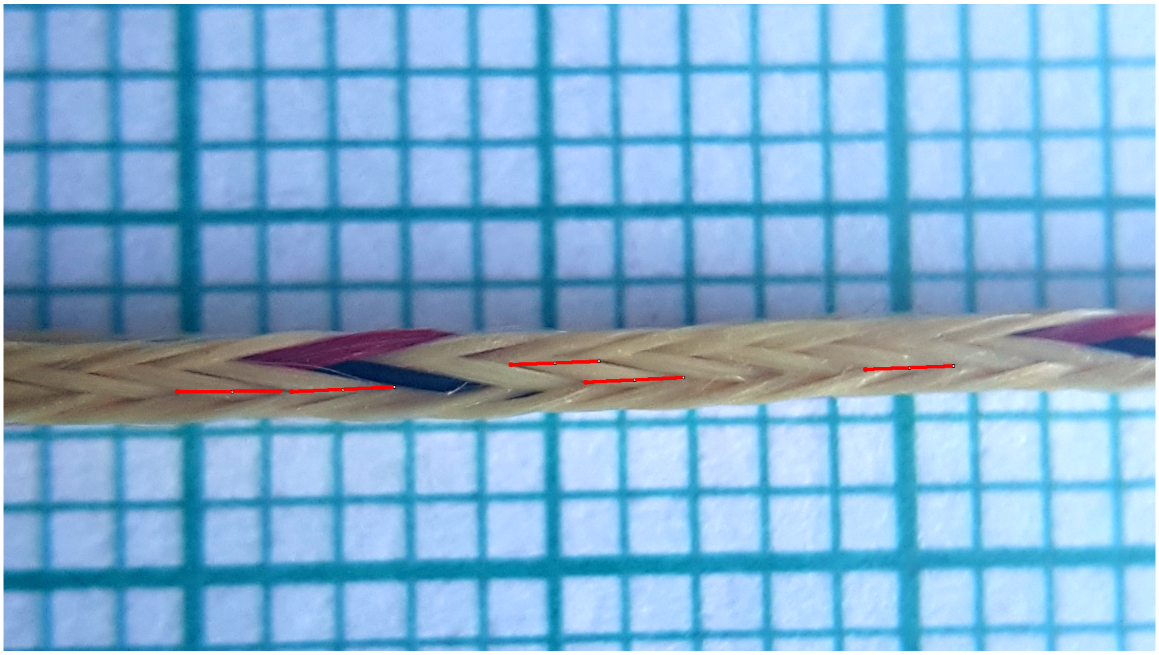


Figure 4.13: Measurement of the height of the unit-cell (braid pitch).

4.4 Conclusions

In this chapter, is a summary of the design and manufacturing of an in-house rotary 3D braiding machine. Additionally, the design decisions are analysed, and the justifications are provided for the decisions. An introduction is provided to demonstrate how the machine is controlled, as well the machine is shown to be able to produce 3D braided specimens. Finally, the measurement of the external properties of the braid is performed and the external braid angle is determined to be 12.19° and the braid pitch to be 1.43 mm for a 200-denier Kevlar® braid.

References

[7] F. Ko, “Tensile strength and modulus of a three-dimensional braid composite,” in *Composite Materials (Seventh Conference)*, 1984, pp. 392 –403.

[8] F. Ko and C. Pastore, “Structure and properties of an integrated 3-d fabric for structural composites,” in *Recent Advances in Composites in the United States and Japan*, ASTM International, 1985, pp. 428–439. DOI: 10.1520/stp32805s.

[63] M. Tsuzuki, “Three dimensional woven fabric with varied thread orientations,” U.S. Patent US5348056A, Sep. 1994.

[64] G. Yilmaz, T. Ellingham, and L.-S. Turng, “Improved processability and the processing-structure-properties relationship of ultra-high molecular weight polyethylene via supercritical nitrogen and carbon dioxide in injection molding,” *Polymers*, vol. 10, no. 1, p. 36, 2017. DOI: 10.3390/polym10010036.

[65] K Ramani and N. Parasnis, “Process-induced effects in compression molding of ultra-high molecular weight polyethylene (UHMWPE),” in *Characterization and Properties of Ultra-High Molecular Weight Polyethylene*, ASTM International, 1998, pp. 5–5–19. DOI: 10.1520/stp11906s.

[66] Universal Laser Systems, *Vls3.50*, Hardware. [Online]. Available: https://www.ulsinc.com.

[67] O. StepperOnline, *Nema 17 bipolar stepper motor (17hs19-1684s-pg19)*, Hardware. [Online]. Available: https://www.omc-stepperonline.com/nema-17-stepper-motor-bipolar-l48mm-w-gear-raio-191-planetary-gearbox-17hs19-1684s-pg19.html.

[68] The Mathworks, Inc., *Matlab 2017a r9.2*, Software, 2017. [Online]. Available: http://www.mathworks.com.

[69] Arduino, *Arduino*, Hardware, 2017. [Online]. Available: http://www.arduino.cc/.

[70] National Institute of Health, *ImageJ*, Software, 2018. [Online]. Available: https://github.com/imagej/imagej1.

Chapter 5

Design of a Geometric Model for Three-Dimensional Rotary Braiding

5.1 Introduction

An outline of the procedure of producing a geometric, as well as a geometric computer aided design (CAD) model is provided in this chapter. To facilitate faster creation of these models, a computer program is developed. The program implements machine emulation to generate the paths of the yarns that are required to develop the geometric model.

The geometric model is imported into a parametric CAD software using another program that is written in the CAD software's own programming language. These programs are able to work together to automate the creation of geometric models for 3D braided composites. In the following sections, an overview of the mathematics of the geometric models, as well as the function of the custom programs are provided.

5.2 Path and Geometric Model Generation

To be able to create the geometric model of the three-dimensional braided structure, the paths of each of the individual yarns must be known. In this section a set of points (x, y, z) are produced by the program that can later be interpreted by three-dimensional modelling/design software to generate the geometric CAD model. The geometric CAD model can be further processed to create the unit and sub-unit cells required for Chapter 6.

The path of the yarns is created utilizing a method known as machine emulation. Machine emulation is the process where the machine and its processes are modelled and replicated to generate the path of the yarns [13, 16, 47]. This method was first introduced by Tolosana *et al.* for 3D braided structures and later used by Deng *et al.* and Yang *et al.* to further analyse 3D braided structures. All of the researchers utilizing this method, have used it to predict the paths of the yarns formed from track and column braiders.

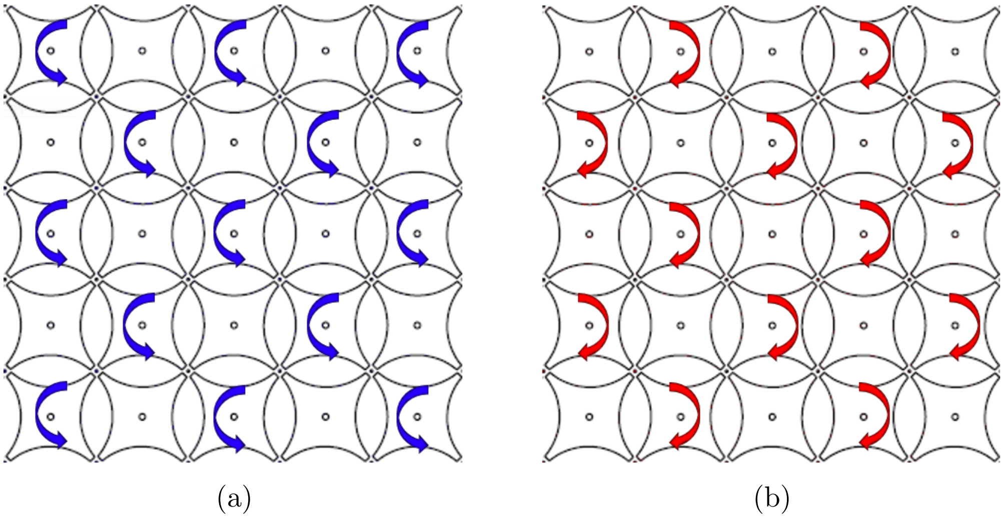
For this research, the machine emulation method is implemented to determine the paths of the yarns for a braid produced from a rotary 3D braider. In the next subsections the braid pattern, which describes the process where the braid is formed, and an overview of the in-house program used to automate the process of creating the geometric and geometric CAD models is discussed.

5.2.1 Braid Pattern

The braiding pattern is a reference to the steps that are taken and repeated to form the braided structure. There are many different patterns that can be accomplished with 3D braiders. For track and column braiders, the dominate pattern used is the 4-step process. While the 4-step is more commonly used due to its simplicity, it lacks the ability to form largely complex shapes. To mitigate this, the universal method that includes a multiple of four steps to complete the braiding pattern was introduced [32]. This new universal method sacrifices speed to be able braid complex shapes. By dividing the braided structure into smaller sections and braiding the smaller sections then braid the sections together near net shape geometry can be achieved [32].

For rotary braiders, the patterns can be quite simple, as in the case of the square braider where the braid pattern can be considered a single step that can operate continuously [22]. However, the braid patterns for rotary braiders can also become more complex having multiple steps or, in the case of producing complex shapes, also adjusting the number of active cams can change the braid pattern. Additionally, the braiding pattern can be changed by varying the number of cams rotating, which cams rotate, how much of a rotation is performed, and the order in which the rotations are performed.

For this research, a simple two-step process, shown in Figure 5.1, is used to create the braided structures. The two-step process consists of a counter-clockwise rotation of the 'odd' cams (Step-1) followed by a clockwise rotation of the 'even' cams (Step-2). After the second step is completed, the braid has returned to a similar position as the starting position. This pattern can now be repeated to form the 3D braided composite structure.


(a)

(b)

Figure 5.1: Braiding pattern show (a) the first step (odd rotation) and (b) the second step (even rotation)

By understanding how the braid is formed, the geometry of the braid can be determined. Using a process called machine emulation [11], the steady state formation of the braid can be determined. This process involves emulating the machine's movements, and from the movement of the machine calculate the positions of the yarns in the formed braid. Once the position of the yarns is known, the geometry of the braid can be determined before the braid is created. This allows for further analysis of the braided structure and the prediction of the material properties before the braid is created.

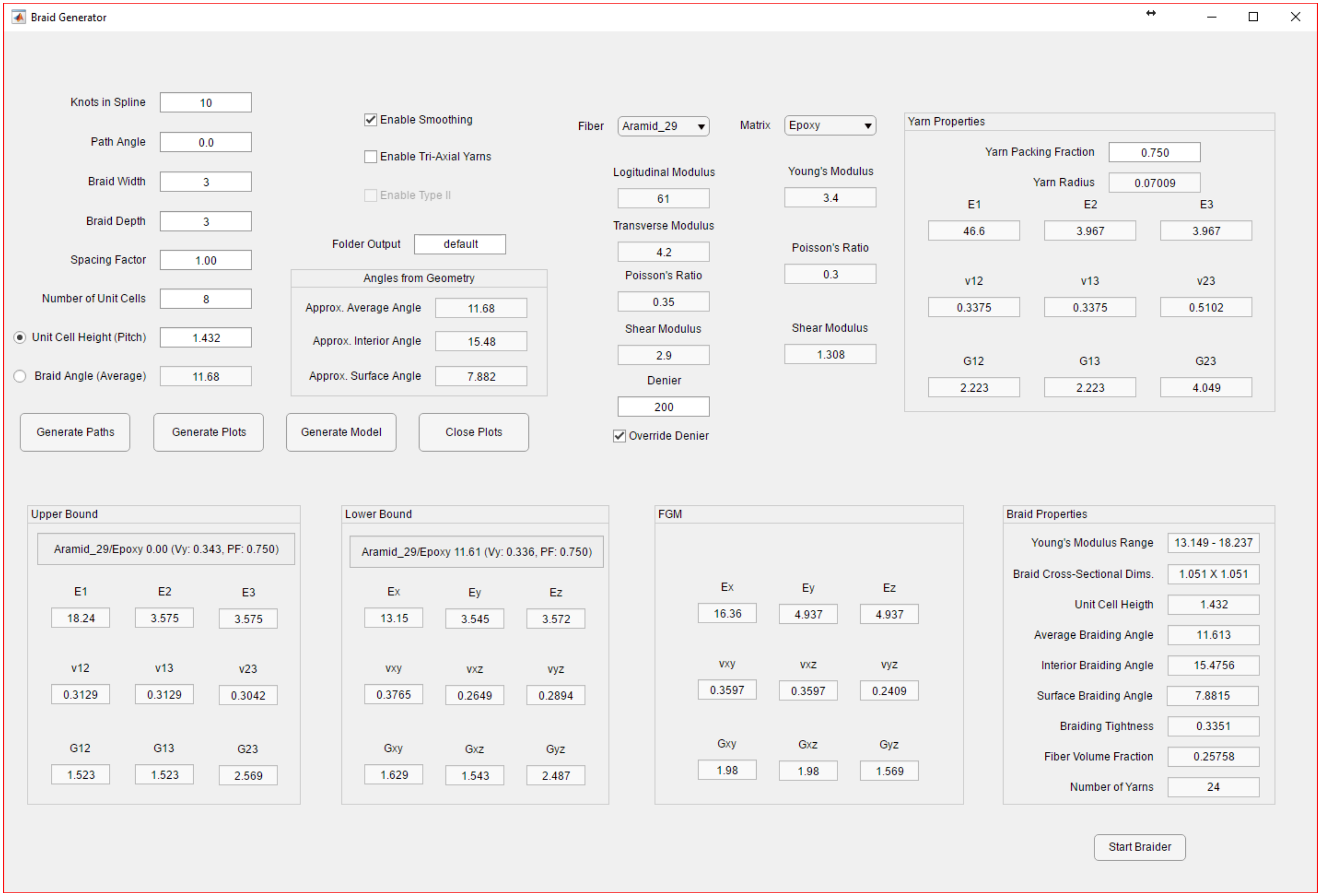
A discussion on the implementation of machine emulation on a rotary 3D braider is made in the following subsections. Further, an introduction to the mathematics of the geometric model is made. Finally, the creation of a geometric CAD model is discussed.

5.2.2 Program for Geometric and CAD Model

For this study, a program is designed to automate the creation of the geometric model, as well as the creation of a computer-aided design (CAD) model of the braided structure. Additionally, the program is designed to provide an understanding of the estimated properties of the braided structure by providing theoretical limits, and the approximation by the implementing of the widely used fabric geometry model (FGM). The program itself is divided into a few different sections: material properties, braid definition, estimated properties, path generation, visualization, and CAD export that can be seen in Figure 5.2. For the full code, please refer to Appendix C.2.

Material Properties

To assist in the design of three dimensional braided structures, the properties of the individual materials for the braid must be known. The properties for various materials have been recorded and stored in a separate file that is read in when the program is executed, see Appendix C.1. This library of materials is coded in the extensible markup language (XML) coding format and is divided into two sections: fibre properties and matrix properties. The library is written in a way that assumes the fibres to be anisotropic (specifically, transversely isotropic) and the matrix is assumed to be isotropic. For the fibre material properties, the name of fibre, the axial Young's modulus, transverse Young's modulus, Poisson's ratio, shear modulus,



Braid Generator

Knots in Spline: 10

Path Angle: 0.0

Braid Width: 3

Braid Depth: 3

Spacing Factor: 1.00

Number of Unit Cells: 8

● Unit Cell Height (Pitch): 1.432

○ Braid Angle (Average): 11.68

☑ Enable Smoothing

□ Enable Tri-Axial Yarns

□ Enable Type II

Folder Output: default

Angles from Geometry

Approx. Average Angle: 11.68

Approx. Interior Angle: 15.48

Approx. Surface Angle: 7.882

Fiber: Aramid_29

Logitudinal Modulus: 61

Transverse Modulus: 4.2

Poisson's Ratio: 0.35

Shear Modulus: 2.9

Denier: 200

☑ Override Denier

Matrix: Epoxy

Young's Modulus: 3.4

Poisson's Ratio: 0.3

Shear Modulus: 1.308

Yarn Properties

Yarn Packing Fraction: 0.750

Yarn Radius: 0.07009

E1	E2	E3
46.6	3.967	3.967
v12	v13	v23
0.3375	0.3375	0.5102
G12	G13	G23
2.223	2.223	4.049

Upper Bound

Aramid_29/Epoxy 0.00 (Vy: 0.343, PF: 0.750)

E1	E2	E3
18.24	3.575	3.575
v12	v13	v23
0.3129	0.3129	0.3042
G12	G13	G23
1.523	1.523	2.569

Lower Bound

Aramid_29/Epoxy 11.61 (Vy: 0.336, PF: 0.750)

Ex	Ey	Ez
13.15	3.545	3.572
vxy	vxz	vyz
0.3765	0.2649	0.2894
Gxy	Gxz	Gyz
1.629	1.543	2.487

FGM

Ex	Ey	Ez
16.36	4.937	4.937
vxy	vxz	vyz
0.3597	0.3597	0.2409
Gxy	Gxz	Gyz
1.98	1.98	1.569

Braid Properties

Young's Modulus Range	13.149 - 18.237
Braid Cross-Sectional Dims.	1.051 X 1.051
Unit Cell Heigth	1.432
Average Braiding Angle	11.613
Interior Braiding Angle	15.4756
Surface Braiding Angle	7.8815
Braiding Tightness	0.3351
Fiber Volume Fraction	0.25758
Number of Yarns	24

Figure 5.2: GUI of the program for creating the geometric model of the braid.

and density are stored. Additionally, for the fibre properties, a default denier can be stored in the library as well, though this can be overridden at any point in the program, see Figure 5.2. For the matrix material properties, only the name of the material, the Young's modulus, and Poisson's ratio are recorded.

These material properties are combined with the yarn packing factor to determine the properties of the impregnated yarns (this is just an individual yarn with the matrix impregnating the fibres). The majority of the material properties are combined using the rule of mixtures, as shown in Equation (5.1) - (5.4).

$$E_{11} = E_{1,f} V_f + E_m V_m \tag{5.1}$$

$$E_{22} = E_{33} = \frac{E_m E_{2,f}}{E_{2,f} V_m + E_m V_f} \tag{5.2}$$

$$G_{12} = G_{13} = \frac{G_m G_f}{G_f V_m + G_m V_f} \tag{5.3}$$

$$\nu_{12} = \nu_{13} = \nu_f V_f + \nu_m V_m \tag{5.4}$$

Where the fibre volume fraction of the individual impregnated yarns, V_f, and matrix volume fraction, V_m, are defined with the yarn packing fraction [7], κ,

$$V_f = \kappa \tag{5.5}$$

$$V_m = 1 - V_f = 1 - \kappa. \tag{5.6}$$

The remaining material properties are derived using the composite cylinder assemblage model [71], as shown in Equation (5.7) - (5.10).

$$G_{23} = \frac{E_{22}}{2\left(1 + \nu_{23}\right)} \tag{5.7}$$

$$\nu_{23} = \frac{K^* - c\, G_{23}}{K^* + c\, G_{23}} \tag{5.8}$$

$$c = 1 + \frac{4 K^* \nu_{13}^2}{E_{11}} \tag{5.9}$$

The bulk modulus, K^*, of the composite is given by

$$K^* = \frac{K_m \left(K_f + G_m\right) V_m + K_f \left(K_m + G_m\right) V_f}{\left(K_f + G_m\right) V_m + \left(K_m + G_m\right) V_f}, \tag{5.10}$$

where the bulk moduli of the fibres, K_f, and matrix, K_m, are given by Equation (5.11) and (5.12), respectively.

$$K_f = \frac{E_{1,f}}{2\left(1+\nu_f\right)\left(1-2\nu_f\right)} \tag{5.11}$$

$$K_m = \frac{E_m}{2\left(1+\nu_m\right)\left(1-2\nu_m\right)} \tag{5.12}$$

Finally, the radius for the cross-section of the yarns is determined from the denier and the linear density of the yarns. Adapted from [28]. Equation (5.13).

$$r = \sqrt{\frac{\zeta}{9000\,\pi\,\rho}} \tag{5.13}$$

$$\zeta := \mathit{Denier}\ ,\ \left[\frac{\mathrm{g}}{9000\,\mathrm{m}}\right]$$

$$\rho := \mathit{Density}\ ,\ \left[\frac{\mathrm{g}}{\mathrm{cm}^3}\right]$$

Braid Definition

Within the program is a braid definition section. This section allows one to select the materials for the yarns and the matrix, as well as make further adjustments to the set-up and layout of the machine and the design of the braided structure. The different features that can be adjusted are the number of knots, the path angle, the width and the depth of the braid, the spacing factor, the number of unit cells, and either the unit cell height or the average braid angle.

The number of knots (points defining the path of the yarns) controls the number of points that are generated between the start and end positions of a cam movement. The number of knots has little to no effect on the on the geometric model, but has more of an effect on the CAD model. The more points that are generated the more knots there will be defining the B-Spline curve used to generate the yarns in the CAD software. The increase in the number of knots also makes the creation of the CAD model take more or less time. The number of knots used in the follow study is kept at a constant ten knots.

The path angle is a property that alters the path that the yarn is assumed to take. It bows the beginning and end of the path allowing for a different representation of the yarn path. For this study, the angle is assumed to be zero, meaning that the yarns have no alteration to the path.

The width and depth of the braid control the number of active cams in the model. For this study, only square braids are considered.

The spacing factor, SF, is an adjustment factor for the space between yarns. The spacing factor can be a number between zero and infinity, though numbers that are approximately equal to one are more realistic, as open structure 3D braids are more likely to collapse when tension is applied. Though the minimum spacing factor can be calculated (see Equation (5.14)), it is noted that for the braids in the study, all the spacing factors are close to one due to the yarn diameter, d, being much smaller than the unit cell height, z. Because of this, the spacing factor is approximated as one throughout this study.

$$SF = \frac{z}{\sqrt{z^2 - 4\,d^2}} \tag{5.14}$$

Finally, either the height of a unit cell or the average braid angle can be set. Because the unit cell height and the average braid angle are fundamentally linked, through Equation (5.17), only one of the parameters needs to be set to determine the other. For the study, the unit cell height is set to 1.432 [mm] as is determined in Section 4.3.1. As a result, the braid angle is automatically set to 11.68°, which is with in the result of $12.19 \pm 1.87°$ provided in Section 4.3.1. With all of the parameters set the program can then start calculating the mechanical properties.

Estimated Properties

Additionally, the program is able to calculate some of the theoretical limits of the elastic properties, as well as provide the results from the well-established FGM model. The theoretical limits are found from taking the results of a unidirectional angle lamina at the average braid angle of the 3D braided composite and from a unidirectional lamina with no angle offset.

The program is able to estimate the internal, external, and average braid angle, as well as the braid pitch. These braid parameters are estimated based on the geometry of the braid and the assumption that for the calculation the yarns follow linear paths. For both the internal and external braid angle, the yarns move by $2\sqrt{2}\,r\,SF$ in the transverse directions, and move by $z/2$ and z in the axial direction, respectively. From this, the braid angle is estimated using the trigonometric relationships. The internal, external, and average braid angle for an n by m braid are derived and shown in Equations (5.15), (5.16), and (5.17), respectively.

$$\theta_{int} = \tan^{-1}\left(\frac{4\sqrt{2}\,r\,SF}{z}\right) \tag{5.15}$$

$$\theta_{ext} = \tan^{-1}\left(\frac{2\sqrt{2}\,r\,SF}{z}\right) \tag{5.16}$$

$$\begin{aligned}\theta_{ave} = &\frac{2\,n\,m - n - m}{2\,n\,m + n + m}\cdot\tan^{-1}\left(\frac{4\sqrt{2}\,r\,SF}{z}\right)\\ &+\frac{2\,(n+m)}{2\,n\,m + n + m}\cdot\tan^{-1}\left(\frac{2\sqrt{2}\,r\,SF}{z}\right)\end{aligned} \tag{5.17}$$

Where z is the braid pitch, defined in Equation (5.18), and SF is the spacing factor, defined previously in Equation (5.14).

$$z = \frac{4\,r\,\sqrt{1+\cos^2(\theta_{int})}}{\sin(\theta_{int})} \tag{5.18}$$

Path Generation

The paths that are created by the machine emulation process consist of straight segments that are connected. The individual yarns positions are stored in a matrix, with the initial matrix shown in Figure 5.3a. In the matrix, the yarns are located by the number 1, the odd cams are assign -1, and the even cams are assigned -2, while unoccupied spaces are assigned a 0. To track the individual yarns as they move, the 1's are replaced by incrementing numbers from 1 to the number of yarns in the braid, shown in Figure 5.3b.

The machine emulation is created by introducing functions that are able perform the rotation of the cams in the matrix. For the rotary 3D braid, there are two functions one that looks for the odd cams, -1, and rotates the surrounding yarns in one direction. Similarly there is another function that looks for the even cams, -2, and rotates the surrounding yarns in the opposite direction. After each step in the braiding process is completed, the new locations of the yarns are saved into a new index in the matrix. This creates a 3D matrix where each layer in the third direction is the locations of the yarns at a specified height.

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & -1 & 1 & -2 & 1 & -1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & -2 & 1 & -1 & 1 & -2 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & -1 & 1 & -2 & 1 & -1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 & 2 & 0 & 3 & 0 \\ 4 & -1 & 5 & -2 & 6 & -1 & 7 \\ 0 & 8 & 0 & 9 & 0 & 10 & 0 \\ 11 & -2 & 12 & -1 & 13 & -2 & 14 \\ 0 & 15 & 0 & 16 & 0 & 17 & 0 \\ 18 & -1 & 19 & -2 & 20 & -1 & 21 \\ 0 & 22 & 0 & 23 & 0 & 24 & 0 \end{bmatrix}$$

(a) (b)

Figure 5.3: Initial (a) and indexed (b) tracking matrices, used for determining the location of yarns during machine emulation.

After the locations of the yarns are determined for each step, the matrix is translated into a set of x, y, z points for each yarn. The number of x, y, z points are then increased by determining the yarn positions between the steps of the braiding process. The number of additional points is specified by the user in the GUI of the program and will produce equally spaced points in the x-direction (along the braid axis).

Due the stepwise nature of the braiding process, the use of machine emulation creates a non-smooth model with bends and cusps. The yarn paths are then processed to create a more continuous and smoother path for the yarns, emulating the undulations found in physical braided samples. The smoothing is performed by a moving average filter, however, due to the phase shift of running a filter in one direction (forwards), a zero-phase filter is used instead.

The benefit of using a zero-phase moving average filter over a simple moving average filter is shown in Figure 5.4. As seen in the figure, the simple forward smoothing process causes a shift of the original path, in the braiding axis direction. This shift causes an inaccuracy with predicting the location of the yarns. The zero-phase filter mitigates this by running the smoothing algorithm both forwards and backwards resulting in a smoothed yarn path that still lines up with the original yarn path.

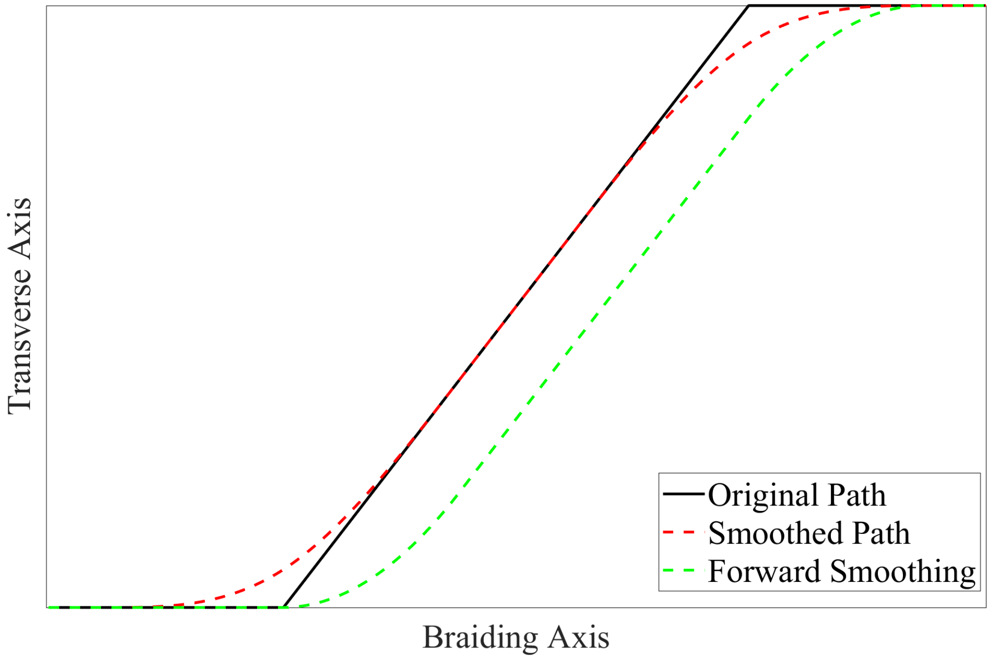


Figure 5.4: Comparison of the affect of forward smoothing (green) and zero phase smoothing (red).

Visualization and Output

Being able to visualize the different properties of a 3D braided composite is important to understand if the structure is suitable for an application. The program offers a number of different visualizations and outputs to help the user better understand the structure of the 3D braided composite. The main outputs from the program are in the form of tabled data displayed in the GUI, Graphs to help interpret the braided structure, and an output file that summarizes the braids paths that are created.

The first visual output that the program offers is the reference on a yarn jamming plot, see Figure 5.5. In this output, the current parameters for the fibre volume fraction and braid angle are plotted with respect to the yarn jamming line. This provides an estimate for the maximum braid angle and maximum fibre volume fraction of the current structure. For tailoring the properties, this provides a reference for how much the braid angle and fibre volume fraction are allowed to change before a change to the structure is required.

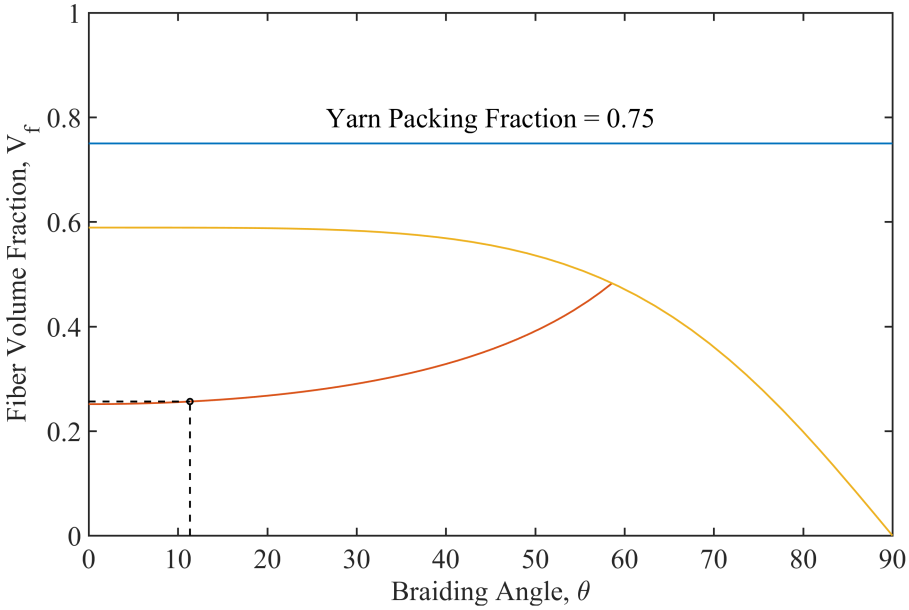


Figure 5.5: Plot of fibre volume fraction as a function of the braid angle (red) with respect to the yarn jamming (yellow)

To provide a visual reference of how the geometry of the braid will look, the program creates an approximate plot of the 3D surfaces that will make the braid, shown in Figure 5.6. This is performed by plotting a 3D tube along the path of the yarns. The visualization of the yarns allows for the detection of yarn interference or of the yarns being to far apart, allowing parameters, such as the spacing factor, to be adjusted. By providing this visualization in the program, the 3D structure can be confirmed and time is saved generating the geometric CAD model.

Using the plots of the yarn paths with a yarn highlighted, shown in Figure 5.7, provides insight to how the yarns will move in the braid. Further, these plots can be viewed from the top to determine how many different loops the yarns will travel in. For the three by three braided structure presented in this work, there are three different paths that the yarns will traverse. These paths are shown in Figure 5.8, where it should be noted that once a yarn starts on one of these three paths it will remain there as long as the braiding pattern remains unchanged.

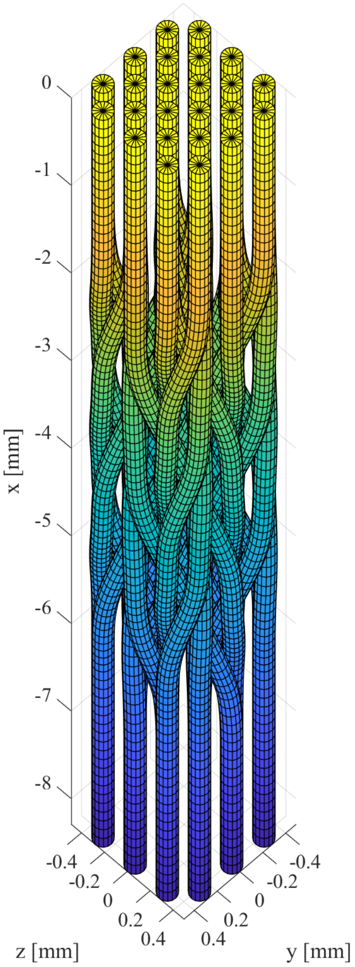


Figure 5.6: Three-dimensional visual representation of the yarn paths.

The final output from the program is an output file. This output file makes sure that the settings and parameters can be found later, either to help with the recreation of a specific braided structure or as a simple log/summary. The output file is saved as plaintext in the properties file, `BRAID.PROPERTIES`, and an example of the output is included in Appendix C.3.

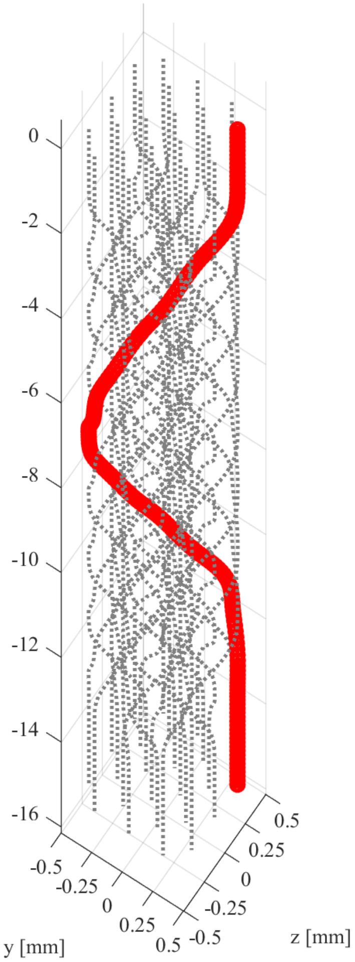
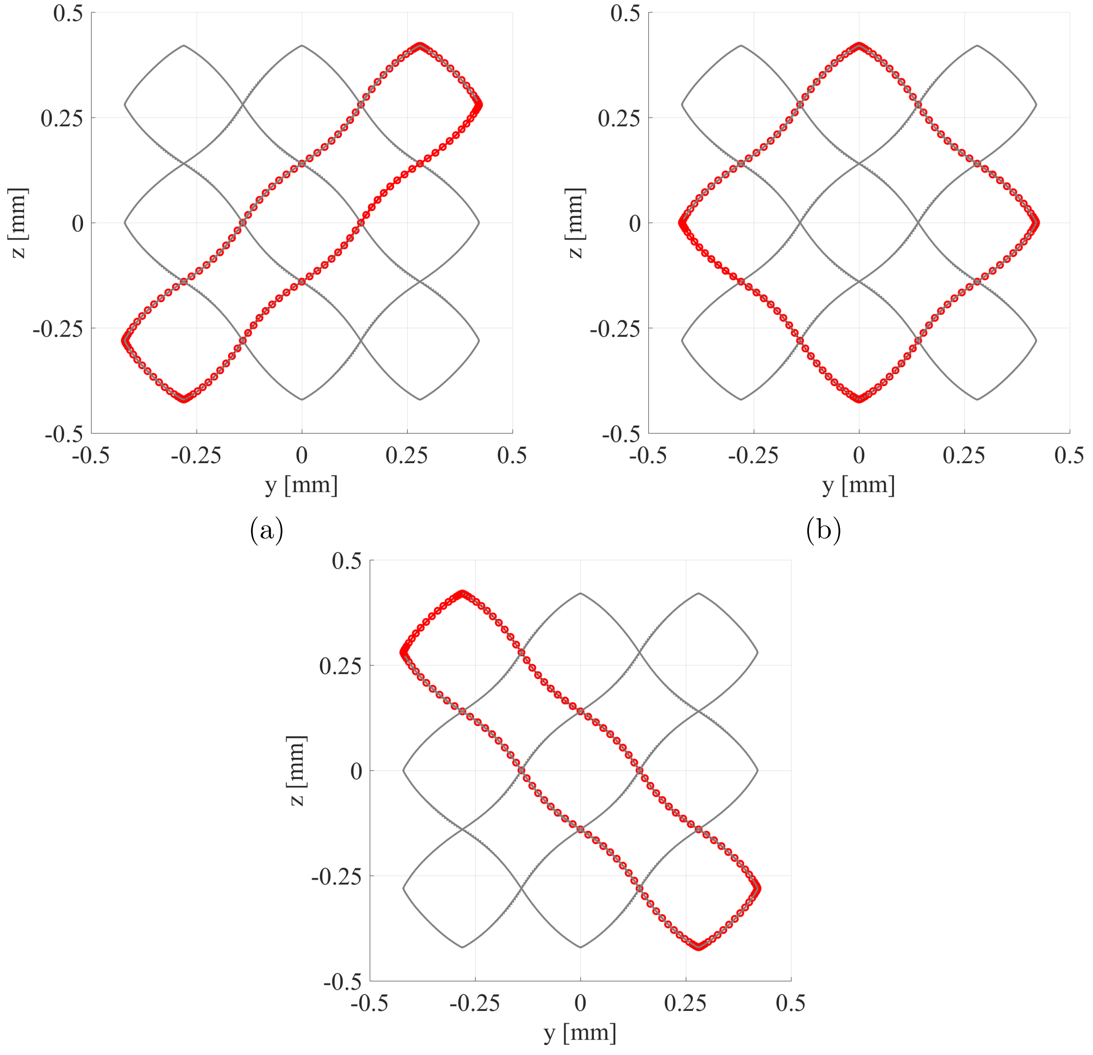


Figure 5.7: Three-dimensional visual representation of the yarn paths with a single path highlighted for clarity.

CAD Export

The program can be used to automatically export the yarn paths to create a 3D geometric CAD model. The program initiates the creation of the CAD model by invoking a system command. The system command is set to open the CAD software and run the macro written in Visual Basic for Applications [72] (VBA) to create the CAD model. The follow section will provide a more in depth description of this



(a)

(b)

(c)

Figure 5.8: Figure showing the two-dimensional paths that the yarns follow during the braiding process.

macro, and the creation of the geometric CAD model.

5.3 Geometric CAD Model Generation

The geometric CAD model, shown in Figure 5.9, serves as a visual 3D model of the braided structure that can be easily manipulated and inspected. The CAD model also provides an intermediate step between the geometric model discussed previously, and the finite element model discussed in Chapter 6. Further, the geometric CAD model provides a method for confirming the geometry of the final braid.

The three-dimensional representation of the braided structure is generated using in-house created software, available in Appendix D. The software utilises Visual Basic for Applications (VBA) to autonomously generate the model based on the individual data for the yarns, and the general braid data file. The program is able to read in the data about the spacing, as well as the data about the yarn diameter, number of yarns, and dimensions of the braid. After the braid data is read in, the x, y, z data for each of the yarns is read in and a spline is created as a path for extruding the yarn. A cylindrical yarn is extruded by extruding a circle, with the same diameter as the yarns, along the spline path. After all the yarns are created, a solid block is created for the resin and the yarns are then cavated from the resin creating an assemblage of parts. The full assembly, as well as the individual constituents (yarns and matrix) are shown in Figure 5.10.

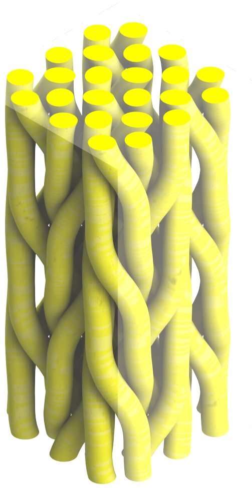


Figure 5.9: Three-dimensional CAD model of the braided structure.

Additionally, the geometric CAD model can be compared directly with the braids produced from a 3D braiding machine. Figure 5.11 shows a braid produced on the 3D rotary braided presented in Chapter 4 next to the geometric CAD model of a braid with the same properties. It can be seen that the braids share a similar structure and that the paths of the yarns in the geometric model follow closely to those present in the braided sample, however, it is noted that the geometry is not exact due to the assumption that the yarns have a circular cross-sectional area. The development of a model that is able to further take in account the changing cross-sectional area of the yarns may be addressed in the future work following this study.

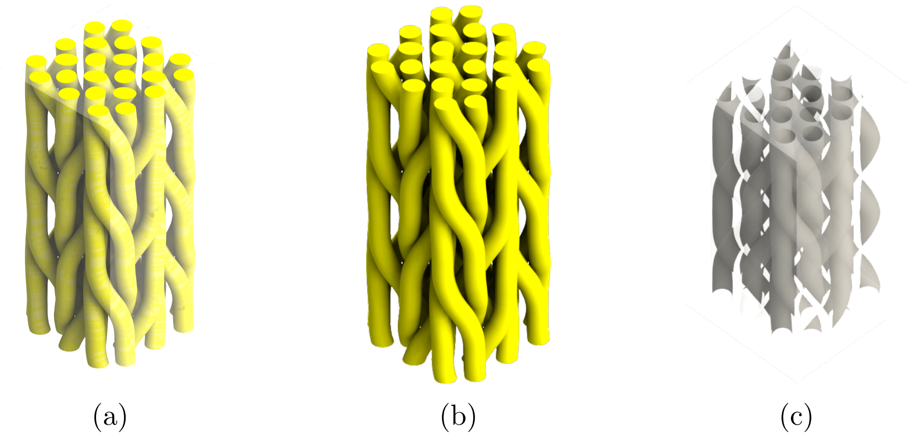
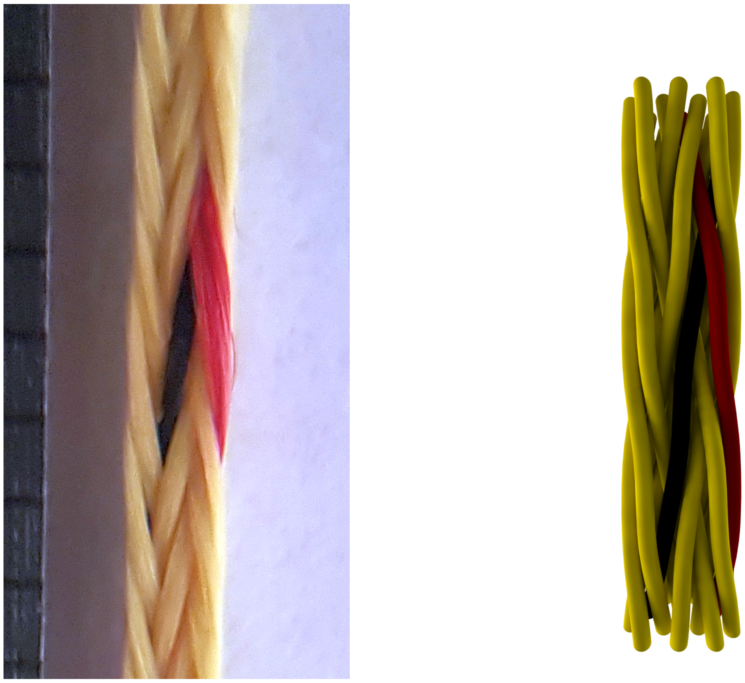

(a) (b) (c)

Figure 5.10: Constituents of the CAD model, showing (a) the combined model, (b) the fibres, and (c) the matrix.

Finally, the integration of the two programs allows the automatic creation of the geometric model, as well as the automatic creation of the geometric CAD model. This reduces the time required to create the necessary models for finite element modelling that is discussed in Chapter 6. And further, facilitates the means to simulate a tensile test to determine the elastic properties of the braided structure.

5.4 Conclusions

In this chapter, the procedure of producing a geometric, as well as a geometric CAD model are introduced. The geometric model is produced using the machine emulation method. The machine emulation is performed by an in-house computer program. Furthermore, this computer program is able to provide the user with valuable feedback regarding the properties and the structure of the 3D braids. Lastly, the computer program is able to export the geometric model to a second in-house program that is capable of producing a geometric CAD model.



(a) (b)

Figure 5.11: Comparison of (a) the produced braid and (b) the rendered braid

The two programs are able to work together to automate the creation of geometric models. And the results of the models are shown to have the same structure as the braids produced from the in-house rotary 3D braider. In the follow chapter, the geometric CAD model is used to generate a finite element model to determine the elastic properties of the braided structure.

References

[7] F. Ko, “Tensile strength and modulus of a three-dimensional braid composite,” in *Composite Materials (Seventh Conference)*, 1984, pp. 392 –403.

[11] L. Chen, X. Tao, and C. Choy, “Mechanical analysis of 3-d braided composites by the finite multiphase element method,” *Composites Science and Technology*, vol. 59, no. 16, pp. 2383–2391, 1999. DOI: 10.1016/s0266-3538(99)00087-1.

[13] N. Tolosana, S. Lomov, and A. Miravete, “Development of a geometrical model for a 3d braiding unit call based on machine emulation,” 2007.

[16] H. Y. Yang, X. G. Zhou, J. S. Yu, and Z. Luo, “Entity simulation of rectangular preform for 3d braided composite based on ansys,” *Advanced Materials Research*, vol. 800, pp. 336 –340, 2013, ISSN: 1662-8985. DOI: 10.4028/www.scientific.net/amr.800.336.

[22] M. F. Smith, “Apparatus and method for automated braiding of square rope and rope product produced thereby,” U.S. Patent US4803909A, Feb. 1989.

[28] Y. T. Gao, F. K. Ko, and H. Hu, “Integrated design for manufacturing of braided preforms for advanced composites part ii: 3d braiding.,” *Applied Composite Materials*, vol. 20, no. 6, pp. 1065 –1075, 2013, ISSN: 0929189X. DOI: 10.1007/s10443-012-9304-5.

[32] T. Kostar and T.-W. Chou, “A methodology for cartesian braiding of three-dimensional shapes and special structures,” *Journal of Materials Science*, vol. 37, pp. 2811 –2814, 2002.

[47] C. Deng, J. J. Jiang, and L. C. Fang, “Algorithm design of four-step three-dimensional braided composite structures in matlab environment,” in *Applied Mechanics and Materials*, ser. Machine design and manufacturing engineering; (ICMDME 2013), vol. 365, Trans Tech Publications, 2013, pp. 1144 –1147. DOI: 10.4028/www.scientific.net/amm.365-366.1144.

[71] Z. Hashin, “Theory of fiber reinforced materials,” Pennsylvania University, Tech. Rep., Mar. 1972.

[72] Microsoft, *Visual basic for applications*, Software, 2016.

Chapter 6

Finite Element Modelling of the Tensile Properties for a Three-Dimensional Rotary Braid

6.1 Introduction

This chapter addresses the need for a method and model that is able to predict the properties of three-dimensional braided composites. The model developed uses the concept of sub-unit cells first introduced by Chen [11] to predict the properties of a braid formed by a 3D rotary braider. A design of experiments is implemented to study the results and compare results to a well know modelling method.

6.2 Model Development

The imported geometry from the CAD model (Figure 6.1) is set up in the commercially available finite element analysis (FEA) software ABAQUS [73] to allow further analysis to be performed. In this section, a focus on the response to iso-strain conditions are analysed to determine the mechanical properties of the full and sub-unit cell models. Further, the results of a new model for predicting three-dimension (3D) braided composites are investigated.

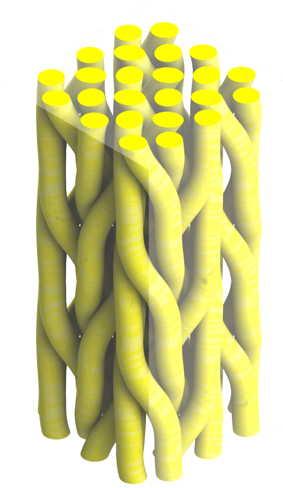


Figure 6.1: Imported CAD model of the three full unit cells for a 3 by 3 braided structure.

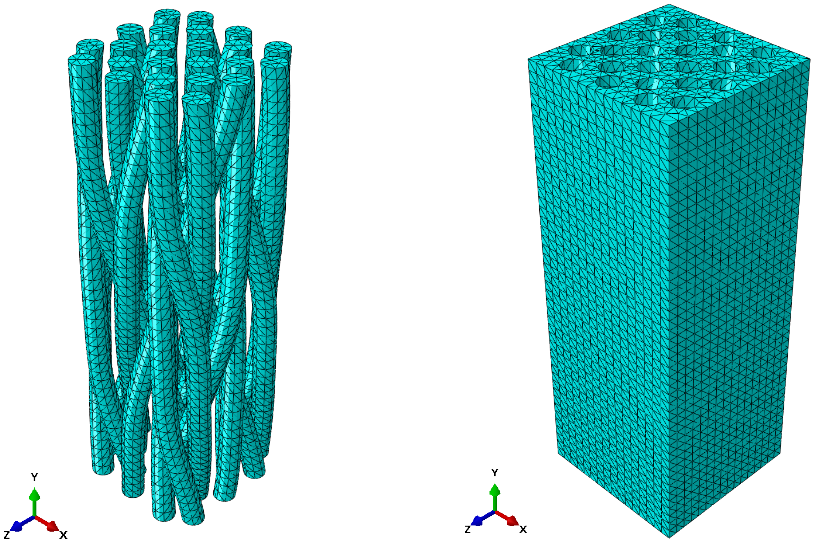
6.2.1 Unit Cell

Unit cells are the smallest repeating element that is representative of the entire structure. Researchers use unit cells to simplify the complex repeating geometry into a finite structure. Because of this, they have been utilized in many developments to understand composite materials [38–40]. In this research, the development of a full-unit cell is required to validate the properties of a proposed sub-unit cell model.

The full-unit cell is analysed at three different levels: one unit cell, two unit cells, and three unit cells in height. This is done to show that the results at the midplane converged and can be considered representative of the full braided structure. The results show that the results are converged, and the there is no significant effect on the remaining results of the study.

All models are consistently meshed at three levels: an initial course mesh, an intermediate mesh size chosen to roughly double the number of elements in the model, and, finally, a fine mesh chosen to roughly quadruple the number of element from the initial mesh. Examples of the meshes for the yarns and matrix are shown in Figure 6.2.

The load that is applied to the model causes an iso-strain condition. This allows for simpler determination of the mechanical properties. By straining the specimen a known amount and measuring the total forces in the midplane, the effective stiffness for the sub-unit cells or the Young's modulus of the full-unit cell can be determined.



(a)

(b)

Figure 6.2: Mesh applied to the yarns (a) and the matrix (b) of the imported geometry.

To recreate these conditions in the FE model, the bottom face of the model is restricted to remain on its plane while the top face is displaced upwards to cause a 1% strain in the model. To prevent large movements and to stabilize the models, a single node is *encastré'd*. All of the applied boundary conditions are seen applied to the 3 by 3 full-unit cell model in Figure 6.3. For the 3 by 3 full-unit cell model, the model is run to determine the global results, for subsequent sub-unit cells.

After the FEA has been run, the results are analysed. For this the deformed model, shown in Figure 6.4, is bisected on its midplane. From the new surface that is created the total force is measured, see Figure 6.5. The total force is then converted back into a engineering stress by dividing the force by the original cross-sectional area of the model. With the mid-plane engineering stress determined, the Young's modulus or resulting stiffness is determined by dividing the engineering stress by the known strain that is applied to the model.

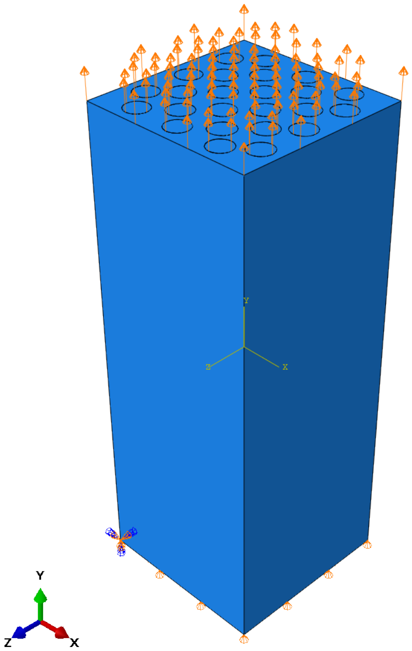
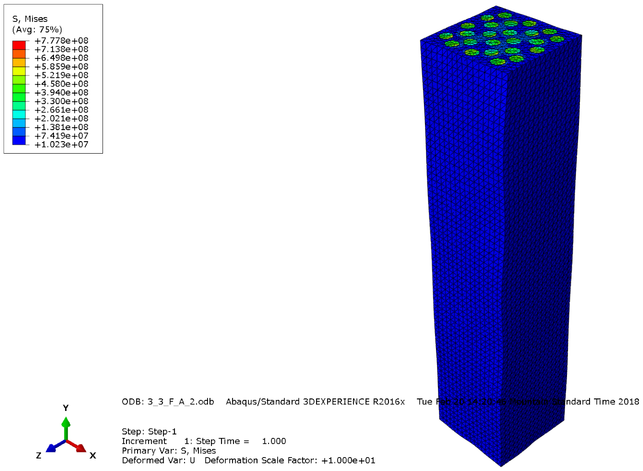


Figure 6.3: Load applied to the FEA model.



```
ODB: 3_3_F_A_2.odb    Abaqus/Standard 3DEXPERIENCE R2016x    Tue Feb 20 14:20:46 Mountain Standard Time 2018

Step: Step-1
Increment     1: Step Time =    1.000
Primary Var: S, Mises
Deformed Var: U   Deformation Scale Factor: +1.000e+01
```

Figure 6.4: The scaled deformation of the FEA model under the applied loads.

6.2.2 Cyclic Unit Cell

For square and rectangular braided structures, there is a smaller unit cell that exists and is due to the rotational symmetry (cyclic unit cell). Cyclic unit cells can be applied to determine the properties of the braided structure. The combination of using a cyclic unit cell with FE analysis allows for the drastic decrease in the time required to solve the model. However, for larger and larger braids the benefit of this

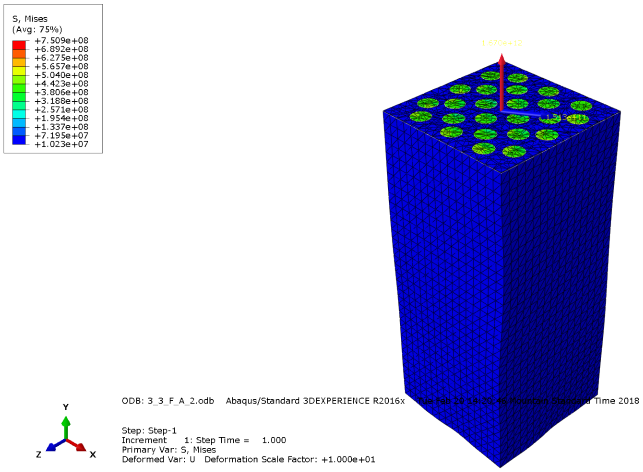

S, Mises
(Avg: 75%)
+7.509e+08
+6.892e+08
+6.275e+08
+5.657e+08
+5.040e+08
+4.423e+08
+3.806e+08
+3.188e+08
+2.571e+08
+1.954e+08
+1.337e+08
+7.195e+07
+1.023e+07
Y
Z
X
ODB: 3_3_F_A_2.odb Abaqus/Standard 3DEXPERIENCE R2016x Tue Feb 20 14:20:46 Mountain Standard Time 2018
Step: Step-1
Increment 1: Step Time = 1.000
Primary Var: S, Mises
Deformed Var: U Deformation Scale Factor: +1.000e+01

Figure 6.5: The resulting force on the model at the midpoint.

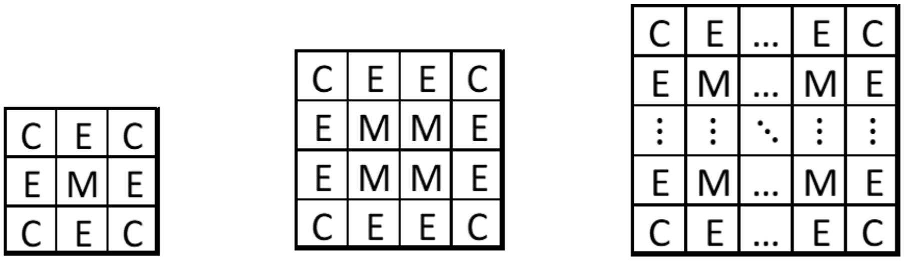
is reduced, and the time required to solve the model drastically increases.

6.2.3 Sub-Unit Cell

Sub-unit cells are the next extension of unit cells. Sub-unit cells are a sub division of the full-unit cell that can be patterned and combined to form the full-unit cell geometry. They were first noted by researchers developing full-unit cell models, however, their effects were not considered for predicting the properties of the braided structures [7]. Later, the properties of the individual sub-unit cells for track and column braiders were analysed and it was noted that the properties differ for each of the sub- unit cells [11]. For this section, the sub unit cells are defined and the FEA modelling method is shown.

The sub-unit cell model utilises multiple repeating sections that can be patterned to form the full braid cross-section. The sub-unit cell model consists of three different sub-unit cells: edge-, middle-, and corner- unit cells. Figure 6.6 shows how the sub-unit cells can be patterned and combined to form the full unit cell.

From these three types of sub-unit cells, the material properties of various, and more arbitrary shapes can be estimated. By utilising the stiffness from each of the individual sub-unit cells, shown in Figure 6.7, they can be combined to estimate the effective Young's modulus of the material in the axial, as well as the transverse directions.



C	E	C
E	M	E
C	E	C

(a) 3 by 3

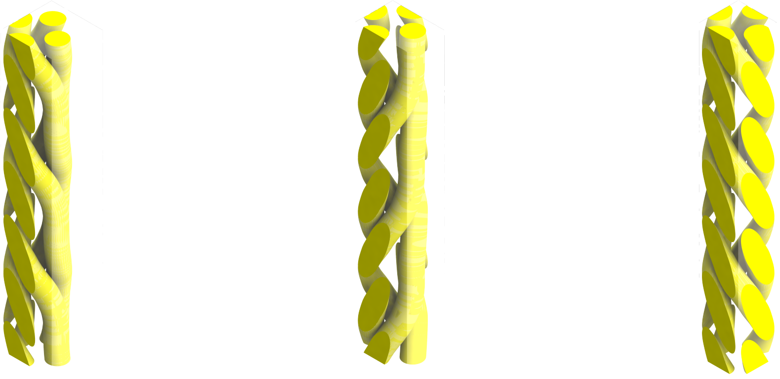
C	E	E	C
E	M	M	E
E	M	M	E
C	E	E	C

(b) 4 by 4

C	E	...	E	C
E	M	...	M	E
⋮	⋮	⋱	⋮	⋮
E	M	...	M	E
C	E	...	E	C

(c) n by m

Figure 6.6: Pattern for sub-unit cells.



(a)

(b)

(c)

Figure 6.7: Sub-unit cells for a rectangular braided structure. Including (a) the centre sub-unit cell, (b) the edge sub-unit cell, and (c) the middle sub-unit cell.

To determine the stiffness of each of the individual sub-unit cells, a full-unit cell model is tested using FEA, then the model is sliced and global-local FEA is performed. Also known as sub-modelling, global-local FEA allows the determination of the stiffness of the sub-unit cells while accounting for the interactions between adjacent sub-unit cells.

The sub-unit cell models are adapted from the full-unit cell model. The full-unit cell model is cut into the sub-unit cells, with the new sub-unit cell models enabled for running global-local (sub-modelling) analysis. Once the sub-unit cell model is linked to the original full-unit cell model, a new sub-modelling boundary condition is applied to the cut faces that further links the results of the models.

Then in a similar process to the full-unit cell model, the sub-unit cell models are meshed and the analysis is run. The results for the stiffness are also obtained in a similar way; by slicing the model on its midplane and measuring the total force on the cut face, the stiffness of the sub-unit cells are determined. These stiffnesses are then utilized for the new method discussed for determining the mechanical properties of the larger braided structures.

6.2.4 Prediction of the Properties of Large versus Small Cross-Section Braids

To predict the mechanical properties of large cross-sectional braided structures, a new method is introduced. The properties of the sub-unit cells from a 3 by 3 braid are combined to predict the properties of larger braided structures. To combine the mechanical properties, the sub-unit cells are assumed to act as springs in parallel and series.

To be able to treat the sub-unit cells as a set of springs the mechanical properties have to be converted to an analogous form. The method utilized in finite element analysis (FEA) to determine the stiffness of the elements is introduced. This treats each individual spring stiffness, k_i, of the system as a function of the mechanical stiffness, E_i, the cross-sectional area, A_i, and the length of each member, L_i. The individual spring stiffness can be seen in Equation (6.1), and just like physical springs the methods for combining the sub-unit cell stiffness for parallel and for series arrangements are shown in Equations (6.2) and (6.3), respectively.

$$k_i = \frac{E_i\,A_i}{L_i} \tag{6.1}$$

$$k_{eff} = \sum_i k_i \tag{6.2}$$

$$k_{eff} = \left[\sum_i \left(\frac{1}{k_i}\right)\right]^{-1} \tag{6.3}$$

Axial Young's Modulus

The Axial Young's modulus is found by combining the stiffnesses of the individual sub-unit cells as if they are a system of parallel springs, as shown in Figure 6.8.

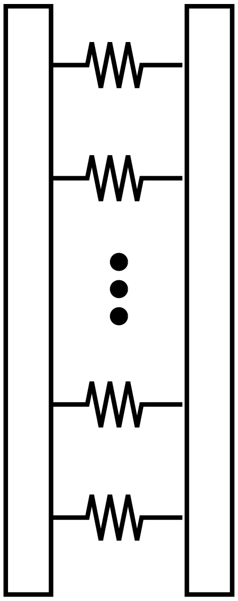


Figure 6.8: Representation of the spring model used to combine the sub-unit cells to determine the axial Young's modulus. Each spring represents a single sub-unit cell.

$$\frac{E_{comp,\,x} A_{comp,\,yz}}{L_{comp,\,x}} = \sum_i \frac{E_i\, A_i}{L_i} \tag{6.4}$$

$$E_{comp,\,x} = \frac{L_{comp,\,x}}{A_{comp,\,yz}} \times \sum_i \frac{E_i\, A_i}{L_i} \tag{6.5}$$

Because each of the sub-unit cells and the composite material as a whole has the same length in the x-direction, this can be reduced to:

$$E_{comp,\,x} = \frac{1}{A_{comp,\,x}} \times \sum_i E_i\, A_i \tag{6.6}$$

This can then be written as

$$E_{comp,\,x} = \frac{n_c\, E_c\, A_c + n_m\, E_m\, A_m + n_e\, E_e\, A_e}{A_{comp,\,yz}} \tag{6.7}$$

for an n by m rectangular 3D braid:

$$\begin{aligned} n_c &:= 4 \\ n_m &:= n\,m - 2m - 2n + 4 \\ n_e &:= 2m + 2n - 8 \end{aligned}$$

Transverse Young's Modulus

The transverse Young's modulus is found similarly to how the axial Young's modulus is found. However, due to the more complex layout of the sub-unit cell with respect to the transverse direction, a different algorithm needs to be implemented to determine the Young's modulus. The new layout involves combining the sub-unit cells together as a system of parallel springs similar to the axial, however, each of the individual parallel springs are comprised of a set of springs in series. A visual representation of the new spring model for determining the effective transverse modulus is shown in Figure 6.9 where each of the springs represents a single sub-unit cell.

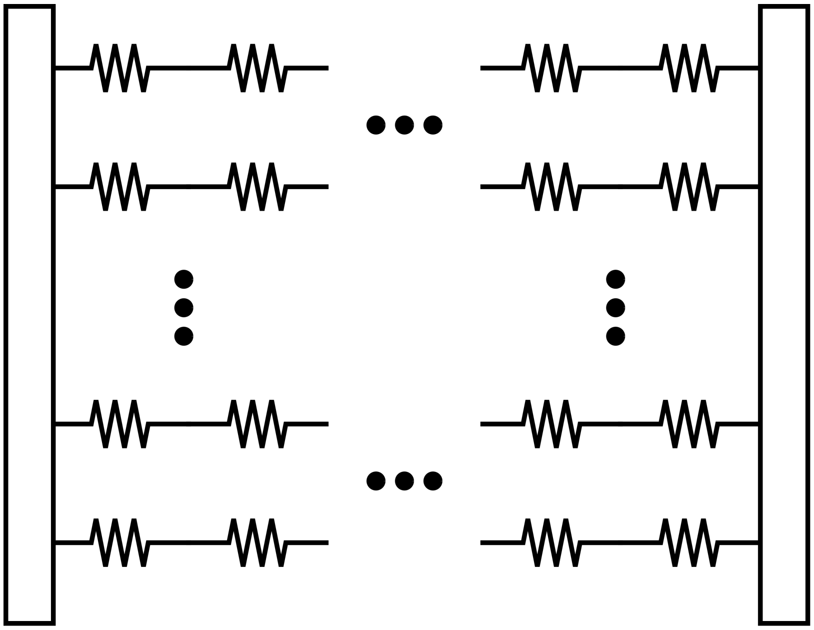


Figure 6.9: Representation of the spring model used to combine the sub-unit cells to determine the transverse Young's modulus. Each spring represents a single sub-unit cell.

To determine the effective Young's modulus the springs must be combined into a single spring that represents the system. To effectively combine the springs two steps are required. The first step involves determining the effective stiffness of each of the sets of springs in series. Next each of these stiffnesses are combined using the previously established methods to determine the overall effective transverse Young's modulus; both steps are summarized by Equation (6.8).

$$\frac{E_{comp,\,y}\,A_{comp,\,xz}}{L_{comp,\,y}} = \sum_i \left[\sum_j \frac{L_{i\,j}}{E_{i\,j}\,A_{i\,j}} \right]^{-1} \tag{6.8}$$

Note that for each of the strips in the j direction have the same cross-sectional area and thus the equation for the effective Young's modulus of the composite can be written as:

$$E_{comp,\,y} = \frac{L_{comp,\,y}}{A_{comp,\,y}} \times \sum_i A_i \left[\sum_j \frac{L_{i\,j}}{E_{i\,j}} \right]^{-1} \tag{6.9}$$

For a rectangular braid of size n by m this can be written as:

$$\begin{aligned} E_{comp,\,y} = \frac{L_{comp,\,y}}{A_{comp,\,xz}} \times & \left[2A_1 \left(\frac{E_c\,E_{e2}}{2E_{e2}L_c + (n-2)E_cL_{e2}} \right) \right. \\ & \left. +(m-2)A_2 \left(\frac{E_{e1}\,E_m}{2E_mL_{e1} + (n-2)E_{e1}L_m} \right) \right] \end{aligned} \tag{6.10}$$

6.3 Design of Experiments

For analysis of the FE results, a design of experiments (DOE) approach is utilized and analysed using commercially available statistics software, Minitab [74]. The analysis is broken into two sections; the first section is for determining the axial results, while the second section is for determining the transverse results.

For these tests, several factors are manipulated; these factors allow for the analysis of the sub-unit and full-unit cells, as well as the analysis of the effects of mesh refinement, and the effect of multiple unit cells on the measurement of the mechanical properties. The manipulated variables in both the axial case and the transverse case are the same, and they are summarized for the axial and transverse case, in Table 6.1 and Table 6.2, respectively. Four different models are tested at different number of stacked unit cells, and different levels of mesh refinement. Furthermore, the 3 by 3 models also have their sub-unit cells analysed. For the axial case as well as the transverse case, the only responding variable is the measured force on the cross-section.

Table 6.1: Design of experiments factors used for analysis of the axial modulus.

	Manipulated Variable	**Units**	**Levels**				
			-2	**-1**	**0**	**1**	**2**
1	Model	-	3 by 3	4 by 4	-	5 by 5	6 by 6
2	Number of Unit Cells	-	-	1	2	3	-
3	Type of Unit Cell	-	Full	Middle	-	Corner	Edge
4	Refinement	-	-	6.5	10	20	-

Table 6.2: Design of experiments factors used for analysis of the transverse modulus.

	Manipulated Variable	**Units**	**Levels**				
			-2	**-1**	**0**	**1**	**2**
1	Model	-	3 by 3	4 by 4	-	5 by 5	6 by 6
2	Number of Unit Cells	-	-	1	2	3	-
3	Type of Unit Cell	-	Full	Middle	Corner	Edge1	Edge2
4	Refinement	-	-	6.5	10	20	-

To analyse the data, multiple regression analysis is implemented in the form of ANOVA testing. To validate the use of ANOVA, the three assumptions that ANOVA makes must be held true. The three assumptions for ANOVA are that the each case or test must be independent of any other test, the residuals must be normally distributed, and the variances are equal. Because of the nature of the tests, all the runs are separate from each other, satisfying the assumption that the tests must be independent. A linear line trend in the half normal plot of the residuals shows that residuals from the analysis are normally distributed, which fulfils the second assumption of ANOVA testing. Finally, the plot of the residuals versus the order number must show no order, trend, or pattern leading to the conclusion that the variances are equal and that the final assumption for ANOVA is fulfilled. Because these assumptions for ANOVA are met, further analysis using ANOVA can be performed.

6.4 Results and Discussion

6.4.1 Axial Young's Modulus

It can be seen from the results in Table 6.3 that each of the sub-unit cells has differing stiffnesses. These differences can be attributed to the difference in the angle of the fibres within the sub-unit cells, as well as the fibre volume fraction of each of the sub-unit cells. Because of the difference in the angle of the exterior and interior yarns, there is a difference in the stiffness of the sub-unit cells.

Table 6.3: Average stiffness of the individual sub-unit cells.

Sub-Unit Cell	**Stiffness [GPa]**
Edge	15.791
Middle	16.121
Corner	14.791

The difference in the angle of the external and internal yarns comes from the movement of the machine. Yarns that currently reside on the internal structure of the braid, are moved on every even or odd movement of the cams. Whereas, yarns that reside on the surface of the braid are only moved every even movement or on every odd movement. This means that yarns on the surfaces are moved half as much as the internal yarns and as a result they have a lower braid angle as well. This lower braid angle increases the mechanical properties in the axial direction.

The fibre volume fraction of each of the sub-unit cells also has an effect on the stiffness of the sub-unit cells. The lower the fibre volume fraction is the lower the stiffness of the sub-unit cell. This occurs due to the increase in the amount of the resin, which has a stiffness that is orders of magnitude lower than that of the fibres, when compared to the fibres.

It is shown, in Figure 6.10, that the data from the tests follows closely to a linear graph. This shows that the data collected is normally distributed, and that the assumption required to run multiple regression is met.

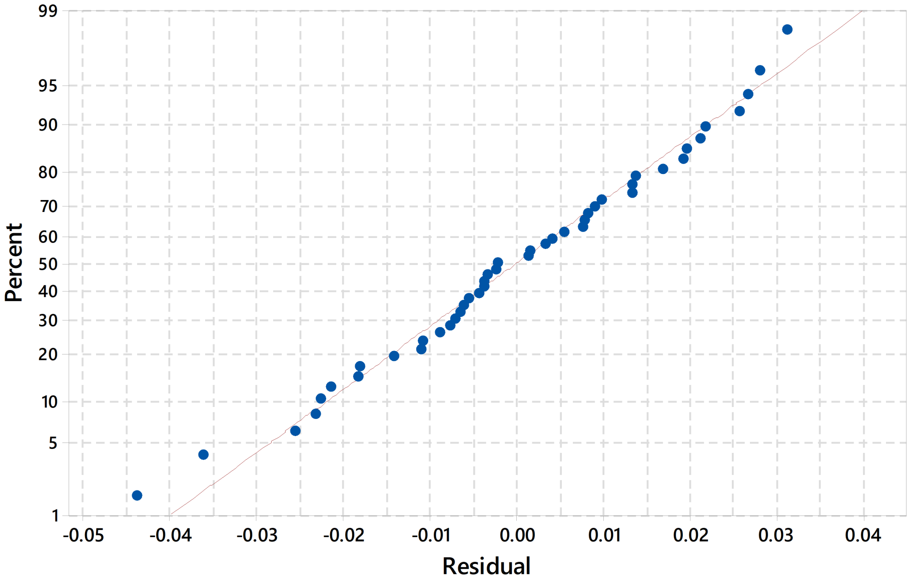


Figure 6.10: Half normal plot of the residuals for the axial Young's modulus.

From the data, multiple regression analysis is performed and the measure of effects for the axial simulations is analysed; this data is shown using the pareto chart from Figure 6.11. From the pareto chart, it can be seen that the mesh refinement has no significant effect on the results. This means that the analysis is independent of the size of the mesh, and the mesh did not significantly effect the results obtained. Further, from the pareto chart, it can be seen that the 'Model' and 'Type of Unit Cell' has a significant effect on the axial modulus Where, the 'Model' refers to the size in terms of number of active cams, *i.e.*, the cross-sectional area. While, 'Type of Unit Cell' refers to the cross-section in terms of being a full-, middle-, edge-, or corner-unit cell.

Looking specifically at the result of the 'Model' on the axial modulus, it can be seen that as the size of the braid increase the median of the modulus increase in a logarithmic fashion. This is shown in the box plot, see Figure 6.12, of the 'Model' versus the axial modulus. However, in the box plot it can be seen that minimum and maximum values for the three by three braid encompass the values achieved for the other braid sizes. This is due to the inclusion of the sub-unit cell stiffnesses in the results.

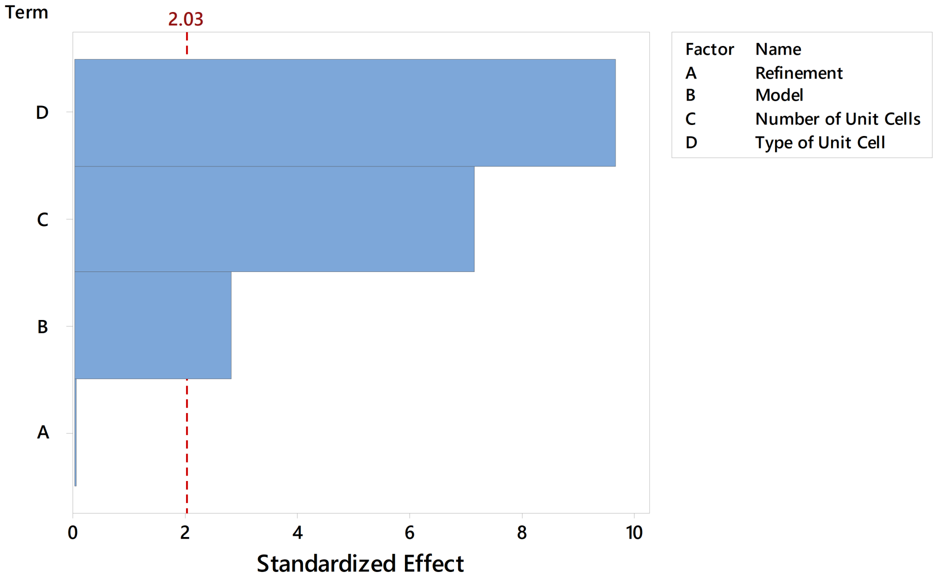


Figure 6.11: Pareto chart showing the significant effects on the axial Young's modulus.

With the sub-unit cell results separated, as shown in Figure 6.13, the trend of the of the axial modulus increasing logarithmically can be clearly seen. As the size of the braid increases, the modulus approaches the stiffness of the middle sub-unit cell. This is expected due to the different rates at which the sub-unit cells grow as the size of the braid increases.

For square braids the number of middle sub-unit cells and edge sub-unit cells are

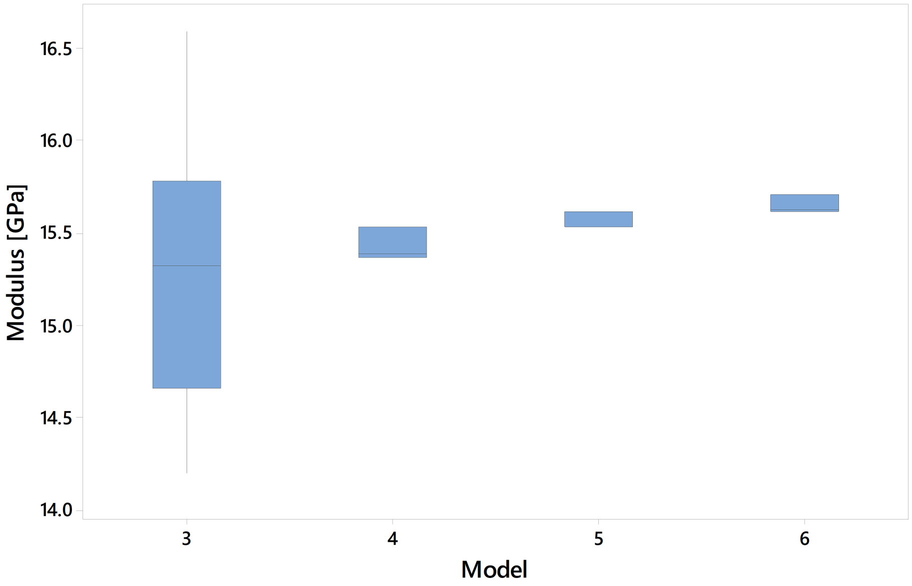


Figure 6.12: Results from the DoE summarized in a boxplot of the axial modulus vs. the number of active cams in the model.

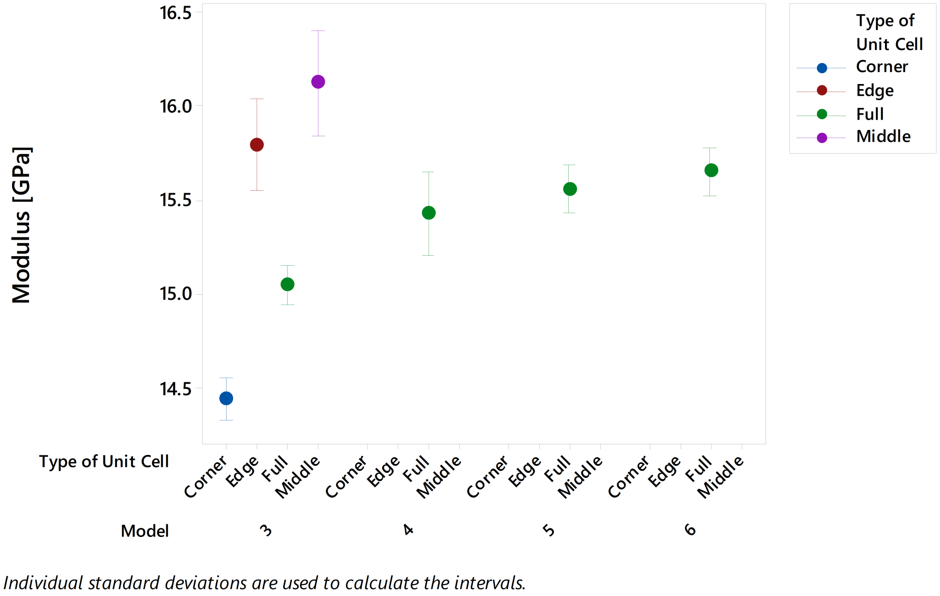


Figure 6.13: Axial results from the DoE, splitting the number of active cams further by the type of sub-unit cell.

given by:

$$n_m = n^2 - 4n + 4 \tag{6.11}$$

$$n_e = 4n - 8 \tag{6.12}$$

$$\tag{6.13}$$

Where it can be shown that the rate at which the middle sub-unit cells grow compared to rate at which the edge sub-unit cells grow is always increasing:

$$dR = d\left(\frac{\frac{d\left(n_m\right)}{dn}}{\frac{d\left(n_m\right)}{dn}}\right)/dn = \frac{d\left((2n-4)/4\right)}{dn} = \frac{1}{2} \tag{6.14}$$

As shown in Figure 6.14, because of the always increasing nature of the number of middle sub-unit cells compared to number of edge sub-unit cells, the effective modulus of larger braided structures can be seen to approach the modulus of the middle sub-unit cell. This shows how for larger braided structures the impact of the other sub-unit cells does not play a significant effect. For larger braided structures, it is sufficient to only consider the effects of the middle sub-unit cell as the repeating volume element. However, for smaller braided structures it is important to consider the effects of the corner and edge sub-unit cells.

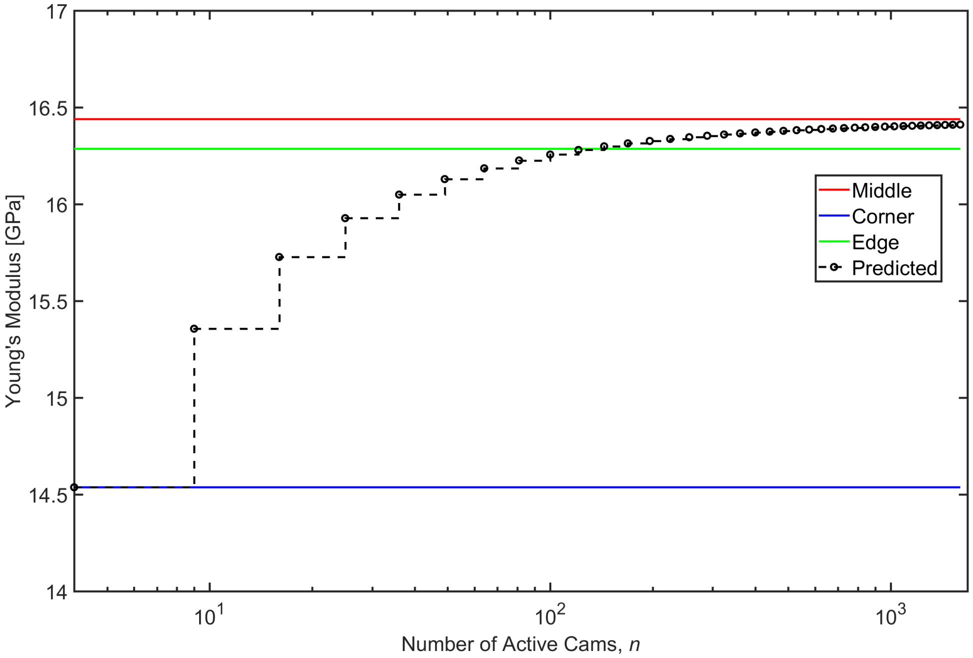


Figure 6.14: Predicted Young's modulus plotted against the number of active cams with the modulus for the middle, edge, and corner sub-unit cells included for reference.

For this study, the predicted axial Young's modulus is compared to that of the simulated one. The predicted axial Young's modulus is found to be within 1.57%, 2.34%, 2.57%, and 2.78% for the 3 by 3, 4 by 4, 5 by 5, and 6 by 6 braided composites, respectively. Whereas, when compared the FGM model for predicting the Young's modulus, the modulus is found to be larger than 8%. These results are summarized in Table 6.4. In Table 6.4, the difference between the predicted modulus and the FEA modulus, as well as the difference between the FGM modulus and the FEA modulus are shown to the right of their respective columns. Both models appear to wander as the size of the braid increases, meaning that the precision in the model degrades.

Table 6.4: Comparison of the axial modulus predicted by the model presented to the full-unit cell results and to the well established FGM model.

Model	**FE Modulus [GPa]**	**Predicted Modulus [GPa]**	**Predicted Difference**	**FGM [GPa]**	**FGM Difference**
3 by 3	15.119	15.356	1.57%	16.363	8.22%
4 by 4	15.367	15.727	2.34%	16.757	9.05%
5 by 5	15.528	15.928	2.57%	16.997	9.46%
6 by 6	15.616	16.049	2.78%	17.159	9.88%

6.4.2 Transverse Young's Modulus

Similar to the results from the axial modulus in Figure 6.15 the data from the tests follows closely to a linear graph. This shows that the data collected is normally distributed, and that the assumption required to run multiple regression is met.

From the data, multiple regression analysis is performed and the measure of effects for the transverse simulations is analysed using the pareto chart in Figure 6.16. From the pareto chart, it can be seen that the mesh refinement has no significant effect on the results. This means that just as in the axial case, the analysis is independent of the size of the mesh.

Further, similar results to the axial case are found. The modulus increased in a logarithmic trend that approaches the value of the middle sub-unit cell. These results

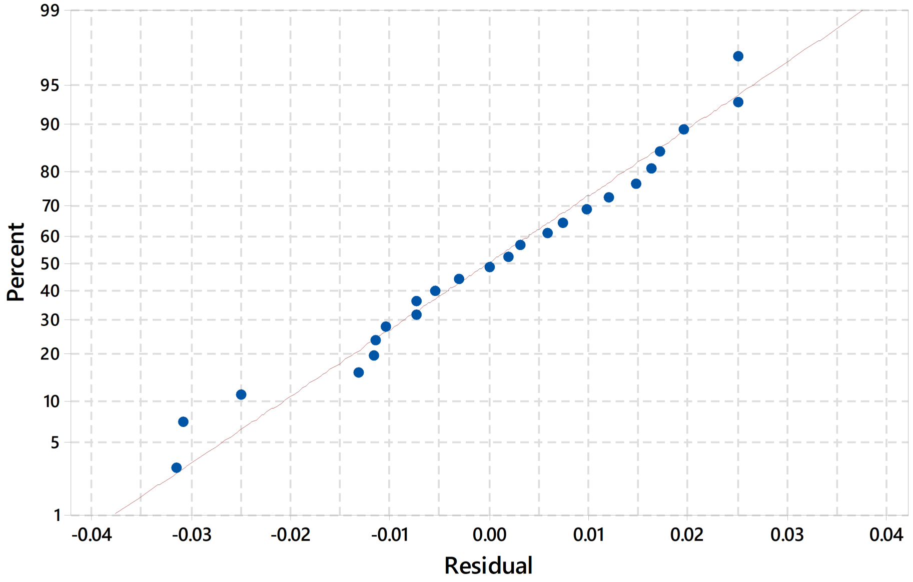


Figure 6.15: Half normal plot of the residuals for the transverse Young's modulus.

are seen in Figure 6.17.

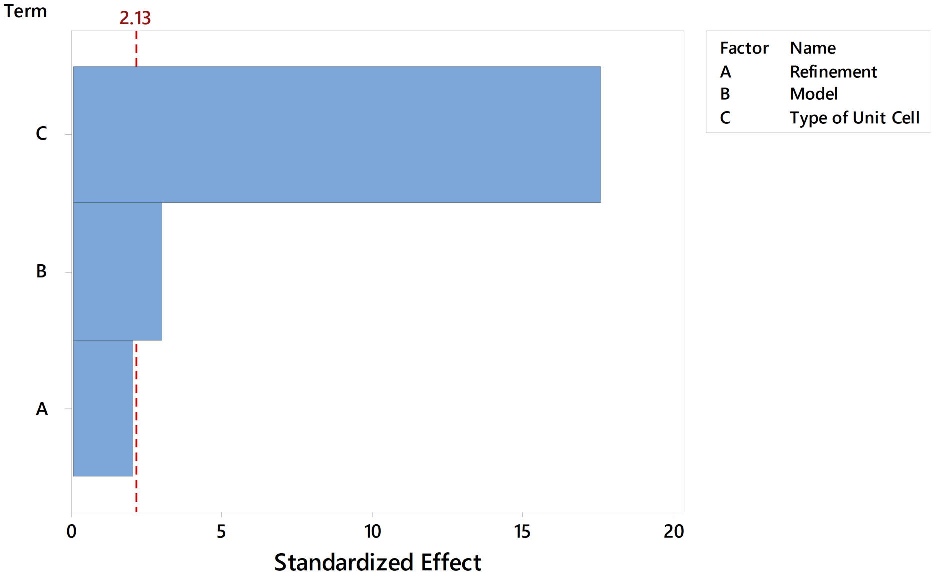


Figure 6.16: Pareto chart showing the significant effects on the transverse Young's modulus.

For this study, the predicted transverse modulus is compared to that of the simulated one as well as the popular FGM model. The predicted transverse modulus is found to be within 0.17%, 0.08%, 1.10%, and 0.21% for the 3 by 3, 4 by 4 , 5 by 5, and 6 by 6 braided composites, respectively. Whereas, when compared the the FGM model for predicting the transverse Young's modulus, the modulus is found to be larger than 17%. These results are summarized in Table 6.5. In Table 6.5, the difference between the predicted modulus and the FEA modulus, as well as the difference between the FGM modulus and the FEA modulus are shown to the right of their respective columns.

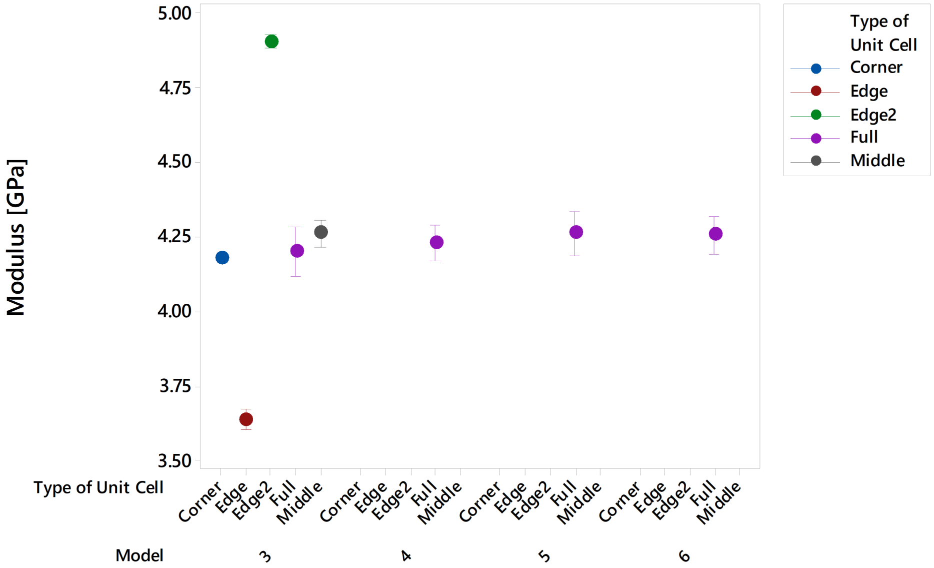


Figure 6.17: Transverse results from the DoE, splitting the number of active cams further by the type of sub-unit cell.

6.5 Conclusions

The progression to the development of a new finite element model for predicting the properties of 3D braids produced from a 3D rotary braider is provided. Additionally, the use of sub-modelling is used to determine the effective stiffness of the previous sub-unit cells. These stiffnesses are combined using a spring based modelling method to further predict the properties of the larger braided structures. The results predicted from the spring based model are compared to those of a full FE analysis and are found to be within 2.78% for the axial modulus, and within 1.10% for the transverse modulus.

Table 6.5: Comparison of the transverse modulus predicted by the model presented to the full-unit cell results and to the well established FGM model.

Model	**FEA Modulus [GPa]**	**Predicted Modulus [GPa]**	**Predicted Difference**	**FGM [GPa]**	**FGM Difference**
3 by 3	4.191	4.198	0.17%	4.937	17.80%
4 by 4	4.216	4.213	0.08%	5.055	19.89%
5 by 5	4.270	4.223	1.10%	5.134	20.23%
6 by 6	4.239	4.231	0.21%	5.190	22.42%

References

[7] F. Ko, “Tensile strength and modulus of a three-dimensional braid composite,” in *Composite Materials (Seventh Conference)*, 1984, pp. 392 –403.

[11] L. Chen, X. Tao, and C. Choy, “Mechanical analysis of 3-d braided composites by the finite multiphase element method,” *Composites Science and Technology*, vol. 59, no. 16, pp. 2383–2391, 1999. DOI: 10.1016/s0266-3538(99)00087-1.

[38] S. Green, M. Matveev, A. Long, D. Ivanov, and S. Hallett, “Mechanical modelling of 3d woven composites considering realistic unit cell geometry,” *Composite Structures*, vol. 118, pp. 284 –293, 2014. DOI: 10.1016/j.compstruct.2014.07.005.

[39] C. Ayranci and J. P. Carey, “Predicting the longitudinal elastic modulus of braided tubular composites using a curved unit-cell geometry,” *Composites: Part B*, vol. 41, pp. 229 –235, 2010. DOI: 10.1016/j.compositesb.2009.10.006.

[40] C. Sun and R. Vaidya, “Prediction of composite properties from a representative volume element,” *Composites Science and Technology*, vol. 56, no. 2, pp. 171–179, 1996. DOI: 10.1016/0266-3538(95)00141-7.

[73] Dassault Systèmes, *Abaqus 2016*, Software, 2016. [Online]. Available: http://www.simulia.com.

[74] Minitab, Inc., *Minitab 18 statistical software*, Software, 2018. [Online]. Available: www.minitab.com.

Chapter 7

Conclusions, Recommendations, & Future Work

7.1 Conclusions

The goals of this thesis are to design and build a rotary three-dimensional braiding machine, develop geometric and geometric CAD model, develop an FE model using sub-unit cells, and finally use their stiffnesses to predict the properties of a full braided structure as well as predict the properties of similar braided structures.

The 3D rotary braider consists of a few sub-assemblies that work together to produce braided structures. The braider is designed to allow for an easy method of increasing the size of the machine as well as to minimize the complexity of the machine. The machine is computer controlled and able to be controlled from the in-house software that calculates the movements and sends the appropriate commands to the braiders microcontroller. Further, the braids produced are analysed to determine the external braid angle and the braid pitch.

Later, the process of using machine emulation to create the geometric model of a braided structure is provided. The method is automated through the creation of an in-house software, and the properties of the braid. The software further demonstrated the ability to assist in the tailoring of braid properties by providing upper and lower bounds based on the rule of mixtures and angled lamina, as well as providing the results from the well-known FGM model. In addition to creating a geometric model, the software is able to create a geometric CAD model.

The geometric model is converted to two FEA models in Chapter 6 by importing the geometric CAD models into commercial FEA software, and adding the appropriate boundary conditions. The first model is for the full-unit cell, while the second model is a new model analysing the sub-unit cells. The full unit cell models were used to determine the axial and transverse moduli of a 3 by 3, 4 by 4, 5 by 5, and 6 by 6 braided structures. While the sub-unit cell models were used to determine the individual stiffnesses of the sub-unit cells in the axial and transverse directions. Later a method of recombining the results from the sub-unit is proposed.

In addition to the conclusions and new sub-unit cell model above, this thesis is able to demonstrate the benefit of sub-unit cell modelling and analysis. This is demonstrated by the development of a new method/model for recombining the sub-unit cells. It is shown that the sub-unit cell modelling can be used to accurately predict the properties of 3D braided structures as well as predict the properties for a family of similar braided structures. The results from this new method are directly compared to the results predicted from the well-established FGM method and found the new method advantageous for determining the axial and transverse moduli. This makes the new method/model ideal for calculating the elastic properties of 3D braided composite materials, and provides more flexibility when producing structures with tailored properties.

7.2 Future Work

In this thesis, an in-house rotary 3D braider is developed, a method of producing geometric and geometric CAD models are developed and automatized the elastic properties of the 3D braided composites are predicted. Expansion of the machine to include individually controlled motors would allow the advancement of creating complex braided structures. These complex structures can include braids that diverge, or purposely introduce separations with in the structure thus further increasing the tailorability of the 3D braids.

Adaptation of the geometric model to account for the changing cross-sectional area of the yarns as they undulate through the braided structure could further increase the accuracy when comparing to the physical braided geometry.

Nonlinear finite element analysis would provide the full stress strain curve of the braided structure. By determining the full stress strain curve other critical design points can be determined. This further development could lead to the prediction of the failure points including the ultimate strength for the braided structure. Finally a comparison of axial and transverse moduli from the braids produced from the developed machine would further verify the results obtained in this study.

Bibliography

[1] D. Gay, S. V. Hoa, and S. W. Tsai, *Composite Materials: Design and Applications*. CRC Press, 2002, ISBN: 9781420031683. DOI: 10.1201/9781420031683.

[2] M. Sauer, M. Kühnel, and E. Witten, "Composites market report (2017)," AVK & CCeV, Tech. Rep., 2017.

[3] D. D. L. Chung, "Applications of composite materials," in *Composite Materials: Functional Materials for Modern Technologies*. London: Springer London, 2003, pp. 1–13, ISBN: 978-1-4471-3732-0. DOI: 10.1007/978-1-4471-3732-0_1. [Online]. Available: https://doi.org/10.1007/978-1-4471-3732-0_1.

[4] J. Hale, "Boeing 787 from the ground up," *Areo Quarterly*, no. 4, 2006.

[5] A. K. Kaw, *Mechanics of Composite Materials*. Taylor & Francis Inc, Oct. 1, 2005, 490 pp., ISBN: 0-8493-1343-0.

[6] R. T. Brown, "Through-the-thickness braided composites for aircraft applications," in *FAA, Ninth DOD(NASA)FAA Conference on Fibrous Composites in Structural Design*, vol. 3, 1992, pp. 1231–1247.

[7] F. Ko, "Tensile strength and modulus of a three-dimensional braid composite," in *Composite Materials (Seventh Conference)*, 1984, pp. 392 –403.

[8] F. Ko and C. Pastore, "Structure and properties of an integrated 3-d fabric for structural composites," in *Recent Advances in Composites in the United States and Japan*, ASTM International, 1985, pp. 428–439. DOI: 10.1520/stp32805s.

[9] J.-M. Yang, C.-L. Ma, and T.-W. Chou, "Fiber inclination model of three-dimensional textile structural composites," *Journal of Composite Materials*, vol. 20, no. 5, pp. 472–484, 1986. DOI: 10.1177/002199838602000505.

[10] G.-W. Du and F. Ko, "Unit cell geometry of 3-d braided structures," *Journal of Reinforced Plastics and Composites*, vol. 12, no. 7, pp. 752 –768, 1993. DOI: 10.1177/073168449301200702.

[11] L. Chen, X. Tao, and C. Choy, "Mechanical analysis of 3-d braided composites by the finite multiphase element method," *Composites Science and Technology*, vol. 59, no. 16, pp. 2383–2391, 1999. DOI: 10.1016/s0266-3538(99)00087-1.

[12] T. Zeng, L. zhi Wu, and L. cheng Guo, "Mechanical analysis of 3d braided composites: A finite element model," *Composite Structures*, vol. 64, no. 3-4, pp. 399–404, 2004. DOI: 10.1016/j.compstruct.2003.09.041.

[13] N. Tolosana, S. Lomov, and A. Miravete, “Development of a geometrical model for a 3d braiding unit call based on machine emulation,” 2007.

[14] N. Tolosana, M. Carrera, R. G. de Villoria, L. Castejon, and A. Miravete, “Numerical analysis of three-dimensional braided composite by means of geometrical modeling based on machine emulation,” *Mechanicas of Advanced Materials and Structures*, vol. 19, pp. 207 –215, 2012, ISSN: 1537-6494. DOI: 10.1080/15376494.2011.578784.

[15] C. Zhang, X. Xu, and K. Chen, “Application of three unit-cells models on mechanical analysis of 3d five-directional and full five-directional braided composites,” *Applied Composite Materials*, vol. 20, no. 5, pp. 803–825, 2012. DOI: 10.1007/s10443-012-9309-0.

[16] H. Y. Yang, X. G. Zhou, J. S. Yu, and Z. Luo, “Entity simulation of rectangular preform for 3d braided composite based on ansys,” *Advanced Materials Research*, vol. 800, pp. 336 –340, 2013, ISSN: 1662-8985. DOI: 10.4028/www.scientific.net/amr.800.336.

[17] C. Zhang and X. Xu, “Finite element analysis of 3d braided composites based on three unit-cells models,” *Composite Structures*, vol. 98, pp. 130–142, 2013. DOI: 10.1016/j.compstruct.2012.11.003.

[18] H. Ahn and W.-R. Yu, “Mechanical analysis of 3d braided and woven composites using fiber-based continuum analysis,” *Composite Structures*, vol. 160, pp. 1105–1118, 2017. DOI: 10.1016/j.compstruct.2016.11.003.

[19] D. D. L. Chung, *Composite Materials: Science and Applications*, Second Edition. New York: Springer, 2010, ISBN: 978-1-84882-830-8.

[20] F. K. Ko, “Three-dimensional fabrics for composites,” *Textile Structural Composites*, pp. 129–171, 1989.

[21] R. A. Florentine, “Magnaweave process - from and fundamentals to applications,” *Textile Research Journal*, pp. 620 –623, 1981.

[22] M. F. Smith, “Apparatus and method for automated braiding of square rope and rope product produced thereby,” U.S. Patent US4803909A, Feb. 1989.

[23] M Tsuzuki, M Kimbara, K Fukuta, and A Machii, “Three-dimensional fabric woven by interlacing threads with rotor driven carriers,” U.S. Patent US5067525, Nov. 1991.

[24] F. Schreiber, F. Ko, H. Yang, E. Amalric, and T. Gries, “Novel three-dimensional braiding approach and its products,” in *ICCM-17: 17th International Conference on Composite Materials*, 2009.

[25] D. Whyte, “Structure and properties of 3-d braid reinforced composites,” PhD thesis, Drexel University, United States, Jan. 1986.

[26] C.-L. Ma, J.-M. Yang, and T.-W. Chou, “Elastic stiffness of three-dimensional braided textile structural composites,” in *Composite Materials: Testing and Design (Seventh Conference)*, ASTM International, 1986, pp. 404–404–18. DOI: 10.1520/STP35360S.

[27] M. B. Dow and H. B. Dexter, "Development of stitched, braided and woven composite structures in the act program and at langley research center," 1997.

[28] Y. T. Gao, F. K. Ko, and H. Hu, "Integrated design for manufacturing of braided preforms for advanced composites part ii: 3d braiding.," *Applied Composite Materials*, vol. 20, no. 6, pp. 1065 –1075, 2013, ISSN: 0929189X. DOI: 10.1007/s10443-012-9304-5.

[29] A. Bogdanovich and D. Mungalov, "Recent advancements in manufacturing 3-d braided preforms and composites," *Composite Systems - Macrocomposites, Microcomposites, Nanocomposites*, pp. 61 –72, 2003.

[30] R. A. Florentine, "Apparatus for weaving a three-dimensional article," U.S. Patent US4312261A, Jan. 1982.

[31] R. M. Roberts and W. A. Douglas, "3 dimensional braiding apparatus," U.S. Patent US5337647A, Aug. 1994.

[32] T. Kostar and T.-W. Chou, "A methodology for cartesian braiding of three-dimensional shapes and special structures," *Journal of Materials Science*, vol. 37, pp. 2811 –2814, 2002.

[33] R. T. Brown and E. D. Ratliff, "Method of sequenced braider motion for multi ply braiding apparatus," U.S. Patent US4621560A, Nov. 1986.

[34] R. T. Brown, "Braiding apparatus," U.S. Patent US4753150A, May 1988.

[35] P. Popper and R. McConnell, "A new 3-d braid for integrated parts manufacturing and improved delamination resistance - the 2-step process," in *The 32nd International SAMPE symposium and exhibition*, 1987, pp. 92–103, ISBN: 0938994344.

[36] W. Li and A. Shiekh, "Effect of processes and processing parameters on 3-d braided preforms for composites," *SAMPE Q.; (United States)*, vol. 19:4, Jul. 1988.

[37] F. K. Ko, F. Schreiber, H.-J. Yand, and T. Gries, "Recent advancements in three-dimensional braiding," in *Proceedings of the 1st World Conference on 3D-Fabrics and their Applications*, Manchester, UK, 2008.

[38] S. Green, M. Matveev, A. Long, D. Ivanov, and S. Hallett, "Mechanical modelling of 3d woven composites considering realistic unit cell geometry," *Composite Structures*, vol. 118, pp. 284 –293, 2014. DOI: 10.1016/j.compstruct.2014.07.005.

[39] C. Ayranci and J. P. Carey, "Predicting the longitudinal elastic modulus of braided tubular composites using a curved unit-cell geometry," *Composites: Part B*, vol. 41, pp. 229 –235, 2010. DOI: 10.1016/j.compositesb.2009.10.006.

[40] C. Sun and R. Vaidya, "Prediction of composite properties from a representative volume element," *Composites Science and Technology*, vol. 56, no. 2, pp. 171–179, 1996. DOI: 10.1016/0266-3538(95)00141-7.

[41] Y. Wang and A. S. D. Wang, “Geometric mapping of yarn structures due to shape change in 3-d braided composites,” *Composite Science and Technology*, vol. 54, no. 4, pp. 359 –370, 1995. DOI: 10.1016/0266-3538(95)00059-3.

[42] T. Alpyildiz, “3d geometrical modelling of tubular braids,” *Textile Research Journal*, vol. 82, no. 5, pp. 443–453, 2011. DOI: 10.1177/0040517511427969.

[43] Y. T. Gao, F. K. Ko, and H. Hu, “Integrated design for manufacturing of braided preforms for advanced composites part i: 2d braiding.,” *Applied Composite Materials*, vol. 20, no. 6, pp. 1007 –1023, 2013, ISSN: 0929189X. DOI: 10.1007/s10443-012-9303-6.

[44] S. C. Queka, A. M. Waasa, K. W. Shahwanb, and V. Agaramb, “Analysis of 2d triaxial flat braided textile composites,” *International Journal of Mechanical Sciences*, vol. 45, no. 6-7, pp. 1077 –1096, 2003. DOI: 10.1016/j.ijmecsci.2003.09.003.

[45] G. W. Melenka, B. K. Cheung, J. S. Schofield, M. R. Dawson, and J. P. Carey, “Evaluation and prediction of the tensile properties of continuous fiber-reinforced 3d printed structures,” *Composite Structures*, vol. 153, pp. 866–875, 2016, ISSN: 0263-8223. DOI: 10.1016/j.compstruct.2016.07.018.

[46] S. Z. Sheng and S. van Hoa, “Modeling of 3d angle interlock woven fabric composites,” *Journal of Thermoplastic Composite Materials*, vol. 16, no. 1, pp. 45–58, 2003. DOI: 10.1177/0892705703016001206.

[47] C. Deng, J. J. Jiang, and L. C. Fang, “Algorithm design of four-step three-dimensional braided composite structures in matlab environment,” in *Applied Mechanics and Materials*, ser. Machine design and manufacturing engineering; (ICMDME 2013), vol. 365, Trans Tech Publications, 2013, pp. 1144 –1147. DOI: 10.4028/www.scientific.net/amm.365-366.1144.

[48] M.-Z. Zhang and H.-J. Li, “Automatically generated geometric description of 3d braided rectangle preform,” *Computational Materials Science*, vol. 39, no. 4, pp. 836 –841, 2007. DOI: 10.1016/j.commatsci.2006.10.010.

[49] O. I. Paul, “Modeling and simulation of three dimensionally braided composite and mechanical properties analysis using finite element method (FEM),” *The International Journal of Engineering and Science (IJES)*, vol. 3, no. 3, pp. 1 –8, 2014, ISSN: 2319 - 1813.

[50] S. Smith, D. Romanyk, P. Major, and C. Ayranci, “An investigation on the preparation and mechanical properties of three-dimensional braided composite orthodontic archwires,” *Journal of International Oral Health*, vol. 8, no. 5, pp. 554–559, 2016.

[51] C. Ayranci and J. P. Carey, “Predicting the longitudinal elastic modulus of braided tubular composites using a curved unit-cell geometry,” *Composites Part B: Engineering*, vol. 41, no. 3, pp. 229–235, 2010. DOI: 10.1016/j.compositesb.2009.10.006.

[52] T. Ting, *Anisotropic Elasticity: Theory and Applications*, 4. New York: Oxford University Press, 1996, vol. 63, p. 1056. DOI: 10.1115/1.2787237.

[53] B. A. A. E. H. Love, "Xvi. the small free vibrations and deformation of a thin elastic shell," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 179, pp. 491–546, 1888, ISSN: 0264-3820. DOI: 10.1098/rsta.1888.0016. eprint: http://rsta.royalsocietypublishing.org/content/179/491.full.pdf. [Online]. Available: http://rsta.royalsocietypublishing.org/content/179/491.

[54] T. Ishikawa and T. W. Chou, "Stiffness and strength behaviour of woven fabric composites," *Journal of Materials Science*, vol. 17, no. 11, pp. 3211–3220, 1982. DOI: 10.1007/bf01203485.

[55] T. Ishikawa, K. Koyama, and S. Kobayashi, "Elastic moduli of carbon-epoxy composites and carbon fibers," *Journal of Composite Materials*, vol. 11, no. 3, pp. 332–344, 1977. DOI: 10.1177/002199837701100307.

[56] T. Ishikawa, "Anti-symmetric elastic properties of composite plates of satin weave cloth," *Fibre Science and Technology*, vol. 15, no. 2, pp. 127–145, 1981. DOI: 10.1016/0015-0568(81)90066-x.

[57] T. Ishikawa and T.-W. Chou, "Elastic behavior of woven hybrid composites," *Journal of Composite Materials*, vol. 16, no. 1, pp. 2–19, 1982. DOI: 10.1177/002199838201600101.

[58] W. Johnson, J. Masters, I. Raju, and J. Wang, "Classical laminate theory models for woven fabric composites," *Journal of Composites Technology and Research*, vol. 16, no. 4, p. 289, 1994. DOI: 10.1520/ctr10589j.

[59] G. W. Du and F. K. Ko, "Geometric modeling of 3-d braidied preforms for composites," in *Preceedings of 5th Textile Structural Composites Symposium*, Philadelphia, PA: Drexel University.

[60] L. Tong, A. Mouritz, and M. Bannister, *3D Fibre Reinforced Polymer Composites*. Elsevier Science, 2002, ISBN: 0-08-043938-1.

[61] C. E. Hage, R. Younès, Z. Aboura, M. Benzeggagh, and M. Zoaeter, "Analytical and numerical modeling of mechanical properties of orthogonal 3d CFRP," *Composites Science and Technology*, vol. 69, no. 1, pp. 111–116, 2009. DOI: 10.1016/j.compscitech.2007.10.048.

[62] S. Green, A. Long, B. E. Said, and S. Hallett, "Numerical modelling of 3d woven preform deformations," *Composite Structures*, vol. 108, pp. 747–756, 2014. DOI: 10.1016/j.compstruct.2013.10.015.

[63] M. Tsuzuki, "Three dimensional woven fabric with varied thread orientations," U.S. Patent US5348056A, Sep. 1994.

[64] G. Yilmaz, T. Ellingham, and L.-S. Turng, "Improved processability and the processing-structure-properties relationship of ultra-high molecular weight polyethylene via supercritical nitrogen and carbon dioxide in injection molding," *Polymers*, vol. 10, no. 1, p. 36, 2017. DOI: 10.3390/polym10010036.

[65] K Ramani and N. Parasnis, "Process-induced effects in compression molding of ultra-high molecular weight polyethylene (UHMWPE)," in *Characterization and Properties of Ultra-High Molecular Weight Polyethylene*, ASTM International, 1998, pp. 5–5–19. DOI: 10.1520/stp11906s.

[66] Universal Laser Systems, *Vls3.50*, Hardware. [Online]. Available: https://www.ulsinc.com.

[67] O. StepperOnline, *Nema 17 bipolar stepper motor (17hs19-1684s-pg19)*, Hardware. [Online]. Available: https://www.omc-stepperonline.com/nema-17-stepper-motor-bipolar-l48mm-w-gear-raio-191-planetary-gearbox-17hs19-1684s-pg19.html.

[68] The Mathworks, Inc., *Matlab 2017a r9.2*, Software, 2017. [Online]. Available: http://www.mathworks.com.

[69] Arduino, *Arduino*, Hardware, 2017. [Online]. Available: http://www.arduino.cc/.

[70] National Institute of Health, *ImageJ*, Software, 2018. [Online]. Available: https://github.com/imagej/imagej1.

[71] Z. Hashin, "Theory of fiber reinforced materials," Pennsylvania University, Tech. Rep., Mar. 1972.

[72] Microsoft, *Visual basic for applications*, Software, 2016.

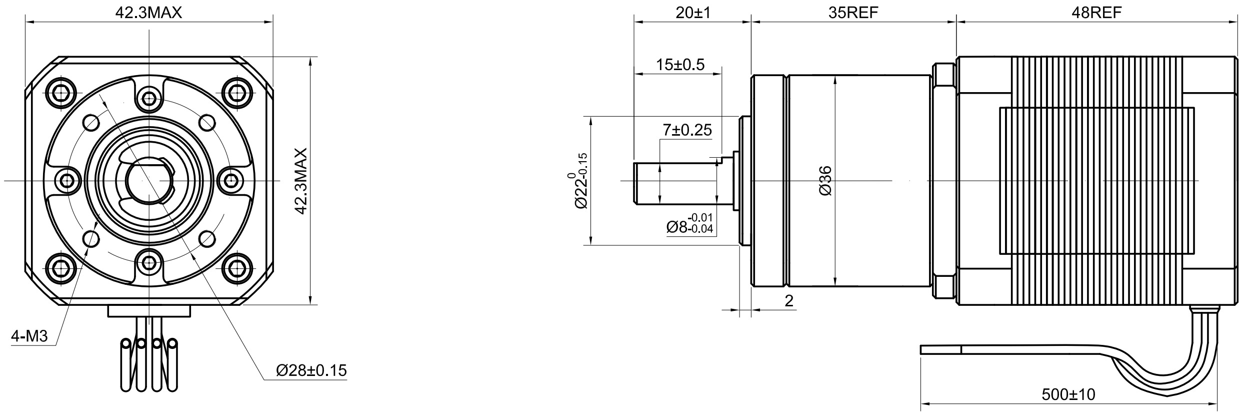
[73] Dassault Systèmes, *Abaqus 2016*, Software, 2016. [Online]. Available: http://www.simulia.com.

[74] Minitab, Inc., *Minitab 18 statistical software*, Software, 2018. [Online]. Available: www.minitab.com.

Appendix A: Braider Documentation

A.1 Motor and Motor Driver Specifications

A.1.1 Stepperonline 17HS19-1684S-PG19 - Datasheet

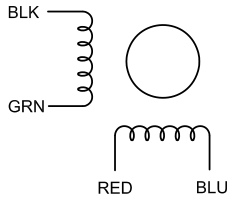


SPECIFICATION \ CONNECTION	BIPOLAR
VOTAGE(VOC)	2.80
AMPS/PHASE	1.68
RESISTANCE/PHASE(Ohms)@25°C	1.65±10%
INDUCTANCE/PHASE(mH)@1KHz	2.80±20%
HOLDING TORQUE w/o GEARBOX(Nm)[lb-in]	0.44[3.89]
GEAR RATIO	$19\frac{38}{187}$
EFFICIENCY	81.00%
STEP ANGLE w/o GEARBOX(°)	1.80
BACKLASH@NO-LOAD	<=1°
MAX.PERMISSIBLE TORQUE(Nm)	3.00
MOMENT PERMISSIBLE TORQUE(Nm)	5.00
SHAFT MAXIMUM AXIAL LOAD(N)	50.00
SHAFT MAXIMUM RADIAL LOAD(N)	100.00
WEIGHT(Kg)[lb]	0.60[1.32]
TEMPERATURE RISE:MAX.80°C（MOTOR STANDSTILL;FOR 2PHASE ENERGIZED）	
AMBIENT TEMPERATURE -10°C~50°C[14°F~122°F]	
INSULATION CLASS B 130°C[266°F]	

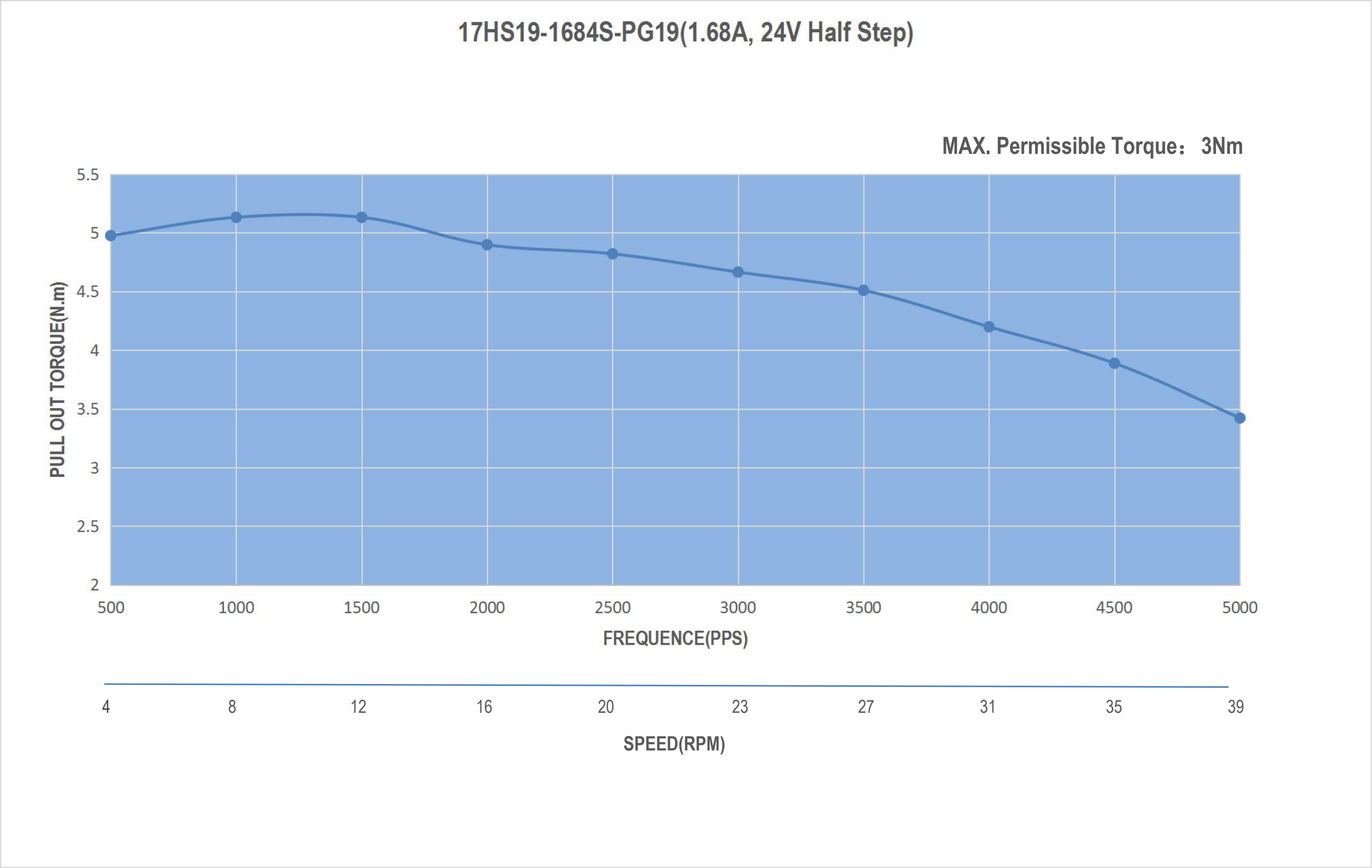
TYPE OF CONNECTION (EXTERN)		MOTOR	
PIN NO	BIPOLAR	LEADS	WINDING
1	A —	BLK	A
2	A\ —	GRN	A\
3	B —	RED	B
4	B\ —	BLU	B\

FULL STEP 2 PHASE-Ex. ,
WHEN FACING MOUNTING END (X)

STEP	A	B	A\	B\	CW	CCW
1	+	+	-	-	↓	↑
2	-	+	+	-		
3	-	-	+	+		
4	+	-	-	+		



STEPPERONLINE Motors&Electronics		APVD		STEPPER MOTOR
		CHKD		
	1:1	DRN		17HS19-1684S-PG19
	SCALE	SIGNATURE	DATE	



17HS19-1684S-PG19(1.68A, 24V Half Step)

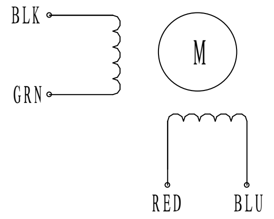
MAX. Permissible Torque：3Nm

A.1.2 Pololu Leadscrew Linear Actuator - Datasheet

HIGH TORQUE HYBRID STEPPING MOTOR SPECIFICATIONS

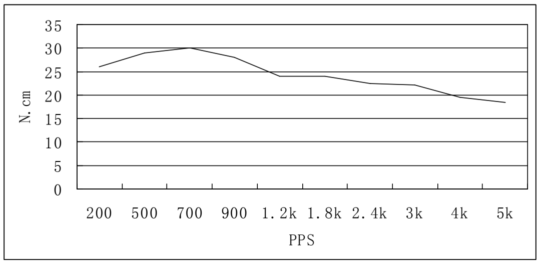
General specifications		Electrical specifications	
Step Angle (∘)	1.8	Rated Voltage (V)	2.8
Temperature Rise (℃)	80 Max (rated current、2 phase on)	Rated Current (A)	1.68
Ambient Temperature (℃)	-20～+50	Resistance Per Phase (±10% Ω)	1.65
Number of Phase	2	Inductance Per Phase (±20% mH)	3.2
Insulation Resistance (MΩ)	100 Min (500VDC)	Holding torque(N.cm)	36
Insulation Class	Class B	Detent Torque(N.cm)	150
Max.radial force (N)	28 (20mm from the flange)	Rotor Torque (N.cm)	54
Max.axial force (N)	10	Weight (kg)	0.28

- Wiring Diagram :



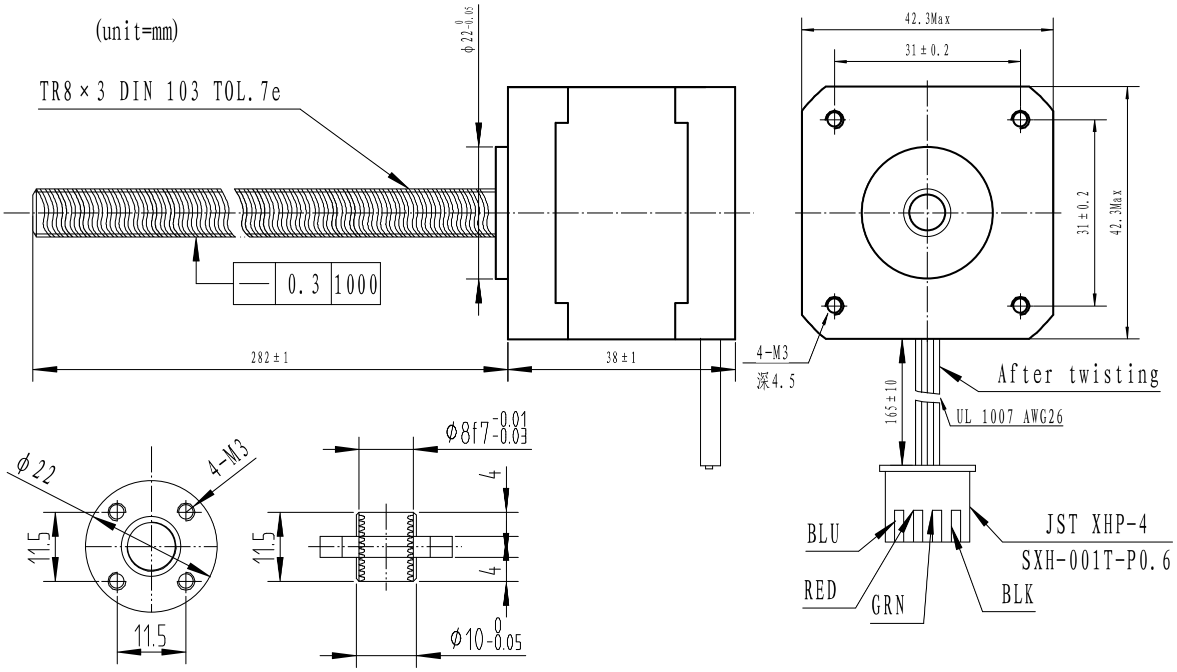
- PULL out turque curve :

VOLTAGE: 24VDC CONSTANT CURRENT : 1.68A HALF STEP



- Dimensions:

(unit=mm)



					SY42STH38-1684A	TECHNICAL CONDITIONS
REV	REVISIONS	DESCRIPTION	BY	DATE		
DRAW	2013/04/23				CHANGZHOU SONGYANG MACHINERY & ELECTRONICS NEW TECHNIC INSTITUTE	063038000
CHECK						
APPROVE						

A.1.3 TB6600HG - Datasheet

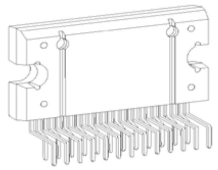
TOSHIBA BiCD Integrated Circuit Silicon Monolithic

TB6600HG

PWM Chopper-Type bipolar
Stepping Motor Driver IC

The TB6600HG is a PWM chopper-type single-chip bipolar sinusoidal micro-step stepping motor driver.
Forward and reverse rotation control is available with 2-phase, 1-2-phase, W1-2-phase, 2W1-2-phase, and 4W1-2-phase excitation modes.
2-phase bipolar-type stepping motor can be driven by only clock signal with low vibration and high efficiency.

TB6600HG



HZIP25-P-1.00F

Weight:
HZIP25-P-1.00F: 7.7g (typ.)

Features

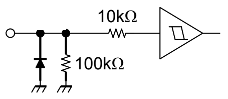
- Single-chip bipolar sinusoidal micro-step stepping motor driver
- Ron (upper + lower) = 0.4 Ω (typ.)
- Forward and reverse rotation control available
- Selectable phase drive (1/1, 1/2, 1/4, 1/8, and 1/16 step)
- Output withstand voltage: Vcc = 50 V
- Output current: I_{OUT} = 5.0 A (absolute maximum ratings, peak)
  I_{OUT} = 4.5 A (operating range, maximal value)
- Packages: HZIP25-P-1.00F
- Built-in input pull-down resistance: 100 kΩ (typ.), (only TQ terminal: 70 k Ω(typ.))
- Output monitor pins (ALERT): Maximum of I_{ALERT} = 1 mA
- Output monitor pins (MO): Maximum of I_{MO} = 1 mA
- Equipped with reset and enable pins
- Stand by function
- Single power supply
- Built-in thermal shutdown (TSD) circuit
- Built-in under voltage lock out (UVLO) circuit
- Built-in over-current detection (ISD) circuit

Pin Functions

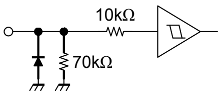
Pin No.	I/O	Symbol	Functional Description	Remark
1	Output	ALERT	TSD / ISD monitor pin	Pull-up by external resistance
2	—	SGND	Signal ground	
3	Input	TQ	Torque (output current) setting input pin	
4	Input	Latch/Auto	Select a return type for TSD.	L: Latch, H: Automatic return
5	Input	Vref	Voltage input for 100% current level	
6	Input	Vcc	Power supply	
7	Input	M1	Excitation mode setting input pin	
8	Input	M2	Excitation mode setting input pin	
9	Input	M3	Excitation mode setting input pin	
10	Output	OUT2B	B channel output 2	
11	—	N_{FB}	B channel output current detection pin	
12	Output	OUT1B	B channel output 1	
13	—	PGNDB	Power ground	
14	Output	OUT2A	A channel output 2	
15	—	N_{FA}	A channel output current detection pin	
16	Output	OUT1A	A channel output 1	
17	—	PGNDA	Power ground	
18	Input	ENABLE	Enable signal input pin	H: Enable, L: All outputs off
19	Input	RESET	Reset signal input pin	L: Initial mode
20	Input	Vcc	Power supply	
21	Input	CLK	CLK pulse input pin	
22	Input	CW/CCW	Forward/reverse control pin	L: CW, H:CCW
23	—	OSC	Resistor connection pin for internal oscillation setting	
24	Output	V_{reg}	Control side connection pin for power capacitor	Connecting capacitor to SGND
25	Output	MO	Electrical angle monitor pin	Pull-up by external resistance

<Terminal circuits>

Input pins
(M1, M2, M3,CLK, CW/CCW,
ENABLE, RESET, Latch/Auto)

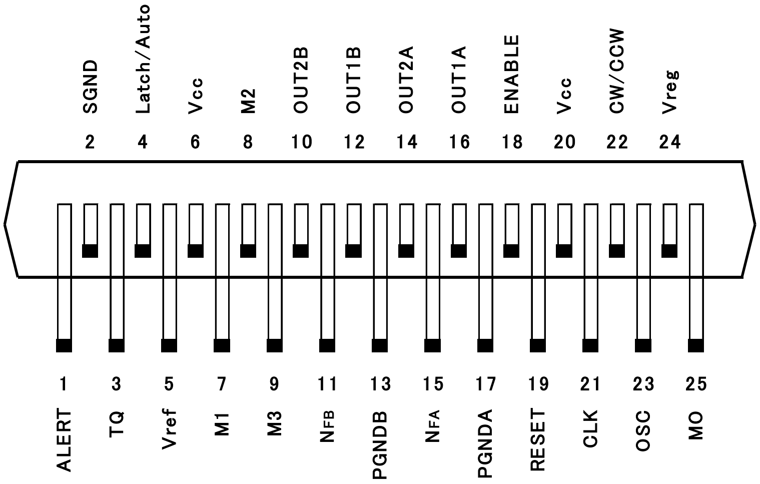


Input pins
(TQ)

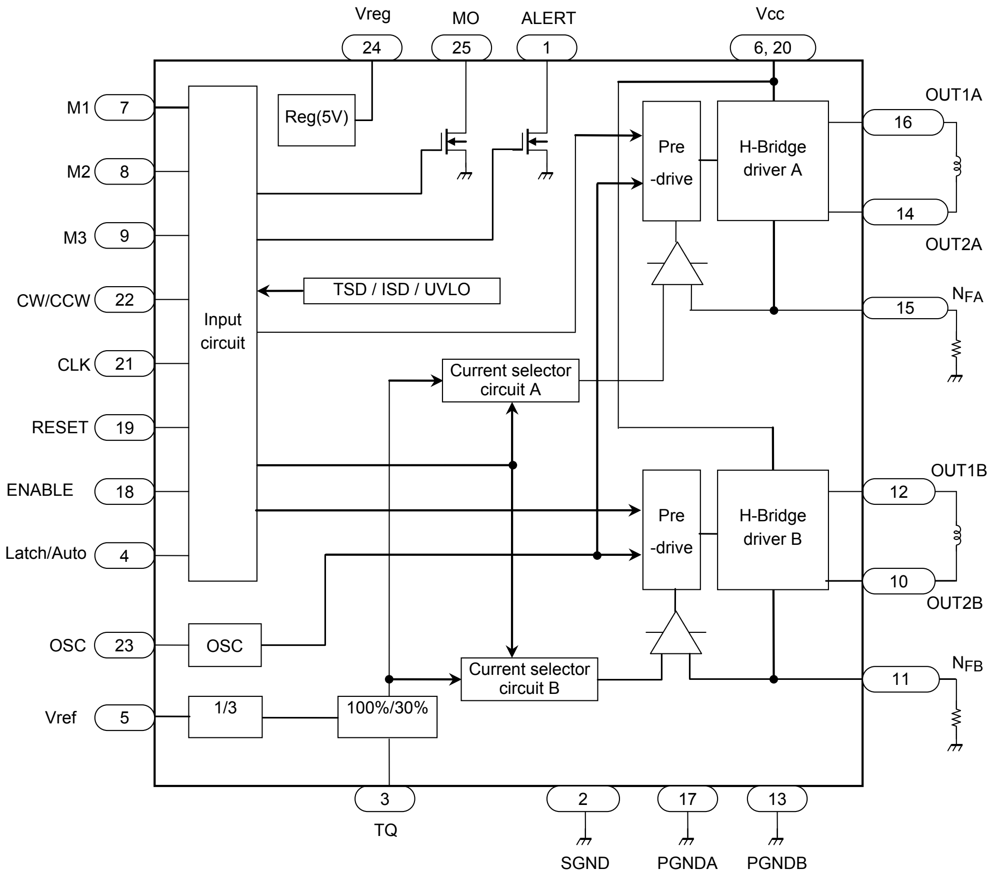


Pin Assignment

（Top View）



Block Diagram



Setting of Vref

Input	Voltage ratio
TQ	
L	30%
H	100%

Description of Functions

1. Excitation Settings

The excitation mode can be selected from the following eight modes using the M1, M2 and M3 inputs. New excitation mode starts from the initial mode when M1, M2, or M3 inputs are shifted during motor operation. In this case, output current waveform may not continue.

Input			Mode (Excitation)
M1	M2	M3	
L	L	L	Standby mode (Operation of the internal circuit is almost turned off.)
L	L	H	1/1 (2-phase excitation, full-step)
L	H	L	1/2A type (1-2 phase excitation A type) (0%, 71%, 100%)
L	H	H	1/2B type (1-2 phase excitation B type) (0%, 100%)
H	L	L	1/4 (W1-2 phase excitation)
H	L	H	1/8 (2W1-2 phase excitation)
H	H	L	1/16 (4W1-2 phase excitation)
H	H	H	Standby mode (Operation of the internal circuit is almost turned off.)

Note: To change the exciting mode by changing M1, M2, and M3, make sure not to set M1 = M2 = M3 = L or M1 = M2 = M3 = H.

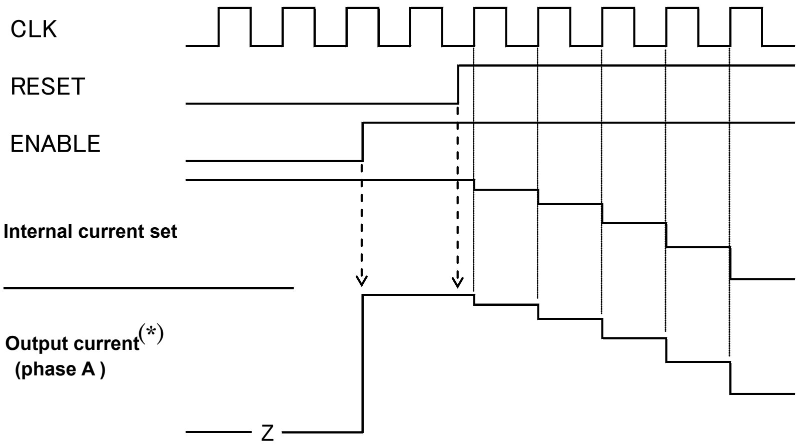
Standby mode

The operation mode moves to the standby mode under the condition M1 = M2 = M3 = L or M1 = M2 = M3 = H.
The power consumption is minimized by turning off all the operations except protecting operation.
In standby mode, output terminal MO is HZ.
Standby mode is released by changing the state of M1=M2=M3=L and M1=M2=M3=H to other state.
Input signal is not accepted for about 200 μs after releasing the standby mode.

2. Function

(1)To turn on the output, configure the ENABLE pin high. To turn off the output, configure the ENABLE pin low.
(2) The output changes to the Initial mode shown in the table below when the ENABLE signal goes High level and the RESET signal goes Low level. (In this mode, the status of the CLK and CW/CCW pins are irrelevant.)
(3) As shown in the below figure of Example 1, when the ENABLE signal goes Low level, it sets an OFF on the output. In this mode, the output changes to the initial mode when the RESET signal goes Low level. Under this condition, the initial mode is output by setting the ENABLE signal High level. And the motor operates from the initial mode by setting the RESET signal High level.

(Example 1)



(*: Output current starts rising at the timing of PWM frequency just after ENABLE pin outputs high.)

Input				Output mode
CLK	CW/CCW	RESET	ENABLE	
↑	L	H	H	CW
↑	H	H	H	CCW
X	X	L	H	Initial mode
X	X	X	L	Z

Command of the standby has a higher priority than ENABLE. Standby mode can be turned on and off regardless of the state of ENABLE.
X: Don't Care

3. Initial Mode

When RESET is used, the phase currents are as follows.

Excitation Mode	Phase A Current	Phase B Current
1/1 (2-phase excitation, full-step)	100%	-100%
1/2A type (1-2 phase excitation A type) (0%, 71%, 100%)	100%	0%
1/2B type (1-2 phase excitation B type) (0%, 100%)	100%	0%
1/4 (W1-2 phase excitation)	100%	0%
1/8 (2W1-2 phase excitation)	100%	0%
1/16 (4W1-2 phase excitation)	100%	0%

current direction is defined as follows.
OUT1A → OUT2A: Forward direction
OUT1B → OUT2B: Forward direction

4. 100% current settings (Current value)

100% current value is determined by Vref inputted from external part and the external resistance for detecting output current. Vref is doubled 1/3 inside IC.

$$Io\ (100\%) = (1/3 \times Vref) \div R_{NF}$$

The average current is lower than the calculated value because this IC has the method of peak current detection.

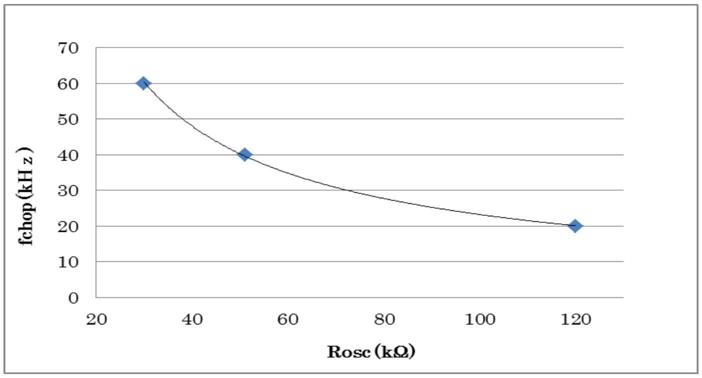
Pleas use the IC under the conditions as follows;

$0.11\Omega \leq R_{NF} \leq 0.5\Omega$, $0.3V \leq Vref \leq 1.95V$

5. OSC

Triangle wave is generated internally by CR oscillation by connecting external resistor to OSC terminal. Rosc should be from 30kΩ to 120kΩ. The relation of Rosc and fchop is shown in below table and figure. The values of fchop of the below table are design guarantee values. They are not tested for pre-shipment.

Rosc(kΩ)	fchop(kHz)		
	Min	Typ.	Max
30	-	60	-
51	-	40	-
120	-	20	-



6. Decay Mode

It takes approximately five OSCM cycles for charging-discharging a current in PWM mode. The 40% fast decay mode is created by inducing decay during the last two cycles in Fast Decay mode.
The ratio 40% of the fast decay mode is always fixed.
The relation between the master clock frequency (fMCLK), the OSCM frequency (fOSCM) and the PWM frequency (fchop) is shown as follows:

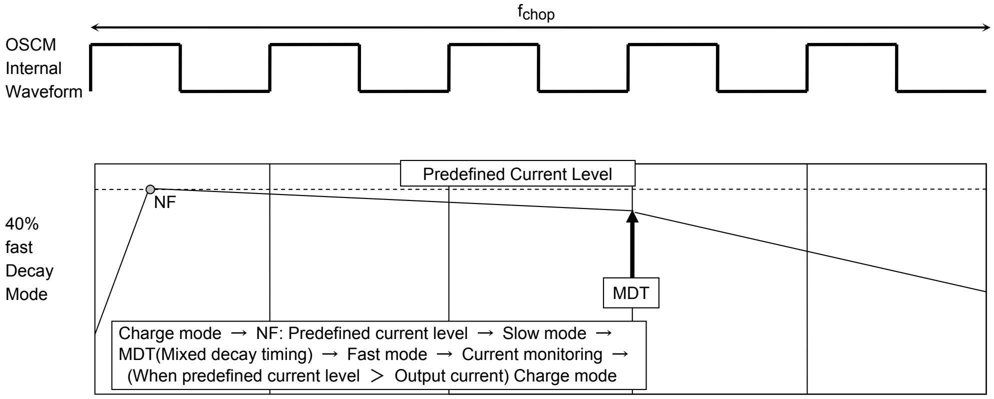
fOSCM = 1/20 ×fMCLK
fchop = 1/100 ×fMCLK

When Rosc=51kΩ, the master clock=4MHz, OSCM=200kHz, the frequency of PWM(fchop)=40kHz.

6-1. Current Waveform and Mixed Decay Mode settings

The period of PWM operation is equal to five periods of OSCM.
The ratio 40% of the fast decay mode is always fixed.
The "NF" refers to the point at which the output current reaches its predefined current level.

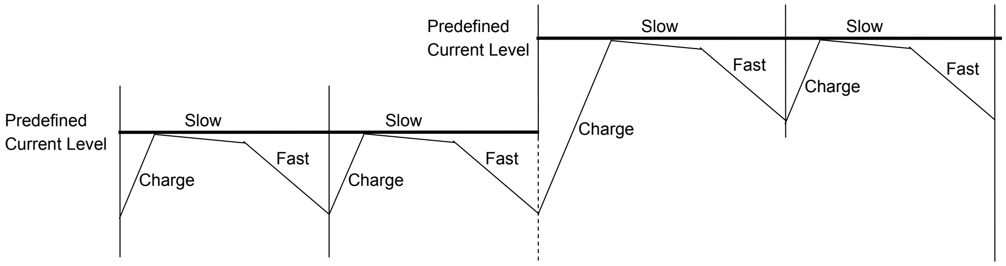
MDT means the point of MDT (MIXED DECAY TIMMING) in the below diagram.



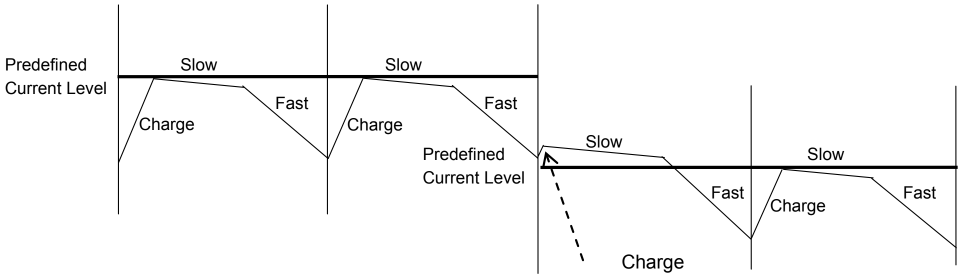
Charge mode → NF: Predefined current level → Slow mode → MDT(Mixed decay timing) → Fast mode → Current monitoring → (When predefined current level ＞ Output current) Charge mode

6-2. Effect of Decay Mode

- Increasing the current (sine wave)

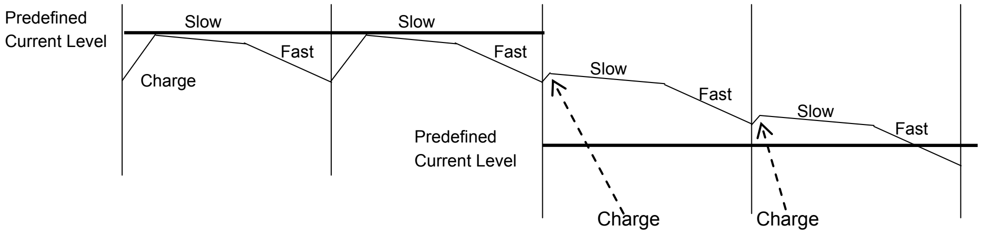


- Decreasing the current (In case the current is decreased to the predefined value in a short time because it decays quickly.)



Even if the output current rises above the predefined current at the RNF point, the current control mode is briefly switched to Charge mode for current sensing.

- Decreasing the current (In case it takes a long time to decrease the current to the predefined value because the current decays slowly.)

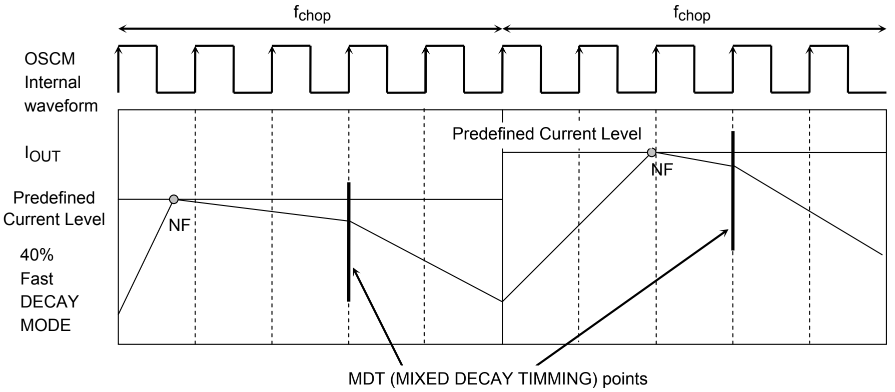


Even if the output current rises above the predefined current at the RNF point, the current control mode is briefly switched to Charge mode for current sensing.

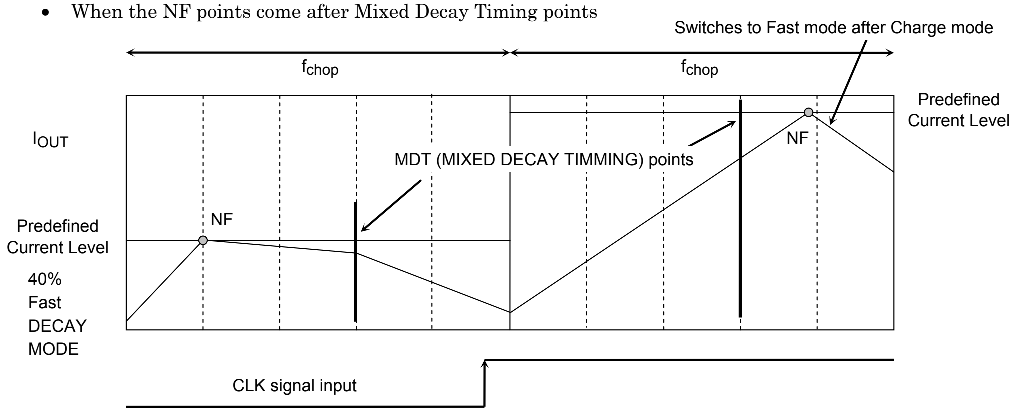
During Mixed Decay and Fast Decay modes, if the predefined current level is less than the output current at the RNF (current monitoring point), the Charge mode in the next chopping cycle will disappear (though the current control mode is briefly switched to Charge mode in actual operations for current sensing) and the current is controlled in Slow and Fast Decay modes (mode switching from Slow Decay mode to Fast Decay mode at the MDT point).

Note: The above figures are rough illustration of the output current. In actual current waveforms, transient response curves can be observed.

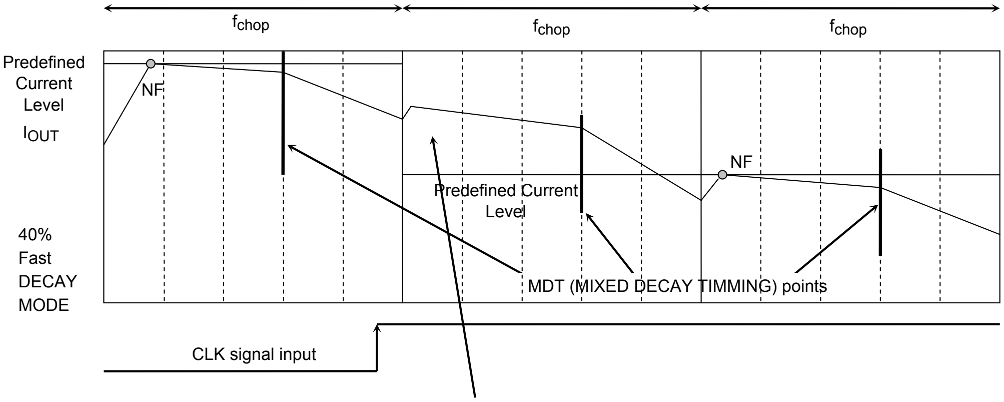
6-3. Current Waveforms in Mixed Decay Mode



- When the NF points come after Mixed Decay Timing points

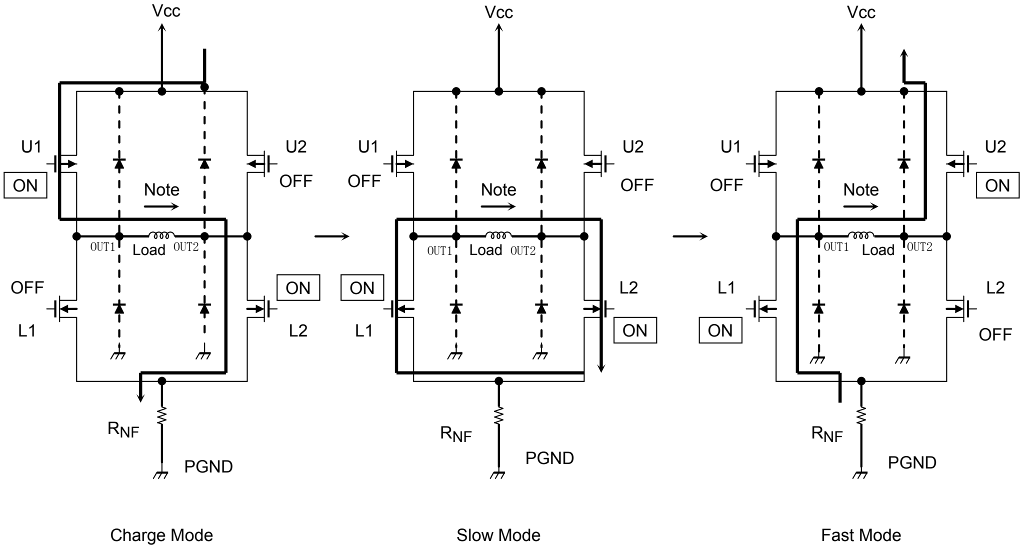


- When the output current value > predefined current level in Mixed Decay mode



Even if the output current rises above the predefined current at the RNF point, the current control mode is briefly switched to Charge mode for current sensing.

Output Stage Transistor Operation Mode



Output Stage Transistor Operation Functions

CLK	U1	U2	L1	L2
CHARGE	ON	OFF	OFF	ON
SLOW	OFF	OFF	ON	ON
FAST	OFF	ON	ON	OFF

Note: The above chart shows an example of when the current flows as indicated by the arrows in the above figures. If the current flows in the opposite direction, refer to the following chart:

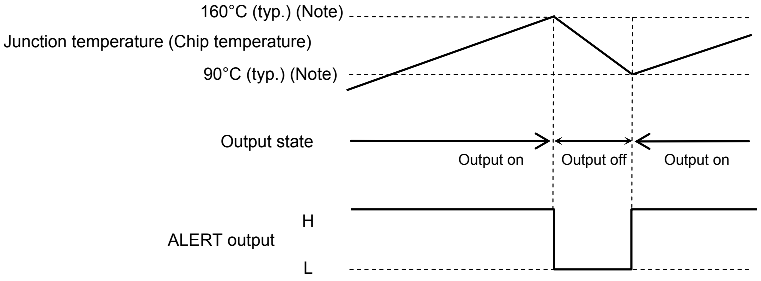
CLK	U1	U2	L1	L2
CHARGE	OFF	ON	ON	OFF
SLOW	OFF	OFF	ON	ON
FAST	ON	OFF	OFF	ON

Upon transitions of above-mentioned functions, a dead time of about 300 ns (Design guarantee value) is inserted respectively.

Thermal Shut-Down circuit (TSD)

(1) Automatic return

TSD = 160°C (typ.) (Note)
TSDhys = 70°C (typ.) (Note)



Automatic return has a temperature hysteresis shown in the above figure.

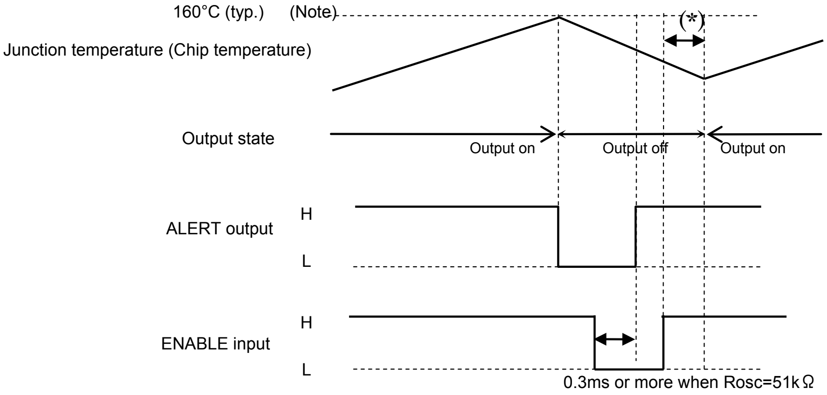
In case of automatic return, the return timing is adjusted at charge start of fchop after the temperature falls to the return temperature (90°C (typ.) in the above figure).

The return period after the temperature falls corresponds to one cycle to two cycles of fchop.

(2) Latch type

TSD = 160°C (typ.) (Note)

(*)Output current starts rising at the timing of PWM frequency just after ENABLE pin outputs high.



The operation returns by programming the ENABLE as H → L → H shown in above figure or turning on power supply and turning on UVLO function. In this time, term of L level of ENABLE should be 0.3ms or more.

To recover the operation, the junction temperature (the chip temperature) should be 90°C or less when ENABLE input is switched from L to H level. Otherwise, the operation does not recover.

Note: Pre-shipment testing is not performed.

·State of internal IC when TSD circuit operates.

The states of the internal IC and outputs, while the shutdown circuit is operating, correspond to the state when ENABLE is L.

The state after automatic return corresponds to the state when ENABLE is H. Please configure the Reset L to rotate the motor from the initial state.

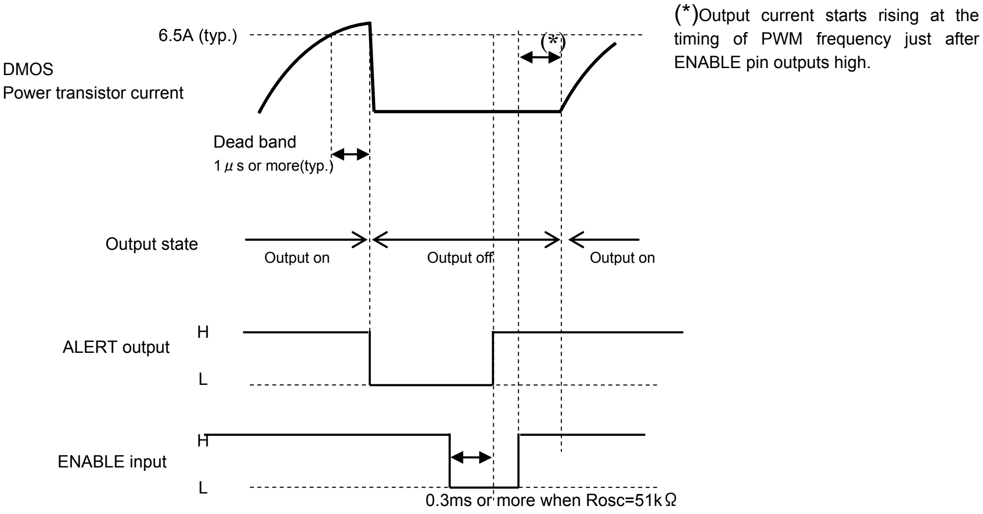
Latch/Auto is an input pin for determining the return method of TSD.
If Latch/Auto pin outputs low, TSD function returns by either of turning on power supply again or programming the ENABLE as H → L → H.
If Latch/Auto pin outputs high, it returns automatically.
In standby mode, TSD function returns automatically regardless of the state of the Latch/Auto pin.
When power supply voltage Vcc is less than 8V, TSD function cannot operate regardless of the state of the Latch/Auto pin.

ISD (Over current detection)

Current that flows through output power MOSFETs are monitored individually. If over-current is detected in at least one of the eight output power MOSFETs, all output power MOSFETs are turned off then this status is kept until ENABLE signal is input. In this time, term of L level of ENABLE should be 0.3ms or more.

Masking term of 1µs or more (typ. when Rosc=51kΩ) (Note) should be provided in order to protect detection error by noise. ISD does not work during the masking term.

Over current detection value ISD=6.5 A (Note)



The operation returns by programming the ENABLE as H → L → H shown in above figure or turning on power supply and turning on UVLO function.

Note: Pre-shipment testing is not performed.

・State of internal IC when ISD circuit operates.

The states of the internal IC and outputs, while the over current detection circuit is operating, correspond to the state when ENABLE is L.

The state after automatic return corresponds to the state when ENABLE is H. Please configure the Reset L to rotate the motor from the initial state.

Return method of ISD

ISD function returns by either of turning on power supply again or programming the ENABLE as H → L → H regardless of the state of the Latch/Auto pin.

In standby mode, ISD function cannot operate.

When power supply voltage Vcc is less than 8V, ISD function cannot operate.

Under Voltage Lock Out (UVLO) circuit

Outputs are shutoff by operating at 5.5 V (Typ.) of Vcc or less.

It has a hysteresis of 0.5 V (Typ.) and returns to output when Vcc reaches 6.0 V (Typ.). The following values are design guarantee values.

･State of internal IC when UVLO circuit operates.

The states of the internal IC and outputs correspond to the state in the ENABLE mode and the initial mode at the same time.

After a return, it can start from the initial mode.

When Vcc falls to around 5.5 V and UVLO operates, output turns off.

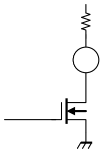
It recovers automatically from the initial mode when both Vcc rise to around 6.0 V or more. The following values are design guarantee values.

ALERT output

ALERT terminal outputs low in detecting either TSD or ISD.

ALERT terminal is connected to power supply externally via pull-up resistance.
V_{ALERT} = 0.5 V (max) at 1 mA

TSD	ISD	ALERT
Under TSD detection	Under ISD detection	Low
Normal	Under ISD detection	
Under TSD detection	Normal	
Normal	Normal	Z

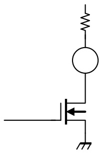


Applied voltage to pull-up resistance is up to 5.5 V. And conducted current is up to 1 mA.
It is recommended to gain 5 V by connecting the external pull-up resistance to Vreg pin.

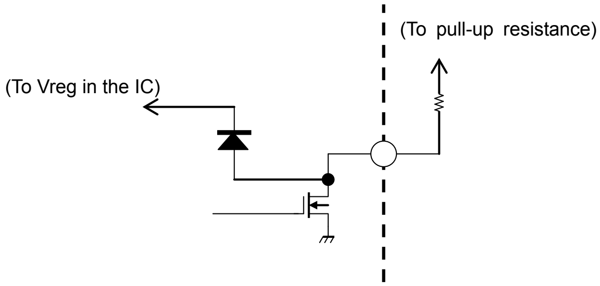
MO output

MO turns on at the predetermined state and output low.
MO terminal is connected to power supply externally via pull-up resistance.
V_{MO} = 0.5 V (max) at 1 mA

State	MO
Initial	Low
Not initial	Z



Applied voltage to pull-up resistance is up to 5.5 V. And conducted current is up to 1 mA.
It is recommended to gain 5 V by connecting the external pull-up resistance to Vreg pin.



Voltage pull-up of MO and ALERT pins

・It is recommended to pull-up voltage to Vreg pin.

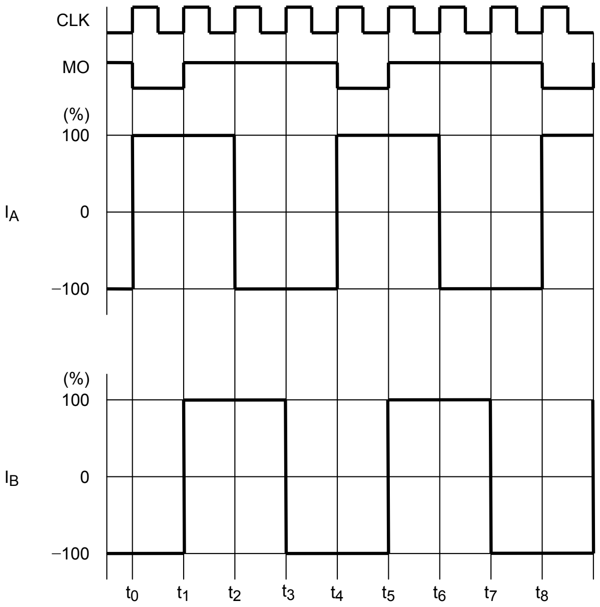
・In case of pull-up to except 5 V (for instance, 3.3 V etc.), it is recommended to use other power supply (ex. 3.3 V) while Vcc output between the operation range. When Vcc decreases lower than the operation range and Vreg decreases from 5 V to 0 V under the condition that other power supply is used to pull-up voltage, the current continues to conduct from other power supply to the IC inside through the diode shown in the figure. Though this phenomenon does not cause destruction and malfunction of the IC, please consider the set design not to continue such a state for a long time.

・As for the pull-up resistance for MO and ALERT pins, please select large resistance enough for the conducting current so as not to exceed the standard value of 1 mA.

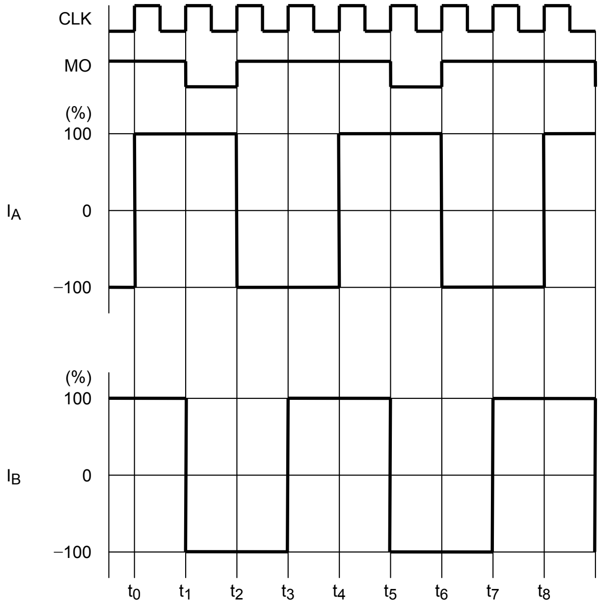
Please use the resistance of 30 kΩ or more in case of applying 5 V, and 20 kΩ or more in case of applying 3.3 V.

Sequence and current level in each excitation mode

1/1-step Excitation Mode (M1: L, M2: L, M3: H, CW Mode)

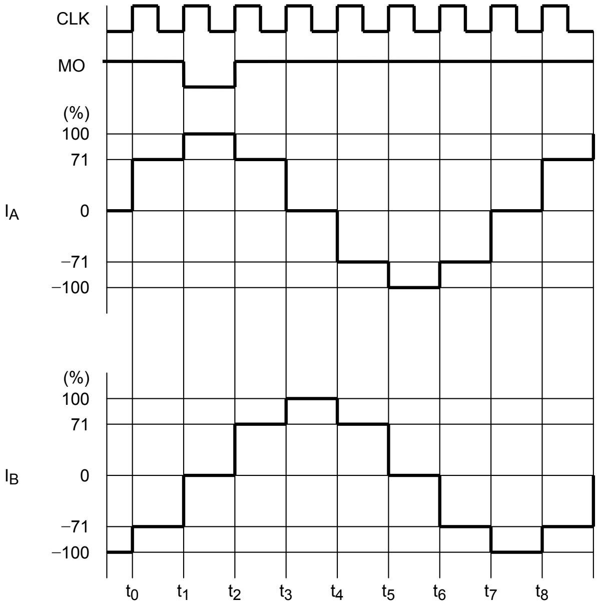


1/1-step Excitation Mode (M1: L, M2: L, M3: H, CCW Mode)

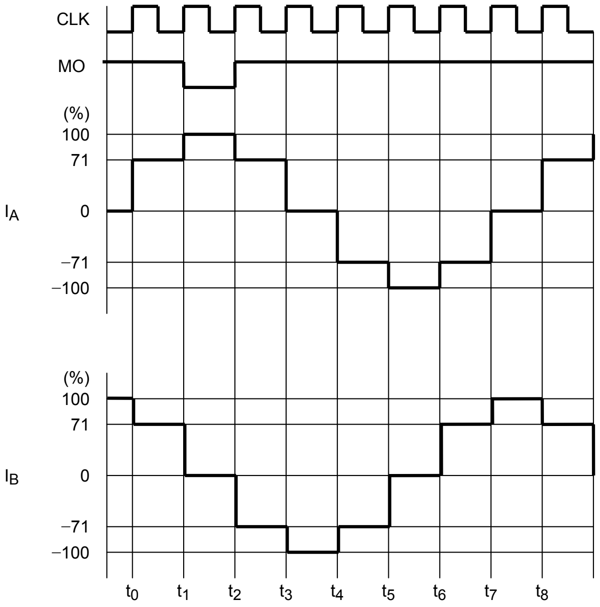


It operates from the initial state after the excitation mode is switched.

1/2-step Excitation Mode (A type) (M1: L, M2: H, M3: L, CW Mode)

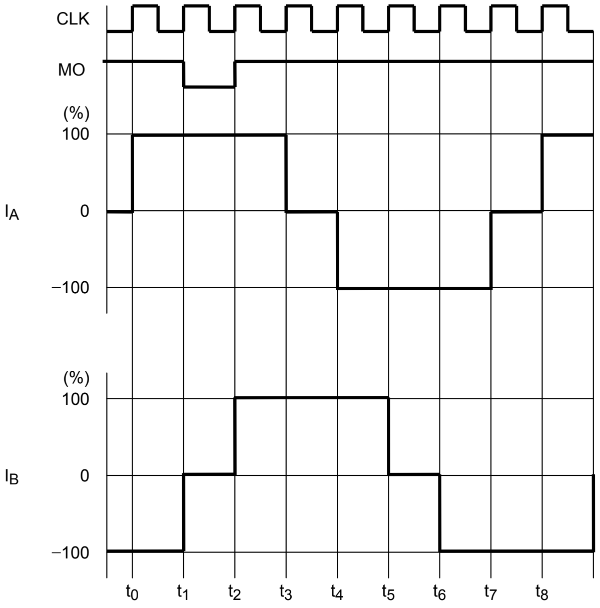


1/2-step Excitation Mode (A type) (M1: L, M2: H, M3: L, CCW Mode)

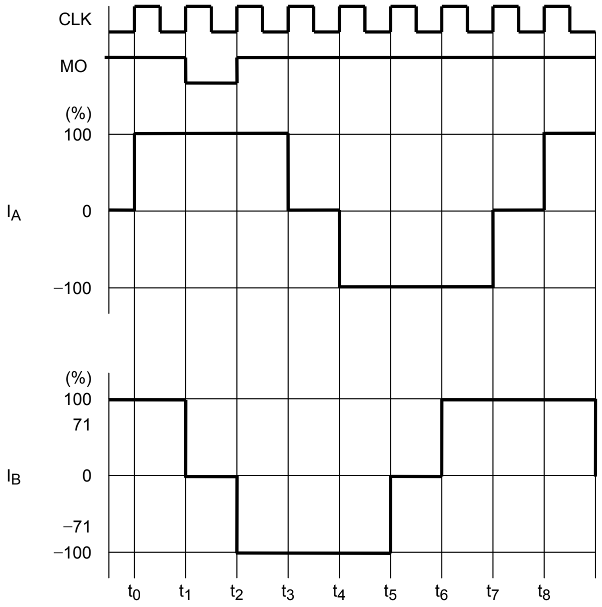


It operates from the initial state after the excitation mode is switched.

1/2-step Excitation Mode (B type) (M1: L, M2: H, M3: H, CW Mode)

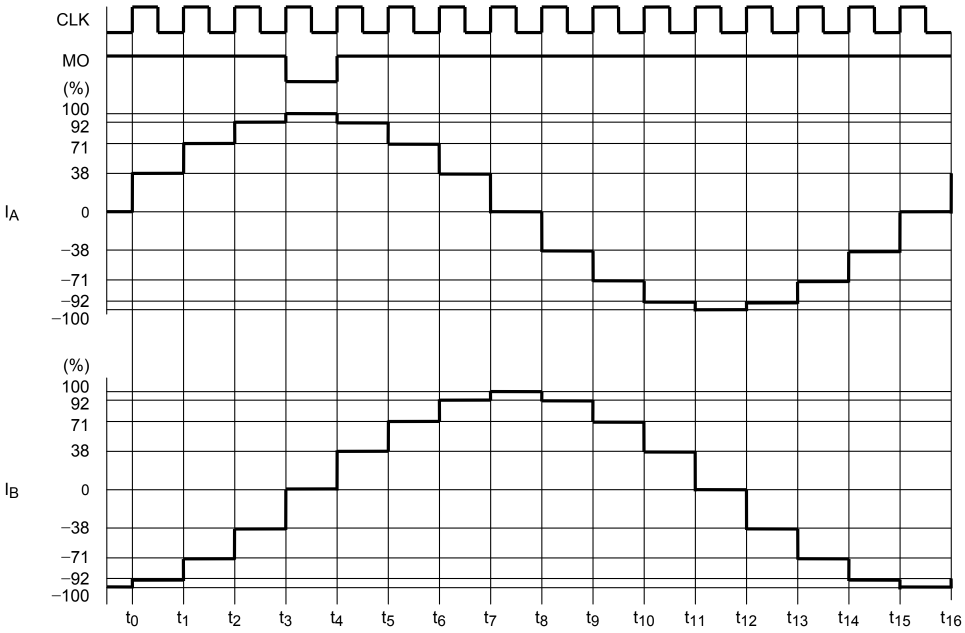


1/2-step Excitation Mode (B type) (M1: L, M2: H, M3: H, CCW Mode)

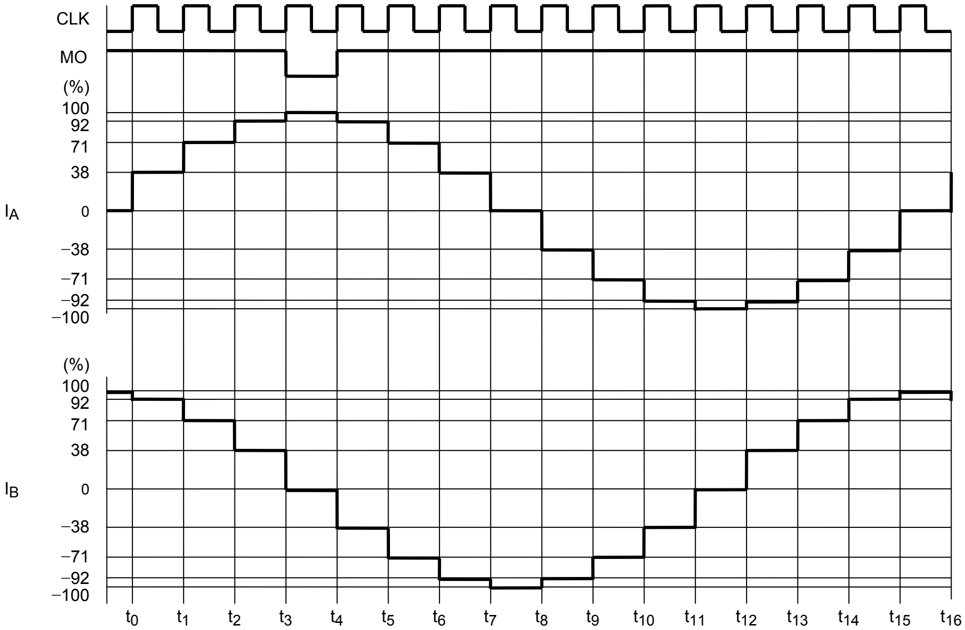


It operates from the initial state after the excitation mode is switched.

1/4-step Excitation Mode (M1: H, M2: L, M3: L, CW Mode)

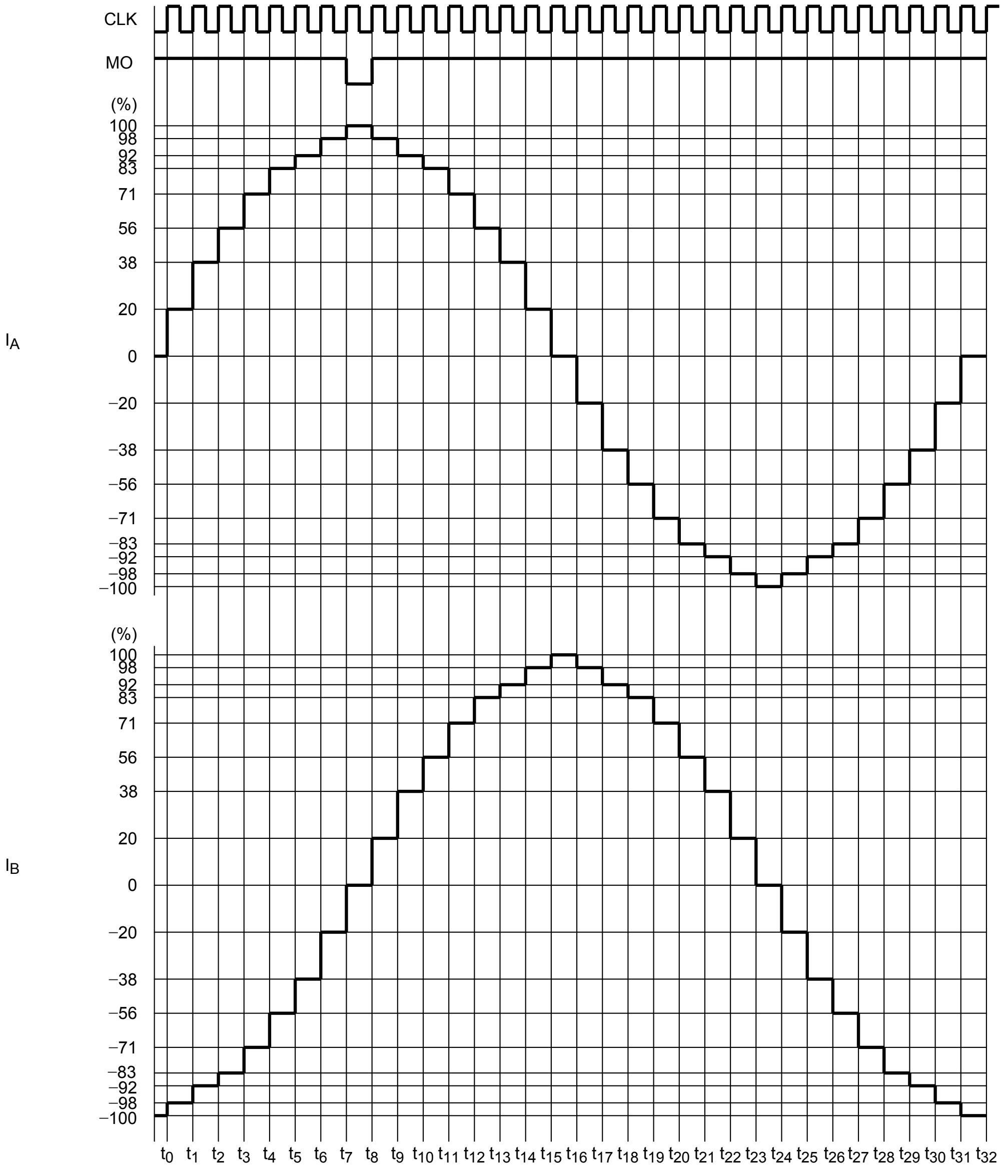


1/4-step Excitation Mode (M1: H, M2: L, M3: L, CCW Mode)



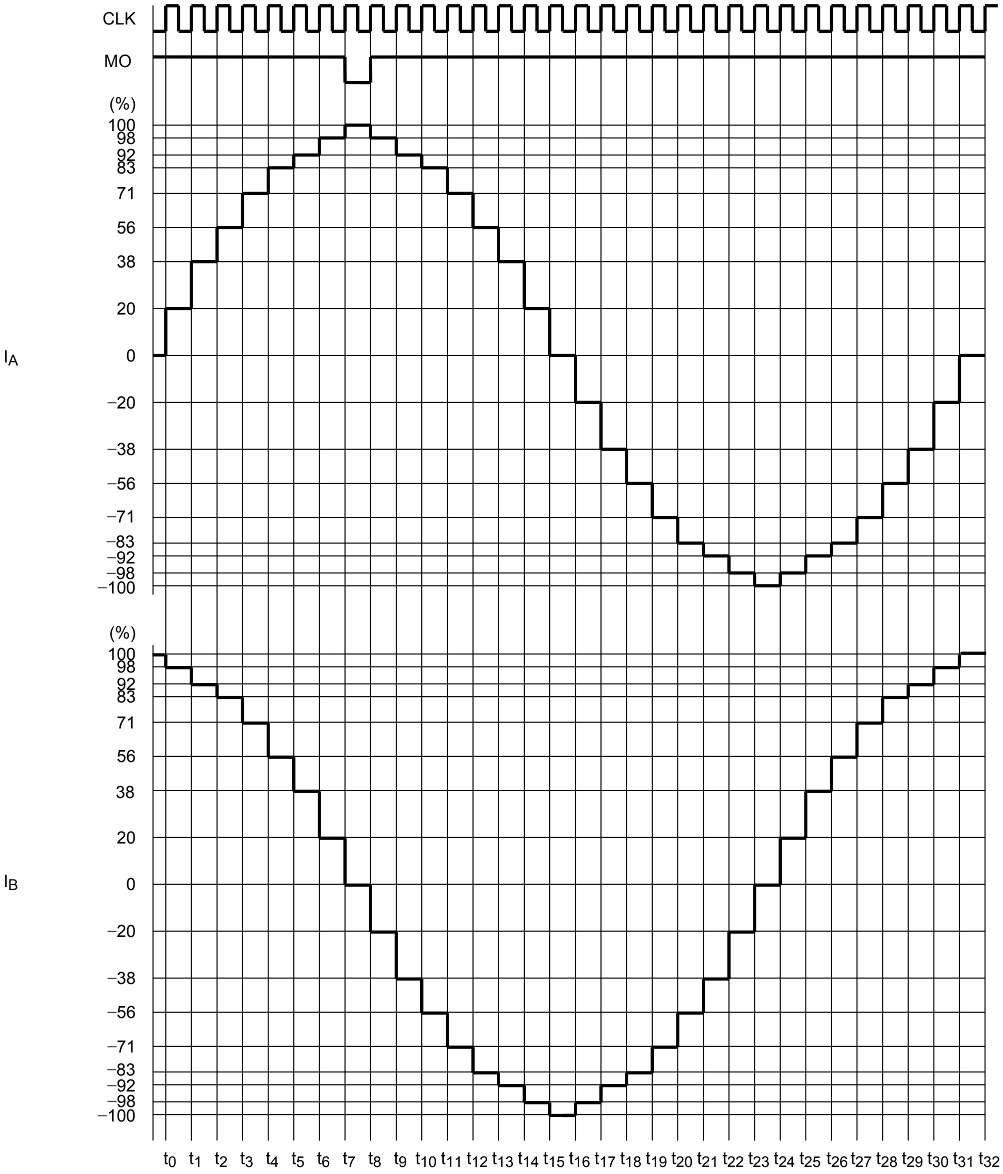
It operates from the initial state after the excitation mode is switched.

1/8-Step Excitation Mode (M1: H, M2: L, M3: H, CW Mode)


CLK
MO
(%)
100
98
92
83
71
56
38
20
0
−20
−38
−56
−71
−83
−92
−98
−100
I_A
I_B
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16} t_{17} t_{18} t_{19} t_{20} t_{21} t_{22} t_{23} t_{24} t_{25} t_{26} t_{27} t_{28} t_{29} t_{30} t_{31} t_{32}

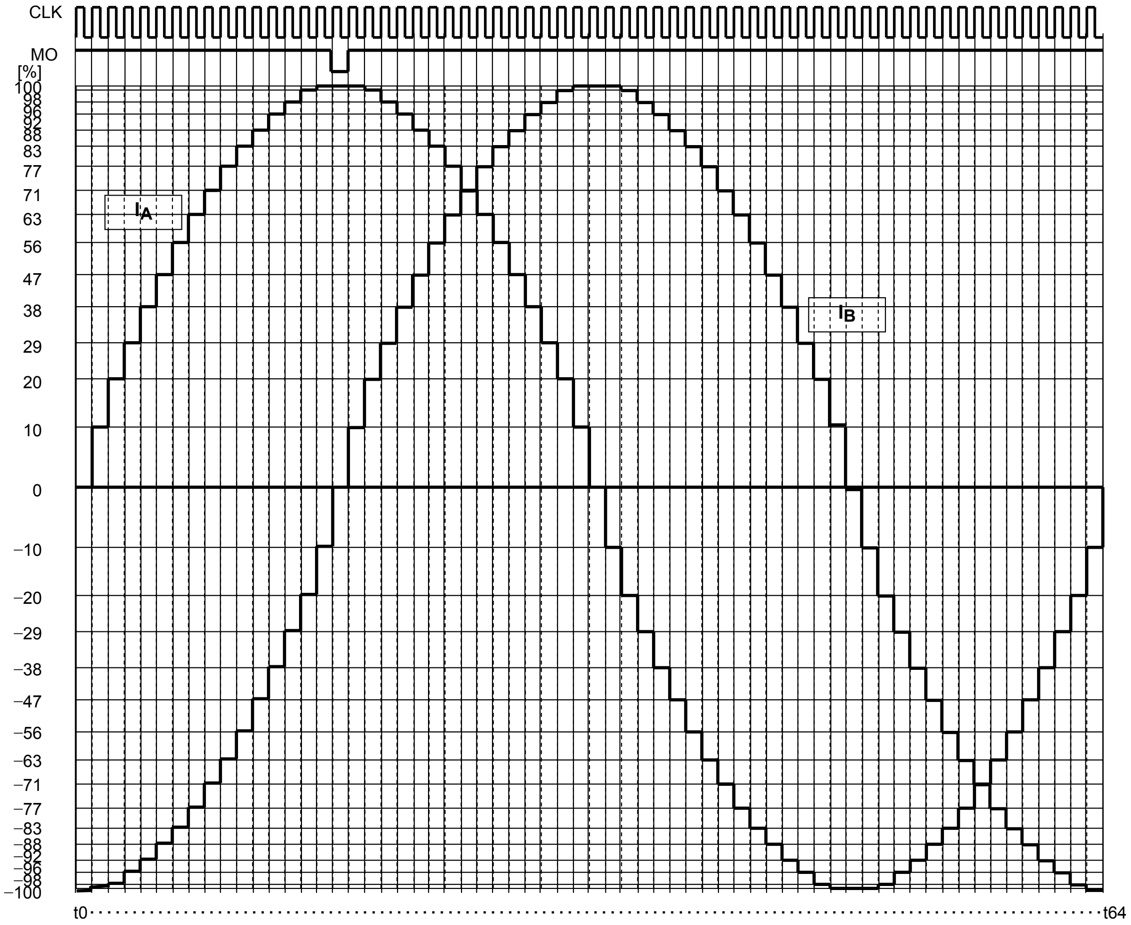
It operates from the initial state after the excitation mode is switched.

1/8-Step Excitation Mode (M1: H, M2: L, M3: H, CCW Mode)



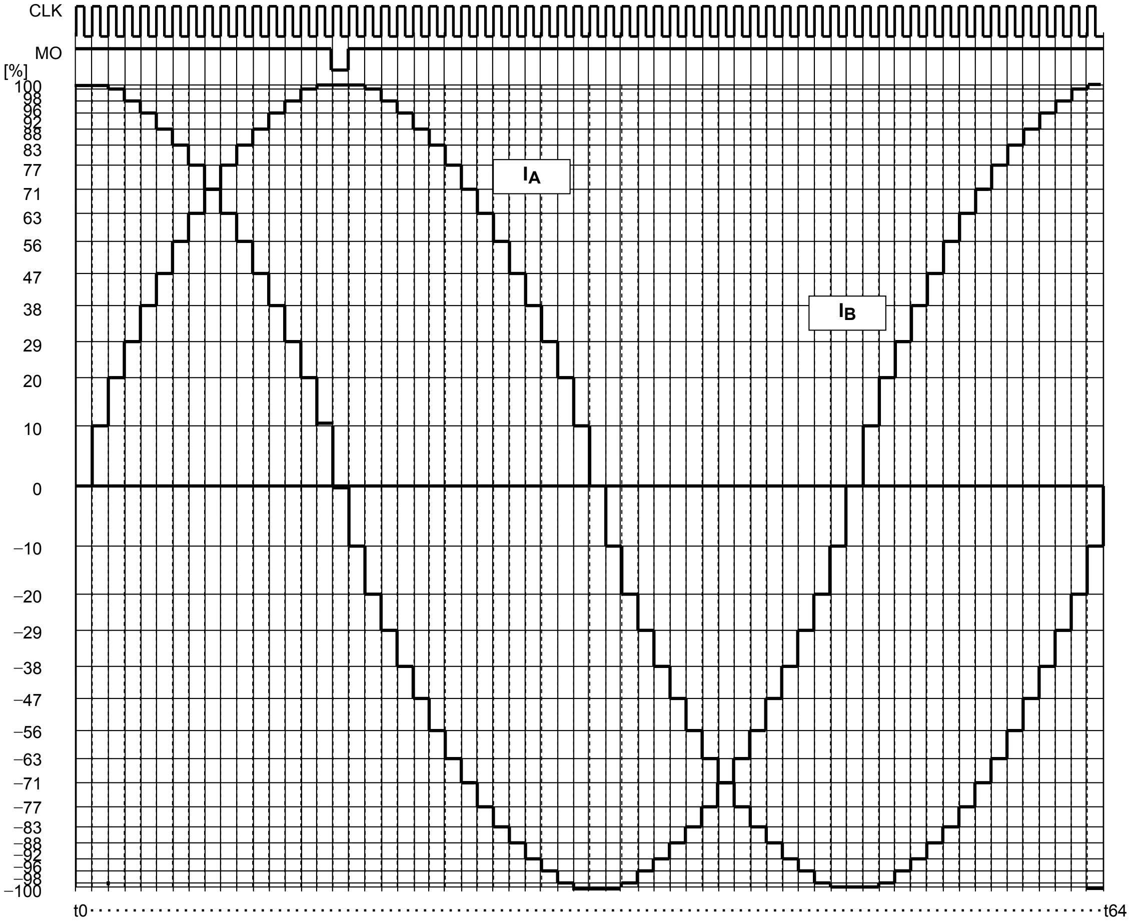
It operates from the initial state after the excitation mode is switched.

1/16-step Excitation Mode (M1: H, M2: H, M3: L, CW Mode)



It operates from the initial state after the excitation mode is switched.

1/16-step Excitation Mode (M1: H, M2: H, M3: L, CCW Mode)



It operates from the initial state after the excitation mode is switched.

Current level

2-phase, 1-2-phase, W1-2-phase, 2W1-2-phase, 4W1-2-phase excitation (unit: %)

Current level（1/16, 1/8, 1/4, 1/2, 1/1 ）

1/16, 1/8, 1/4, 1/2, 1/1	Min.	Typ.	Max.	Unit
θ16	---	100.0	---	%
θ15	95.5	99.5	100.0	
θ14	94.1	98.1	100.0	
θ13	91.7	95.7	99.7	
θ12	88.4	92.4	96.4	
θ11	84.2	88.2	92.2	
θ10	79.1	83.1	87.1	
θ9	73.3	77.3	81.3	
θ8	66.7	70.7	74.7	
θ7	59.4	63.4	67.4	
θ6	51.6	55.6	59.6	
θ5	43.1	47.1	51.1	
θ4	34.3	38.3	42.3	
θ3	25.0	29.0	33.0	
θ2	15.5	19.5	23.5	
θ1	5.8	9.8	13.8	
θ0	---	0.0	---	

Absolute Maximum Ratings (Ta = 25°C)

Characteristic	Symbol	Rating	Unit
Power supply voltage	Vcc	50	V
Output current (per one phase)	I_O (PEAK)	5.0	A
Drain current (ALERT, MO)	I (ALERT) I (MO)	1	mA
Input voltage	V_{IN}	6	V
Power dissipation	P_D	3.2 (Note 1) 40 (Note 2)	W
Operating temperature	T_{opr}	-30 to 85	°C
Storage temperature	T_{stg}	-55 to 150	°C

Note 1: Ta = 25°C, No heatsink

Note 2: Ta = 25°C, with infinite heatsink.

The absolute maximum ratings of a semiconductor device are a set of ratings that must not be exceeded, even for a moment. Do not exceed any of these ratings.
Exceeding the rating (s) may cause the device breakdown, damage or deterioration, and may result injury by explosion or combustion.
Please use the IC within the specified operating ranges.

Operating Range (Ta = −30~85°C)

Characteristic	Symbol	Test Condition	Min	Typ.	Max	Unit
Power supply voltage	Vcc	—	8.0	—	42	V
Output current	I_{OUT}	—	—	—	4.5	A
Input voltage	V_{IN}	—	0	—	5.5	V
	V_{ref}	—	0.3	—	1.95	V
Clock frequency in logical part	f_{CLK}	—	—	—	200	kHz
Chopping frequency	f_{chop}	See page 7.	20	40	60	kHz

Note: Two Vcc terminals should be programmed the same voltage.
The maximum current of the operating range can not be necessarily conducted depending on various conditions because output current is limited by the power dissipation P_D.
Make sure to avoid using the IC in the condition that would cause the temperature to exceed Tj (avg.) =107°C.

The power supply voltage of 42 V and the output current of 4.5 A are the maximum values of operating range. Please design the circuit with enough derating within this range by considering the power supply variation, the external resistance, and the electrical characteristics of the IC. In case of exceeding the power supply voltage of 42 V and the output current of 4.5 A, the IC will not operate normally.

Electrical Characteristics (Ta = 25°C, Vcc = 24 V)

Characteristic		Symbol	Test Condition	Min	Typ.	Max	Unit
Input voltage	High	$V_{IN\ (H)}$	M1, M2, M3, CW/CCW, CLK, RESET, ENABLE, Latch/Auto, TQ	2.0	—	5.5	V
	Low	$V_{IN\ (L)}$		-0.2	—	0.8	
Input hysteresis voltage		V_H		—	400	—	mV
Input current		$I_{IN\ (H)}$	M1, M2, M3, CW/CCW, CLK, RESET, ENABLE, Latch/Auto V_{IN} = 5.0 V	—	50	75	μA
			TQ, V_{IN} = 5.0 V	—	70	105	
		$I_{IN\ (L)}$	M1, M2, M3, CW/CCW, CLK, RESET, ENABLE, Latch/Auto, TQ V_{IN} = 0 V	—	—	1	
Vcc supply current		Icc_1	Output open, RESET: H, ENABLE: H、 M1:L, M2:L, M3:H (1/1-step mode) CLK:L	—	4.2	7	mA
		Icc_2	Output open, RESET: L, ENABLE: L M1:L, M2:L, M3:H (1/1-step mode) CLK:L	—	3.6	7	
		Icc_3	Standby mode (M1:L, M2:L, M3:L)	—	1.8	4	
Vref input circuit	Current limit voltage	V_{NF}	Vref = 3.0 V(Note 1), TQ=H	0.9	1.0	1.1	V
	Input current	$I_{IN}(V_{ref})$	Vref = 3.0 V(Note 1)	—	—	1	μA
	Divider ratio	V_{ref}/V_{NF}	Maximum current: 100%, TQ=H	—	3	—	—
Minimum CLK pulse width		tw_{CLKH}	CLK	2.2	—	—	μs
		tw_{CLKL}					
Output residual voltage		V_{OL} MO	I_{OL} = 1 mA	—	—	0.5	V
		V_{OL} ALERT					
Internal constant voltage		Vreg	External capacitor = 0.1 μF (in standby mode)	4.5	5.0	5.5	V
Chopping frequency		f_{chop}	Rosc=51kΩ	28	40	52	kHz

Note 1: Though Vref of the test condition for pre-shipment is 3.0V, make sure to configure Vref within the operating range which is written in page 26 in driving the motor.

Electrical Characteristics (Ta = 25°C, Vcc = 24 V)

Characteristic		Symbol	Test Condition	Min	Typ.	Max	Unit
Output ON resistor		$Ron_U + Ron_L$	I_{OUT} = 4 A	—	0.4	0.6	Ω
Output transistor switching characteristics		t_r	V_{NF} = 0 V, Output: Open	—	50	—	ns
		t_f		—	500	—	
Output leakage current	Upper side	I_{LH}	Vcc = 50 V	—	—	5	μA
	Lower side	I_{LL}		—	—	5	

Timing Waveforms and Names

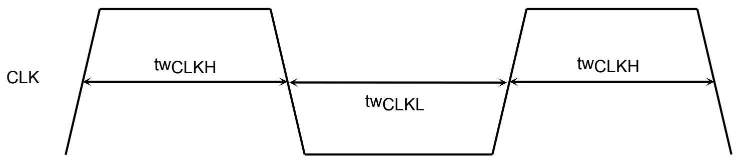


Figure 1 Timing Waveforms and Names

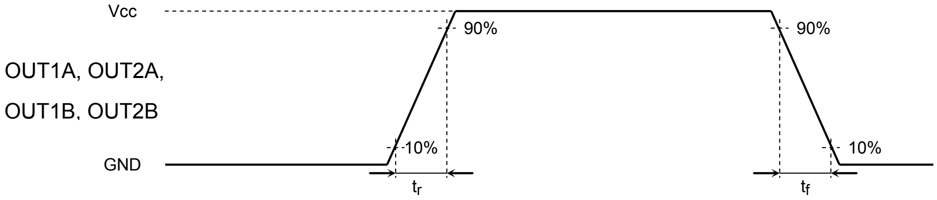
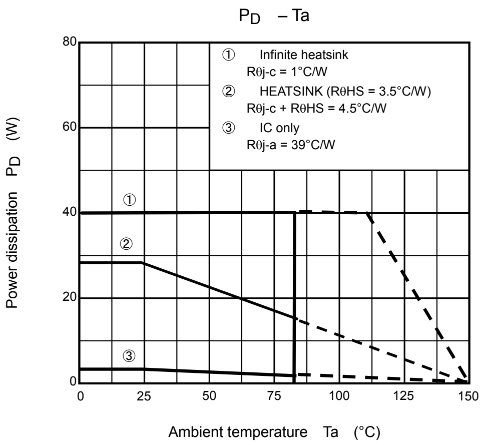


Figure 2 Timing Waveforms and Names

Power Dissipation

TB6600HG



1. How to Turn on the Power

In applying Vcc or shutdown, ENABLE should be Low.
See Example 1(ENABLE = High → RESET = High) and Example 2(RESET = High → ENABLE = High) as follows. In example 1, a motor can start driving from the initial mode.

(1) CLK: Current step proceeds to the next mode with respect to every rising edge of CLK.

(2) ENABLE: It is in Hi-Z state in low level. It is output in high level.

RESET: It is in the initial mode (Phase A=100% and Phase B=0%) in low level.

①ENABLE=Low and RESET=Low: Hi-Z. Internal current setting is in initial mode.

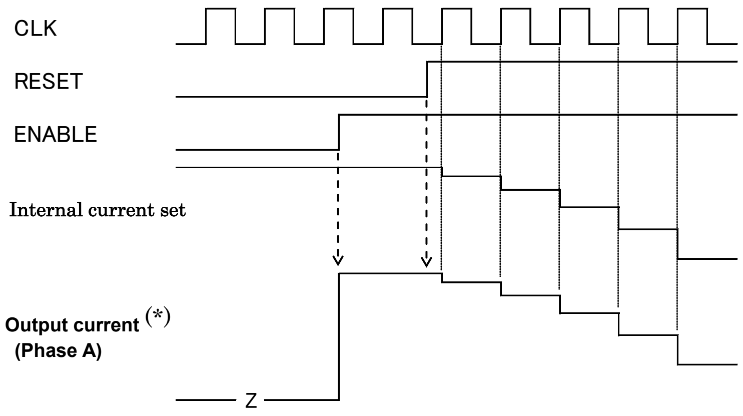
②ENABLE=Low and RESET=High: Hi-Z. Internal current setting proceeds by internal counter.

③ENABLE=High and RESET=Low: Output in the initial mode (Phase A=100% and Phase B=0%).

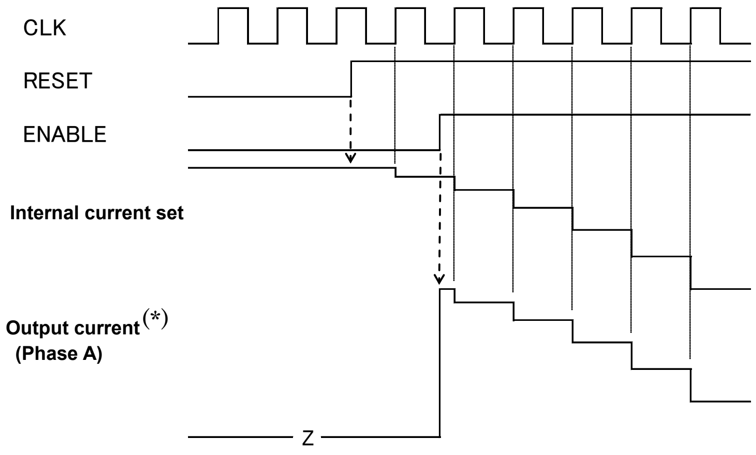
④ENABLE=High and RESET=High: Output at the value which is determined by the internal counter.

<Recommended control input sequence>

(Example 1)

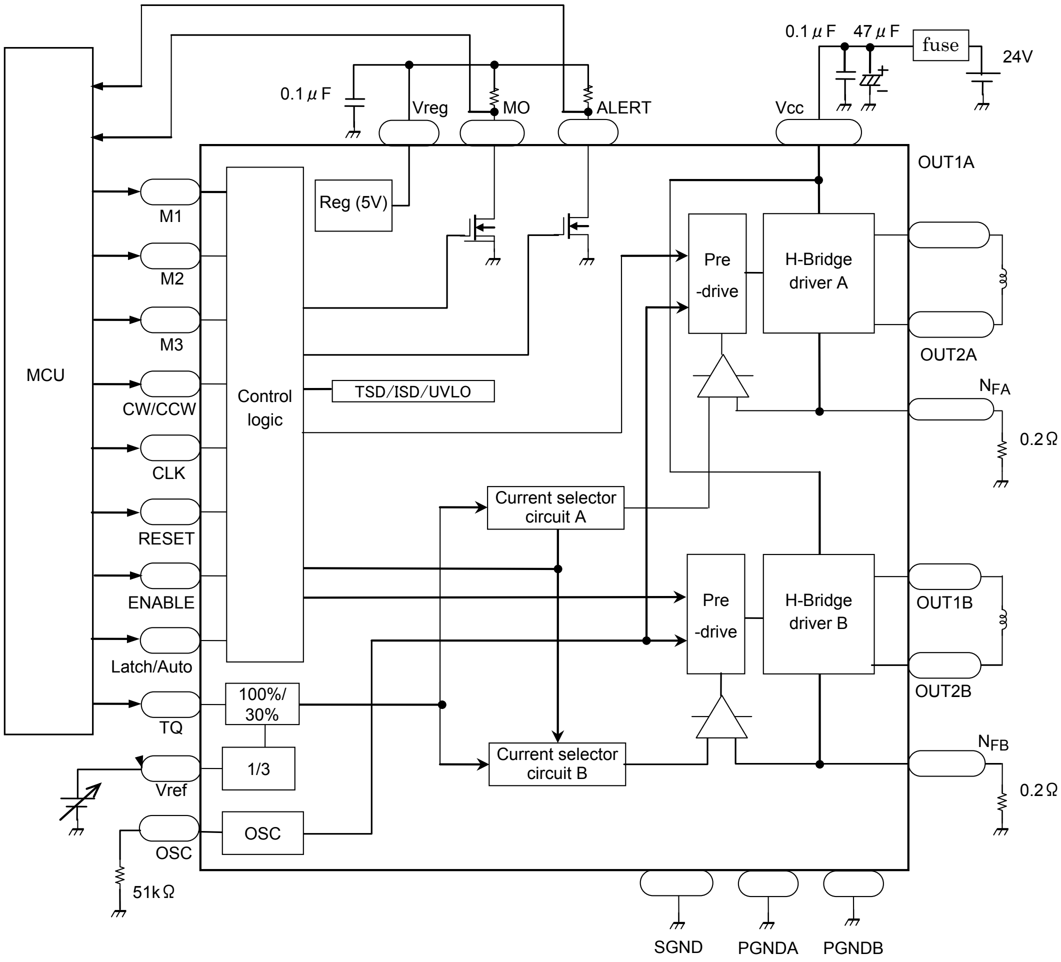


(Example 2)



(* : Output current starts rising at the timing of PWM frequency just after ENABLE pin outputs high.)

Application Circuit



Note 1: Capacitors for the power supply lines should be connected as close to the IC as possible.

Note 2: Current detecting resistances (RNFA and RNFB) should be connected as close to the IC as possible.

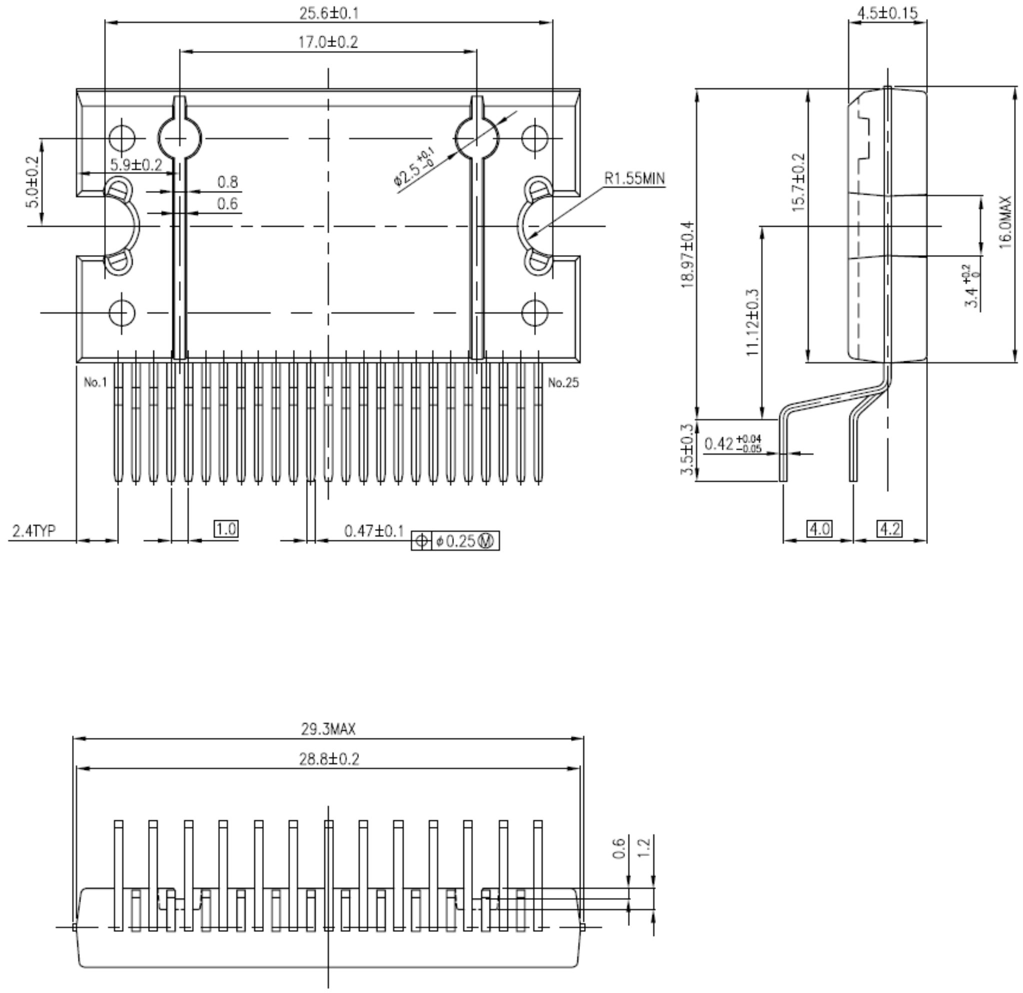
Note 3: Pay attention for wire layout of PCB not to allow GND line to have large common impedance.

Note 4: External capacitor connecting to Vreg should be 0.1μF. Pay attention for the wire between this capacitor and Vreg terminal and the wire between this capacitor and SGND not to be influenced by noise.

Note 5: The IC may not operate normally when large common impedance is existed in GND line or the IC is easily influenced by noise. For example, if the IC operates continuously for a long time under the circumstance of large current and high voltage, the number of clock signals inputted to CLK terminal and that of steps of output current waveform may not proportional. And so, the IC may not operate normally. To avoid this malfunction, make sure to conduct Note.1 to Note.4 and evaluate the IC enough before using the IC.

Package Dimensions

Weight: 7.7 g (typ.)
Unit: mm



Notes on Contents

1. Block Diagrams

Some of the functional blocks, circuits, or constants in the block diagram may be omitted or simplified for explanatory purposes.

2. Equivalent Circuits

The equivalent circuit diagrams may be simplified or some parts of them may be omitted for explanatory purposes.

3. Timing Charts

Timing charts may be simplified for explanatory purposes.

4. Application Circuits

The application circuits shown in this document are provided for reference purposes only. Thorough evaluation is required, especially at the mass production design stage.

Toshiba does not grant any license to any industrial property rights by providing these examples of application circuits.

5. Test Circuits

Components in the test circuits are used only to obtain and confirm the device characteristics. These components and circuits are not guaranteed to prevent malfunction or failure from occurring in the application equipment.

IC Usage Considerations

Notes on handling of ICs

[1] The absolute maximum ratings of a semiconductor device are a set of ratings that must not be exceeded, even for a moment. Do not exceed any of these ratings.

Exceeding the rating(s) may cause the device breakdown, damage or deterioration, and may result injury by explosion or combustion.

[2] Use an appropriate power supply fuse to ensure that a large current does not continuously flow in case of over current and/or IC failure. The IC will fully break down when used under conditions that exceed its absolute maximum ratings, when the wiring is routed improperly or when an abnormal pulse noise occurs from the wiring or load, causing a large current to continuously flow and the breakdown can lead smoke or ignition. To minimize the effects of the flow of a large current in case of breakdown, appropriate settings, such as fuse capacity, fusing time and insertion circuit location, are required.

[3] If your design includes an inductive load such as a motor coil, incorporate a protection circuit into the design to prevent device malfunction or breakdown caused by the current resulting from the inrush current at power ON or the negative current resulting from the back electromotive force at power OFF. IC breakdown may cause injury, smoke or ignition.

Use a stable power supply with ICs with built-in protection functions. If the power supply is unstable, the protection function may not operate, causing IC breakdown. IC breakdown may cause injury, smoke or ignition.

[4] Do not insert devices in the wrong orientation or incorrectly.

Make sure that the positive and negative terminals of power supplies are connected properly.

Otherwise, the current or power consumption may exceed the absolute maximum rating, and exceeding the rating(s) may cause the device breakdown, damage or deterioration, and may result injury by explosion or combustion.

In addition, do not use any device that is applied the current with inserting in the wrong orientation or incorrectly even just one time.

Points to remember on handling of ICs

(1) Over current Detection Circuit

Over current detection circuits (referred to as current limiter circuits) do not necessarily protect ICs under all circumstances. If the over current detection circuits operate against the over current, clear the over current status immediately.

Depending on the method of use and usage conditions, such as exceeding absolute maximum ratings can cause the over current detection circuit to not operate properly or IC breakdown before operation. In addition, depending on the method of use and usage conditions, if over current continues to flow for a long time after operation, the IC may generate heat resulting in breakdown.

(2) Thermal Shutdown Circuit

Thermal shutdown circuits do not necessarily protect ICs under all circumstances. If the thermal shutdown circuits operate against the over temperature, clear the heat generation status immediately.

Depending on the method of use and usage conditions, such as exceeding absolute maximum ratings can cause the thermal shutdown circuit to not operate properly or IC breakdown before operation.

(3) Heat Radiation Design

In using an IC with large current flow such as power amp, regulator or driver, please design the device so that heat is appropriately radiated, not to exceed the specified junction temperature (T_j) at any time and condition. These ICs generate heat even during normal use. An inadequate IC heat radiation design can lead to decrease in IC life, deterioration of IC characteristics or IC breakdown. In addition, please design the device taking into considerate the effect of IC heat radiation with peripheral components.

(4) Back-EMF

When a motor rotates in the reverse direction, stops or slows down abruptly, a current flow back to the motor's power supply due to the effect of back-EMF. If the current sink capability of the power supply is small, the device's motor power supply and output pins might be exposed to conditions beyond absolute maximum ratings. To avoid this problem, take the effect of back-EMF into consideration in system design.

(5) Short-circuiting between outputs, air contamination faults, faults due to improper grounding, short-circuiting between contiguous pins

Utmost care is necessary in the design of the power supply lines, GND lines, and output lines since the IC may be destroyed by short-circuiting between outputs, air contamination faults, or faults due to improper grounding, or by short-circuiting between contiguous pins. They may destroy not only the IC but also peripheral parts and may contribute to injuries for users. Over current may continue to flow in the IC because of this destruction and cause smoke or ignition of the IC. Expect the volume of this over current and add an appropriate power supply fuse in order to minimize the effects of the over current. Capacity of the fuse, fusing time, and the inserting position in the circuit should be configured suitably.

RESTRICTIONS ON PRODUCT USE

- Toshiba Corporation, and its subsidiaries and affiliates (collectively "TOSHIBA"), reserve the right to make changes to the information in this document, and related hardware, software and systems (collectively "Product") without notice.

- This document and any information herein may not be reproduced without prior written permission from TOSHIBA. Even with TOSHIBA's written permission, reproduction is permissible only if reproduction is without alteration/omission.

- Though TOSHIBA works continually to improve Product's quality and reliability, Product can malfunction or fail. Customers are responsible for complying with safety standards and for providing adequate designs and safeguards for their hardware, software and systems which minimize risk and avoid situations in which a malfunction or failure of Product could cause loss of human life, bodily injury or damage to property, including data loss or corruption. Before customers use the Product, create designs including the Product, or incorporate the Product into their own applications, customers must also refer to and comply with (a) the latest versions of all relevant TOSHIBA information, including without limitation, this document, the specifications, the data sheets and application notes for Product and the precautions and conditions set forth in the "TOSHIBA Semiconductor Reliability Handbook" and (b) the instructions for the application with which the Product will be used with or for. Customers are solely responsible for all aspects of their own product design or applications, including but not limited to (a) determining the appropriateness of the use of this Product in such design or applications; (b) evaluating and determining the applicability of any information contained in this document, or in charts, diagrams, programs, algorithms, sample application circuits, or any other referenced documents; and (c) validating all operating parameters for such designs and applications. **TOSHIBA ASSUMES NO LIABILITY FOR CUSTOMERS' PRODUCT DESIGN OR APPLICATIONS.**

- **PRODUCT IS NEITHER INTENDED NOR WARRANTED FOR USE IN EQUIPMENTS OR SYSTEMS THAT REQUIRE EXTRAORDINARILY HIGH LEVELS OF QUALITY AND/OR RELIABILITY, AND/OR A MALFUNCTION OR FAILURE OF WHICH MAY CAUSE LOSS OF HUMAN LIFE, BODILY INJURY, SERIOUS PROPERTY DAMAGE AND/OR SERIOUS PUBLIC IMPACT ("UNINTENDED USE").** Except for specific applications as expressly stated in this document, Unintended Use includes, without limitation, equipment used in nuclear facilities, equipment used in the aerospace industry, medical equipment, equipment used for automobiles, trains, ships and other transportation, traffic signaling equipment, equipment used to control combustions or explosions, safety devices, elevators and escalators, devices related to electric power, and equipment used in finance-related fields. **IF YOU USE PRODUCT FOR UNINTENDED USE, TOSHIBA ASSUMES NO LIABILITY FOR PRODUCT.** For details, please contact your TOSHIBA sales representative.

- Do not disassemble, analyze, reverse-engineer, alter, modify, translate or copy Product, whether in whole or in part.

- Product shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable laws or regulations.

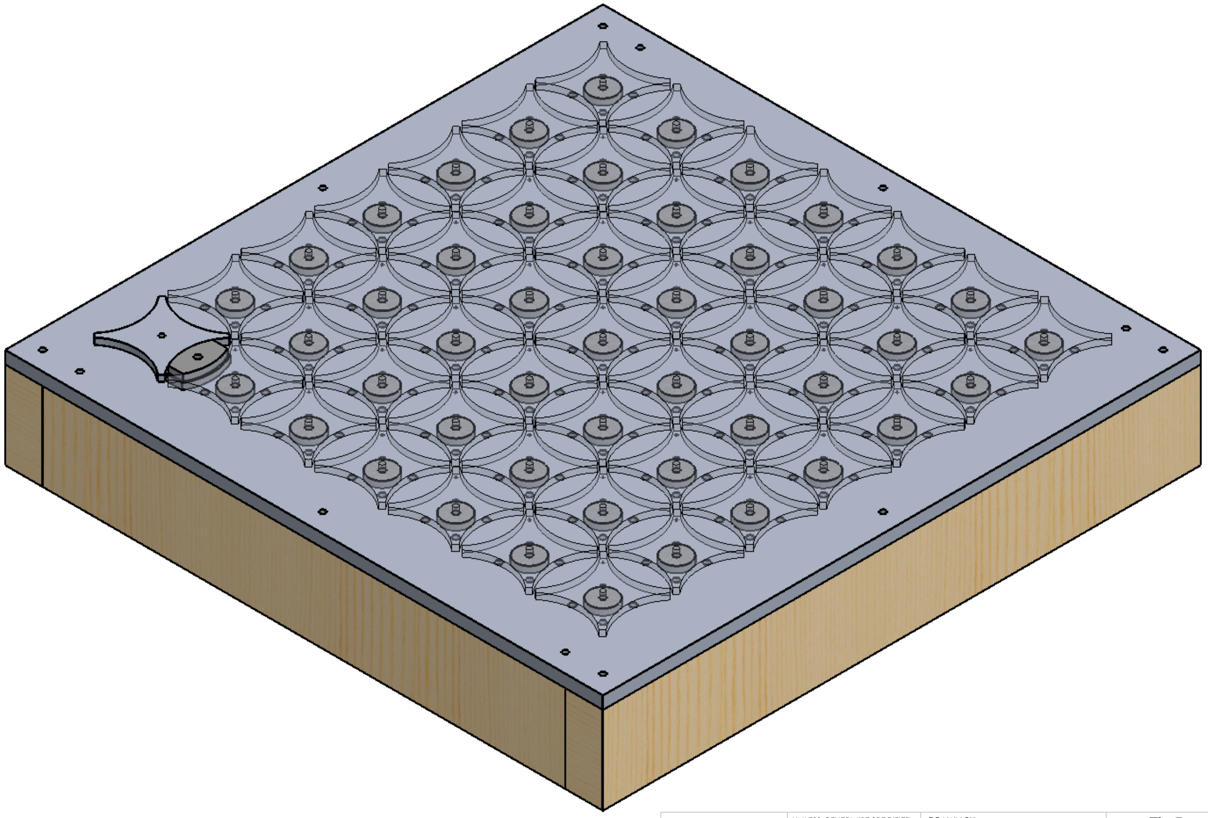
- The information contained herein is presented only as guidance for Product use. No responsibility is assumed by TOSHIBA for any infringement of patents or any other intellectual property rights of third parties that may result from the use of Product. No license to any intellectual property right is granted by this document, whether express or implied, by estoppel or otherwise.

- **ABSENT A WRITTEN SIGNED AGREEMENT, EXCEPT AS PROVIDED IN THE RELEVANT TERMS AND CONDITIONS OF SALE FOR PRODUCT, AND TO THE MAXIMUM EXTENT ALLOWABLE BY LAW, TOSHIBA (1) ASSUMES NO LIABILITY WHATSOEVER, INCLUDING WITHOUT LIMITATION, INDIRECT, CONSEQUENTIAL, SPECIAL, OR INCIDENTAL DAMAGES OR LOSS, INCLUDING WITHOUT LIMITATION, LOSS OF PROFITS, LOSS OF OPPORTUNITIES, BUSINESS INTERRUPTION AND LOSS OF DATA, AND (2) DISCLAIMS ANY AND ALL EXPRESS OR IMPLIED WARRANTIES AND CONDITIONS RELATED TO SALE, USE OF PRODUCT, OR INFORMATION, INCLUDING WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY OF INFORMATION, OR NONINFRINGEMENT.**

- Do not use or otherwise make available Product or related software or technology for any military purposes, including without limitation, for the design, development, use, stockpiling or manufacturing of nuclear, chemical, or biological weapons or missile technology products (mass destruction weapons). Product and related software and technology may be controlled under the applicable export laws and regulations including, without limitation, the Japanese Foreign Exchange and Foreign Trade Law and the U.S. Export Administration Regulations. Export and re-export of Product or related software or technology are strictly prohibited except in compliance with all applicable export laws and regulations.

- Please contact your TOSHIBA sales representative for details as to environmental matters such as the RoHS compatibility of Product. Please use Product in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. **TOSHIBA ASSUMES NO LIABILITY FOR DAMAGES OR LOSSES OCCURRING AS A RESULT OF NONCOMPLIANCE WITH APPLICABLE LAWS AND REGULATIONS.**

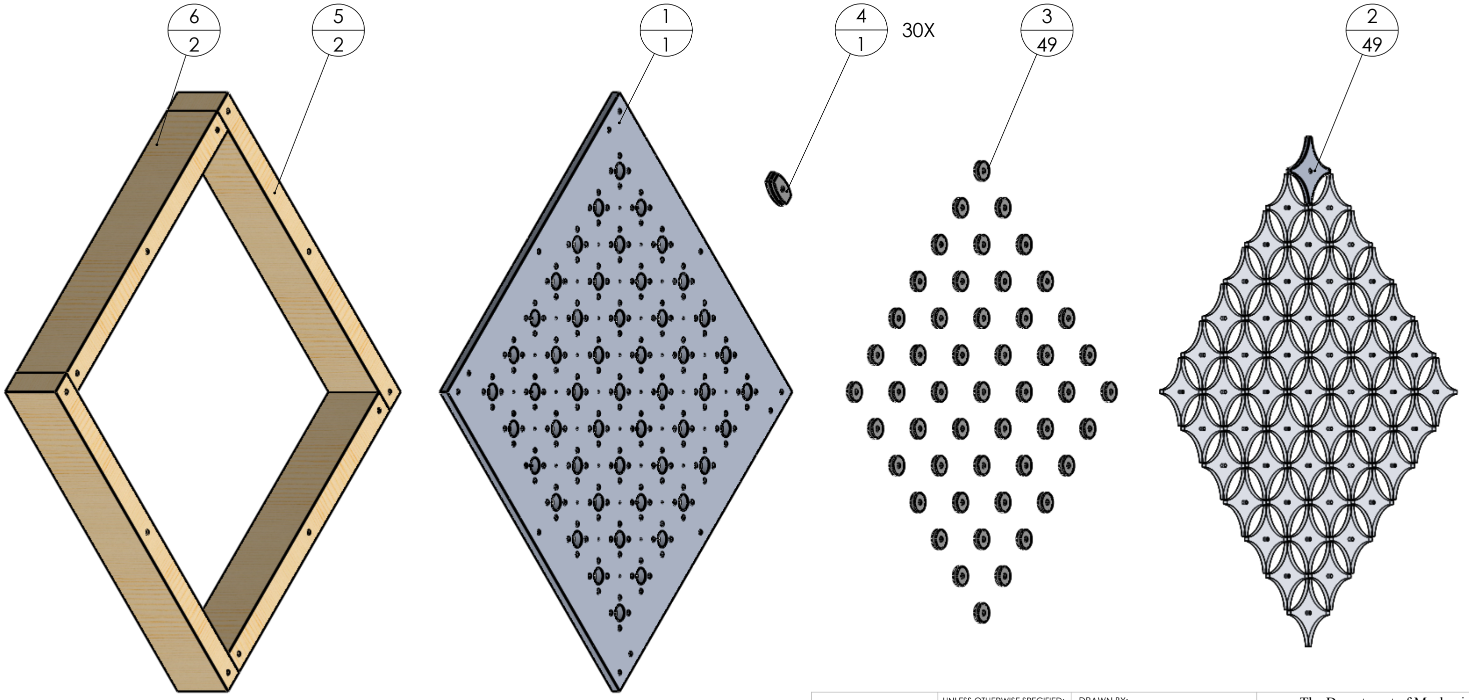
A.2 3D Rotary Braider - Engineering Drawings



SOLIDWORKS Student Edition.
For Academic Use Only.

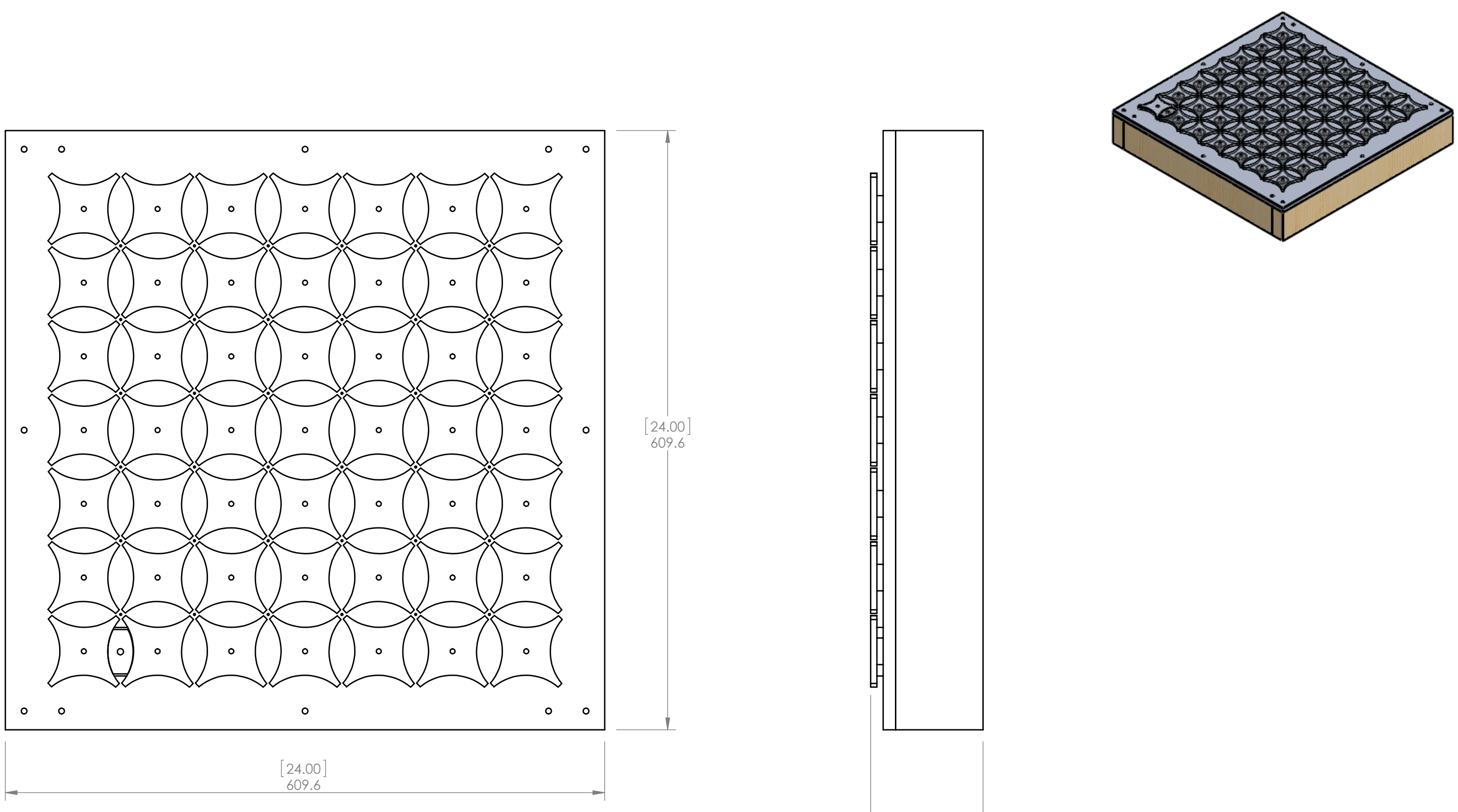
Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH µm 0.6 DO NOT SCALE DRAWING	DRAWN BY: **Daniel Aldrich**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments:		Lab Day	TITLE: Full Assembly
		SM By **Daniel Aldrich**	
		TA Initials	SIZE **B** Assignment Number REV
MATERIAL:		draldric Wednesday, August 31, 2016 9:18:38 AM Wednesday, June 01, 2016 8:16:03 AM	
FILE NAME: Full_Assembly			SCALE: 1:3 Mass: SHEET 1 OF 8

ITEM NO.	PART NUMBER	Material	SW-Author(Author)	QTY.
1	Board	6061 Alloy	Daniel Aldric	1
2	Movement	6061 Alloy	Daniel Aldrich	49
3	Spacer	PE High Density	Daniel Aldrich	49
4	Carrier	13.38	Daniel Aldric	1
5	Box_Long	Pine	Daniel Aldrich	2
6	Box_Short	Pine	Daniel Aldrich	2



SOLIDWORKS Student Edition.
For Academic Use Only.

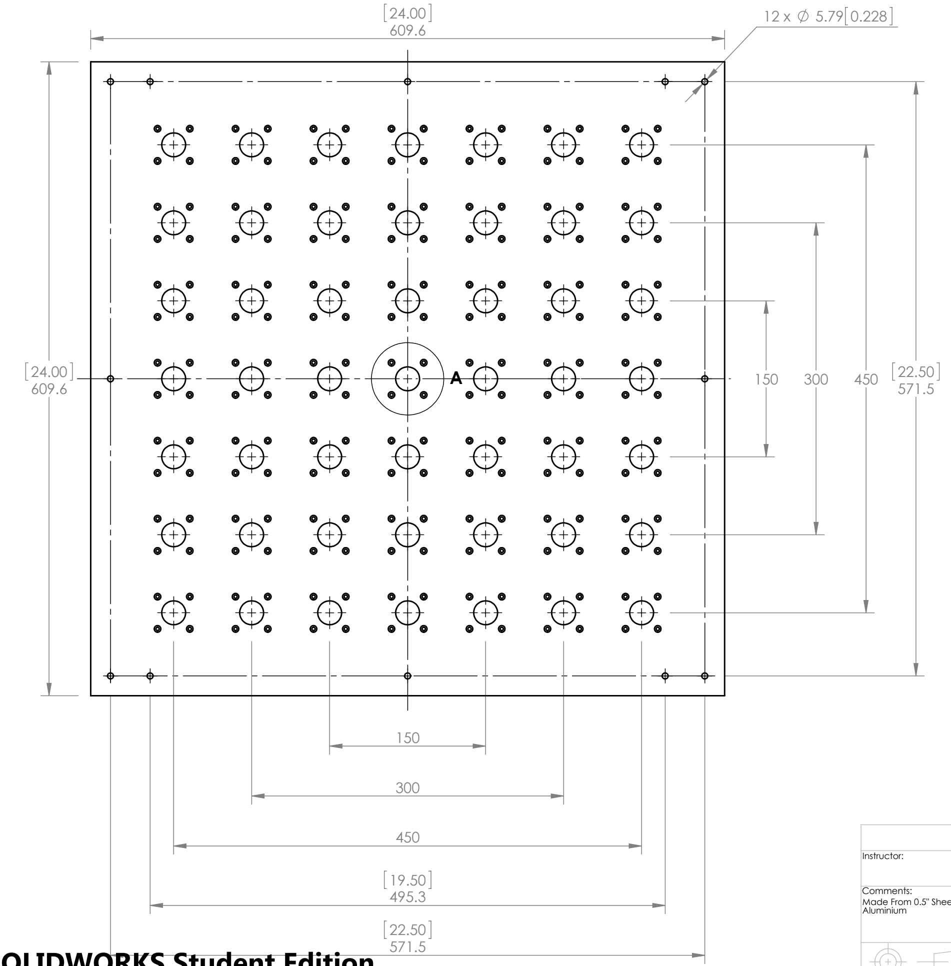
Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH µm 0.6 DO NOT SCALE DRAWING	DRAWN BY: Daniel Aldrich Lab Day SM By Daniel Aldrich TA Initials	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA TITLE: Full Assembly
Comments:		draldric Wednesday, August 31, 2016 9:18:38 AM Wednesday, June 01, 2016 8:16:03 AM	SIZE B | Assignment Number | REV
MATERIAL:			
FILE NAME: Full_Assembly			SCALE: 1:6 | Mass: | SHEET 2 OF 8

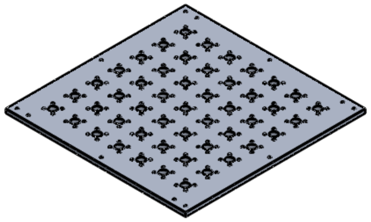


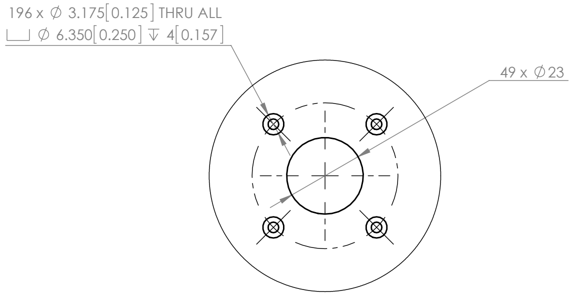
SOLIDWORKS Student Edition.
For Academic Use Only.

Instructor:	UNLESS OTHERWISE SPECIFIED:	DRAWN BY: **Daniel Aldrich**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments:	DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025	Lab Day	TITLE: Full Assembly
	SURFACE FINISH 0.6 µm	SM By **Daniel Aldrich**	
	DO NOT SCALE DRAWING	TA Initials	SIZE B · Assignment Number · REV
MATERIAL:		draldric Wednesday, August 31, 2016 9:18:38 AM Wednesday, June 01, 2016 8:16:03 AM	
FILE NAME: Full_Assembly			SCALE: 1:4 · Mass: · SHEET 3 OF 8

150







DETAIL A
SCALE 4 : 5

SOLIDWORKS Student Edition.
For Academic Use Only.

Instructor:

Comments:
Made From 0.5" Sheet
Aluminium

MATERIAL:
6061 Alloy

FILE NAME:
Board

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
LINEAR
X = ± 0.5
X.X = ± 0.1
X.XX = ± 0.025

SURFACE FINISH 0.6
µm

DO NOT SCALE DRAWING

DRAWN BY:
Daniel Aldrich

Lab Day

SM By **Daniel Aldric**

TA Initials

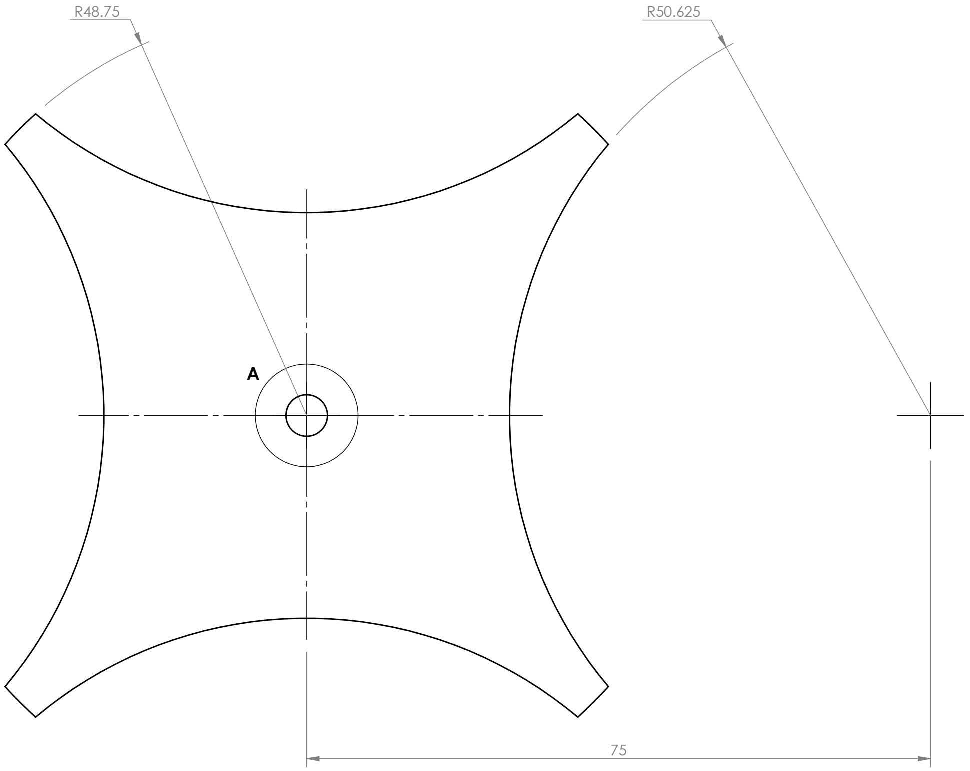
draldric
Wednesday, August 31, 2016 9:26:21 AM
Wednesday, June 01, 2016 8:10:40 AM

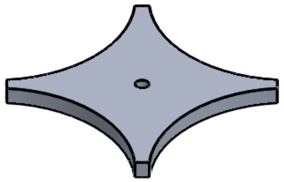
The Department of Mechanical Engineering
UNIVERSITY OF ALBERTA

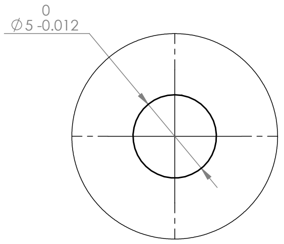
TITLE:
Base Plate

SIZE **B**
Assignment Number
REV

SCALE: 1:4
Mass: 11930.17
SHEET 4 OF 8



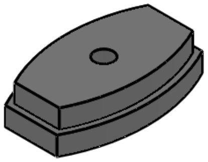


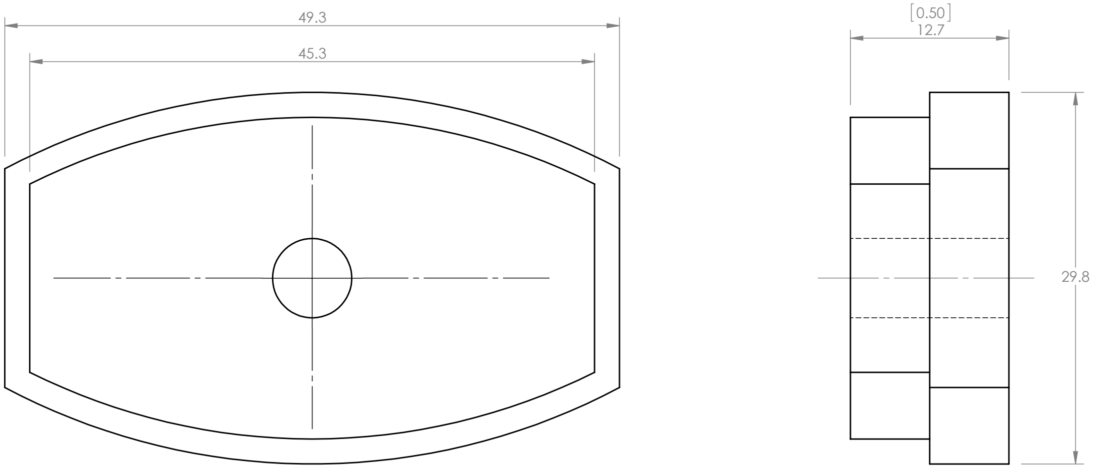


DETAIL A
SCALE 4 : 1

SOLIDWORKS Student Edition.
For Academic Use Only.

Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH 0.6 µm DO NOT SCALE DRAWING	DRAWN BY: **Daniel Aldrich**		The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Made from 0.25" Aluminium Sheet		Lab Day		TITLE: Spindle
		SM By	**Daniel Aldrich**	
		TA Initials		SIZE **B** / Assignment Number / REV
MATERIAL: 6061 Alloy		draldric Wednesday, August 31, 2016 9:10:59 AM Wednesday, June 01, 2016 7:20:54 AM		
FILE NAME: Movement				SCALE: 2:1 / Mass: 53.04 / SHEET 5 OF 8





Instructor:

Comments:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
LINEAR
X = ± 0.5
X.X = ± 0.1
X.XX = ± 0.025

SURFACE FINISH 0.6 µm

DO NOT SCALE DRAWING

MATERIAL:
13.38

FILE NAME:
Carrier

DRAWN BY:
Daniel Aldrich

Lab Day

SM By **Daniel Aldric**

TA Initials

draldric
Wednesday, August 31, 2016 9:00:09 AM
Monday, June 06, 2016 7:44:01 AM

The Department of Mechanical Engineering
UNIVERSITY OF ALBERTA

TITLE:
Carrier

SIZE **B**

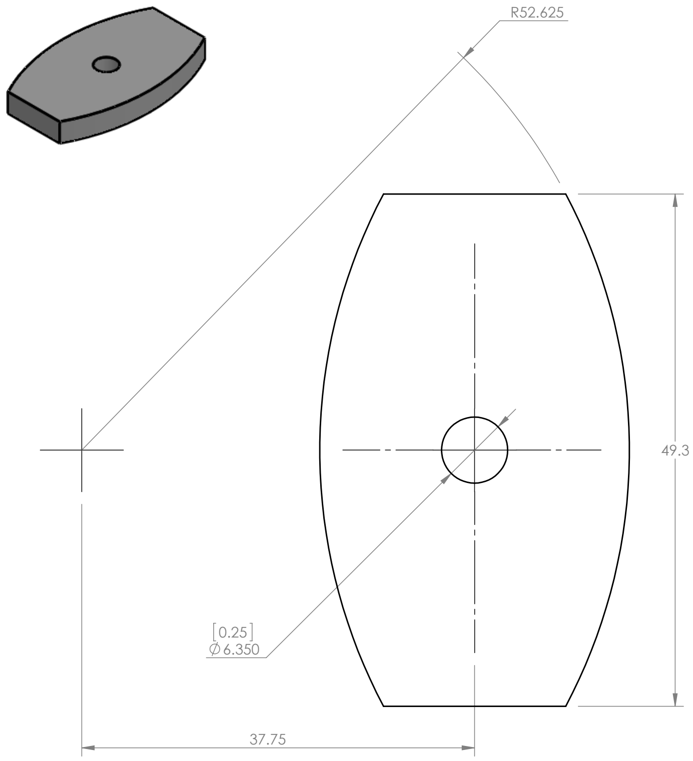
Assignment Number

REV

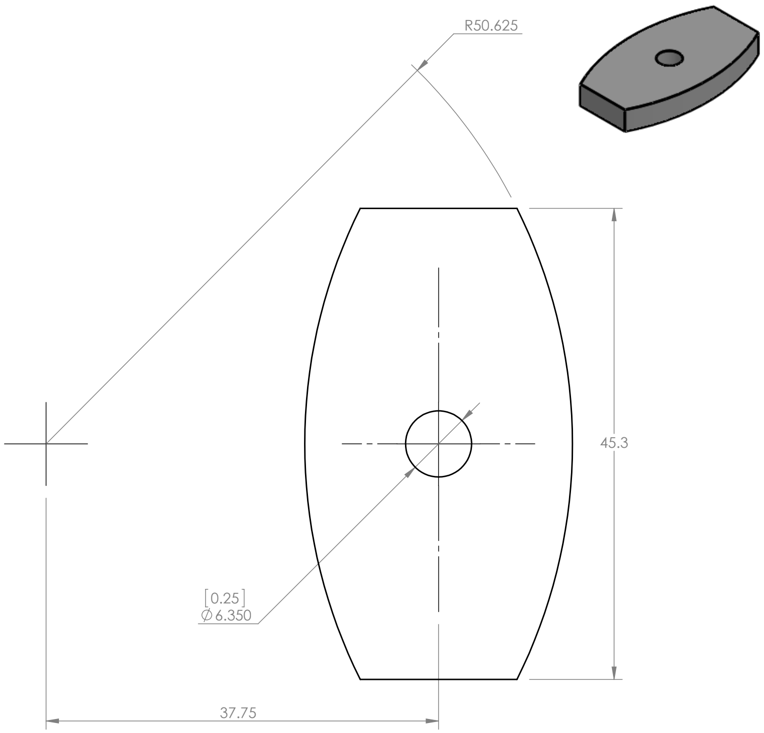
SCALE: 3:1

Mass: 13.38

SHEET 6 OF 8



Carrier (Bottom)



Carrier (Top)

Instructor:

Comments:
Made From 0.25" HDPE.

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MM
TOLERANCES:
ANGULAR: ± 0.5°
LINEAR
X = ± 0.5
X.X = ± 0.1
X.XX = ± 0.025
SURFACE FINISH 0.6 µm
DO NOT SCALE DRAWING

MATERIAL:
PE High Density

FILE NAME:
Carrier_Base

DRAWN BY:
Daniel Aldrich

Lab Day

SM By **Daniel Aldrich**

TA Initials

draldric
Wednesday, August 31, 2016 9:17:36 AM
Monday, June 06, 2016 6:36:07 AM

The Department of Mechanical Engineering
UNIVERSITY OF ALBERTA

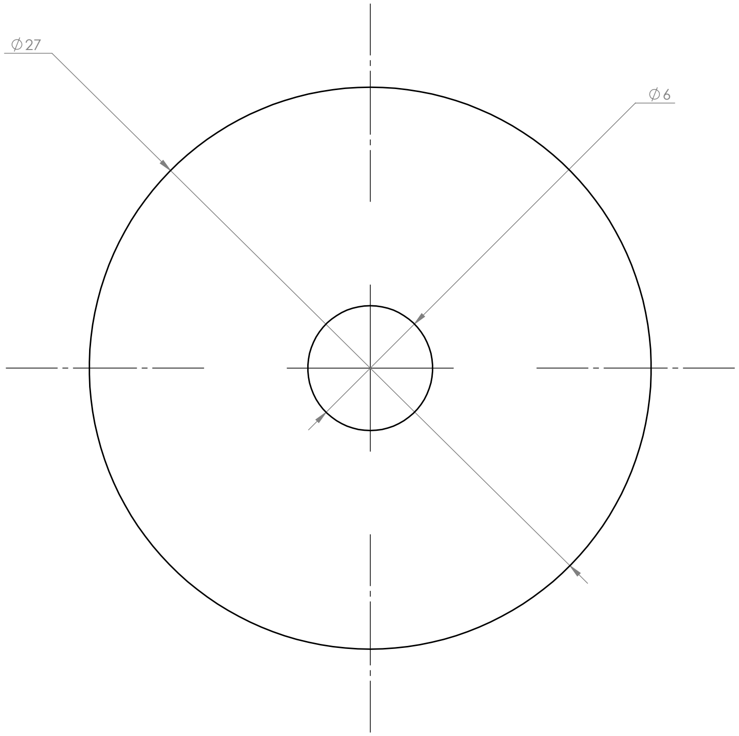
TITLE:
Carrier (Base)

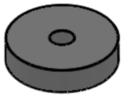
SIZE **B** | Assignment Number | REV

SCALE: 5:2 | Mass: 7.48 | SHEET 7 OF 8

8 7 6 5 4 3 2 1

D





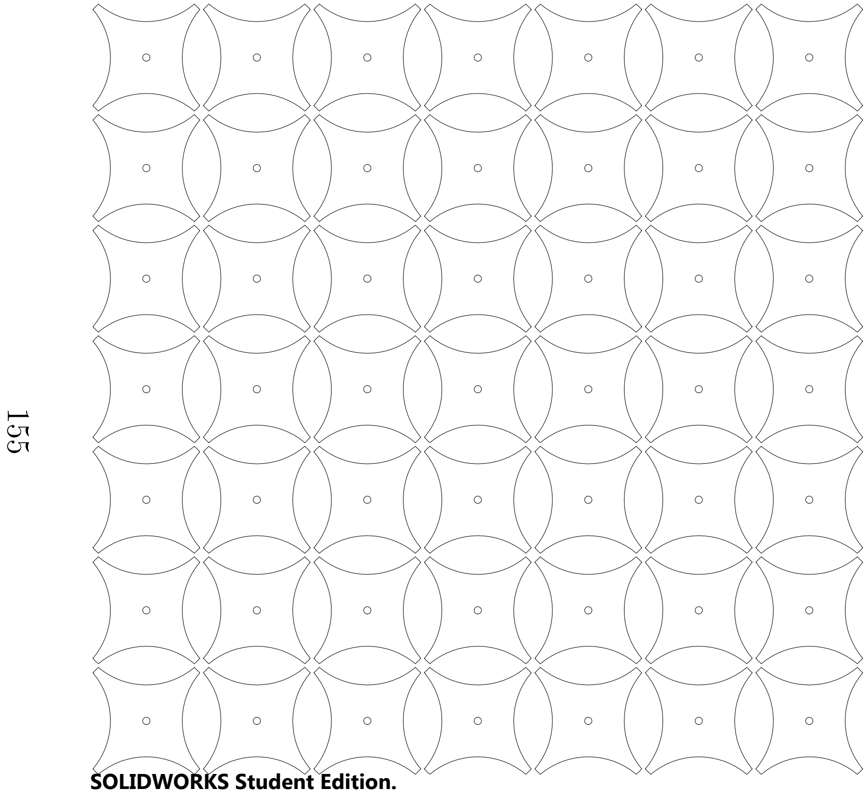
C

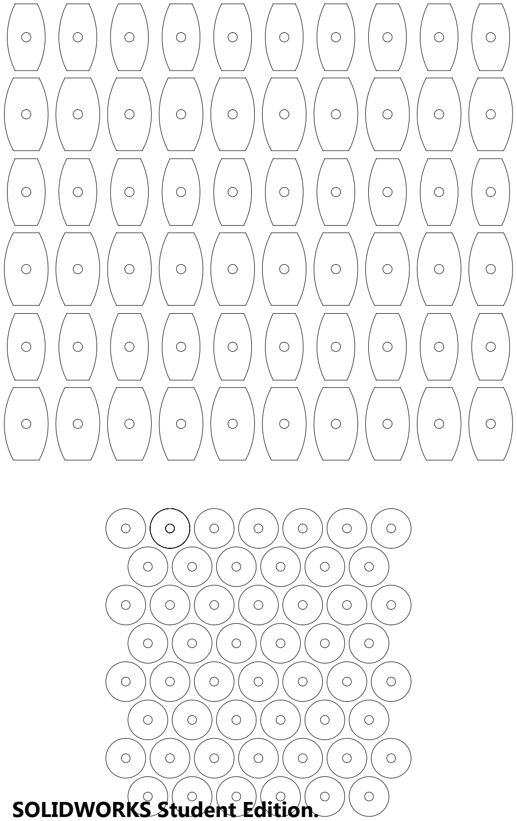
B

A

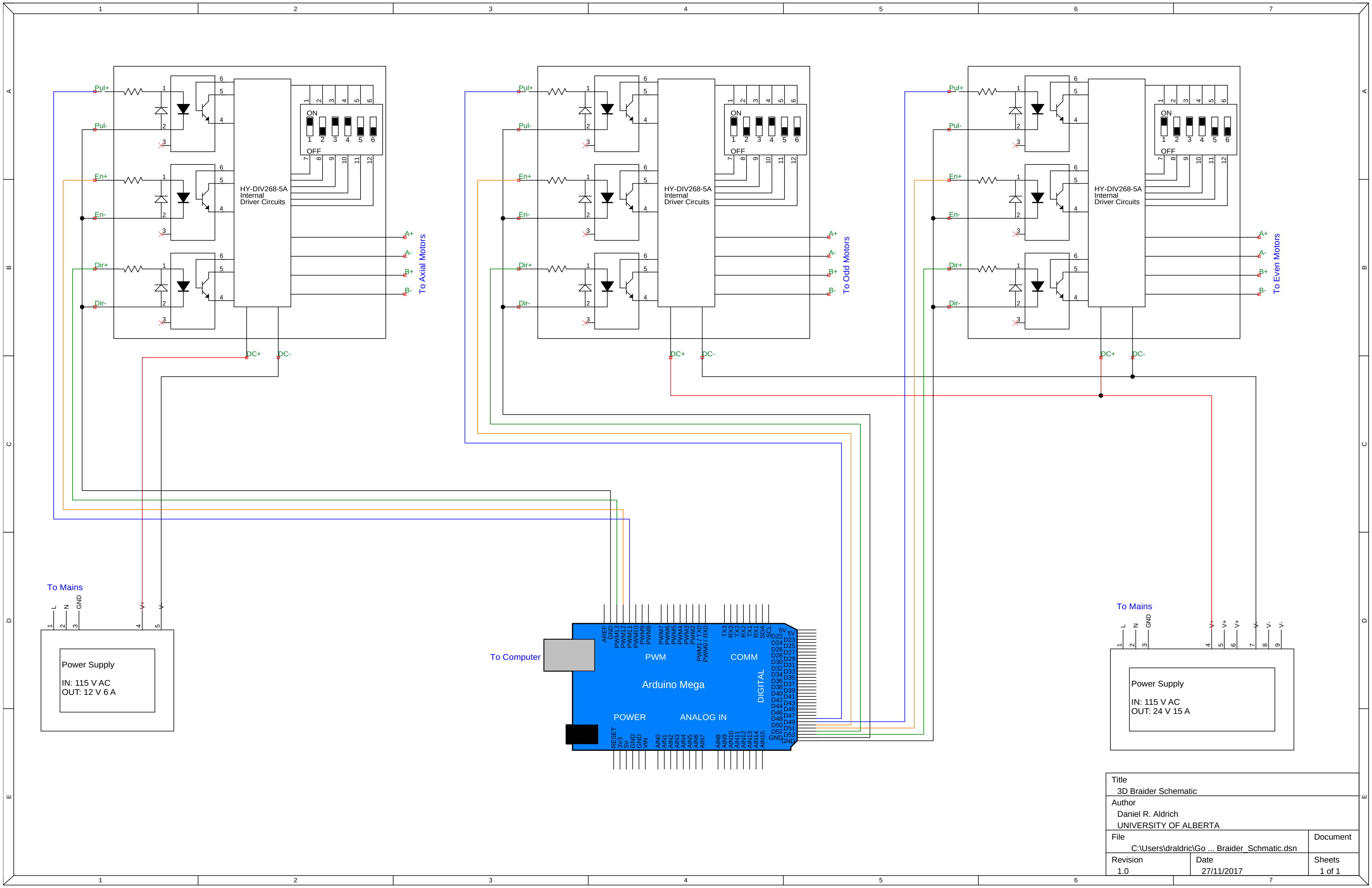
Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH 0.6 µm DO NOT SCALE DRAWING	DRAWN BY: **Daniel Aldrich**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Made From 0.25" HDPE		Lab Day	TITLE: Spacer
		SM By **Daniel Aldrich**	
		TA Initials	SIZE **B** Assignment Number REV
MATERIAL: PE High Density		draldric Wednesday, August 31, 2016 9:18:19 AM Monday, July 18, 2016 1:13:30 PM	
FILE NAME: Spacer			SCALE: 5:1 Mass: 3.29 SHEET 8 OF 8

8 7 6 5 4 3 2 1


155



A.3 3D Rotary Braider - Electrical Schematic



Title
3D Braider Schematic
Author
Daniel R. Aldrich
UNIVERSITY OF ALBERTA
File
C:\Users\draldric\Go ... Braider_Schmatic.dsn
Document
Revision
1.0
Date
27/11/2017
Sheets
1 of 1

Appendix B: 3D Rotary Braider - Code

Listing B.1: User interface for controlling the creation of 3D Braided Samples.

```
classdef MAGIC < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                     matlab.ui.Figure
        PROGRESSGaugeLabel           matlab.ui.control.Label
        PROGRESSGauge                matlab.ui.control.LinearGauge
        BraidCyclesSpinnerLabel      matlab.ui.control.Label
        BraidCyclesSpinner           matlab.ui.control.Spinner
        CONNECTIONPanel              matlab.ui.container.Panel
        CONNECTButton                matlab.ui.control.StateButton
        COMPORTDropDownLabel         matlab.ui.control.Label
        COMPORTDropDown              matlab.ui.control.DropDown
        CONNECTIONDisplay            matlab.ui.control.EditField
        StartButton                  matlab.ui.control.Button
    end


    properties (Access = private)
        serverObject
        comPort = 8;
        baudRate = 115000;
        errorCatch
        propertiesfID
        motorsetData
        evenMotor = 1;
        oddMotor = 2;
    end

    methods (Static)
        function generateMachinePROPERTIES()
            fID = fopen('Machine.PROPERTIES','W');
            fprintf(fID,[...
'#+------------------------------------------------------------------------+',...
'\n#|                                                                        |',...
'\n#| FILENAME : Machine_PROPERTIES                          VERSION : 1.0.0 |',...
'\n#|                                                                        |',...
'\n#| TITLE : Braiding Machine Propties              AUTHOR : Daniel Aldrich |',...
'\n#|                                                                        |',...
'\n#+------------------------------------------------------------------------+',...
'\n#|                                                                        |',...
'\n#| DEPENDENT FILES :                                                      |',...
'\n#|       <none>                                                           |',...
'\n#|                                                                        |',...
'\n#| DESCRIPTION :                                                          |',...
'\n#|       <none>                                                           |',...
'\n#|                                                                        |',...
'\n#| PUBLIC FUNCTIONS :                                                     |',...
'\n#|        <none>                                                           |',...
'\n#|                                                                         |',...
'\n#| NOTES :                                                                 |',...
'\n#|        <none>                                                           |',...
'\n#|                                                                         |',...
'\n#| COPYRIGHT :                                                             |',...
'\n#|        Copyright (c) 2017 Daniel Aldrich                                |',...
'\n#|                                                                         |',...
'\n#|        Permission is hereby granted, free of charge, to any person      |',...
'\n#|        obtaining a copy of this software and associated documentation  |',...
'\n#|        files (the "Software"), to deal in the Software without          |',...
'\n#|        restriction, including without limitation the rights to use,     |',...
'\n#|        copy, modify, merge, publish, distribute, sublicense, and/or      |',...
'\n#|        sell copies of the Software, and to permit persons to whom the   |',...
'\n#|        Software is furnished to do so, subject to the following         |',...
'\n#|        conditions:                                                      |',...
'\n#|                                                                         |',...
'\n#|        The above copyright notice and this permission notice shall be   |',...
'\n#|        included in all copies or substantial portions of the Software.  |',...
'\n#|                                                                         |',...
'\n#|        THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  |',...
'\n#|        EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES  |',...
'\n#|        OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND         |',...
'\n#|        NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT      |',...
'\n#|        HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,     |',...
'\n#|        WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING     |',...
'\n#|        FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR    |',...
'\n#|        OTHER DEALINGS IN THE SOFTWARE.                                  |',...
'\n#|                                                                         |',...
'\n#| CHANGES :                                                               |',...
'\n#|        <none>                                                           |',...
'\n#|                                                                         |',...
'\n#+-------------------------------------------------------------------------+',...
                '\n\n# Even Motor Setup',...
                '\nMotor Set       = even',...
                '\nNumber of Steps = 200',...
                '\nDirection Pin   = 3',...
                '\nEnable Pin      = 4',...
                '\nStep Pin        = 2',...
                '\n\n# Odd Motor Setup',...
                '\nMotor Set       = odd',...
                '\nNumber of Steps = 200',...
                '\nDirection Pin   = 6',...
                '\nEnable Pin      = 7',....
                '\nStep Pin        = 5\n']);
            fclose(fID);
        end
    end

    methods (Access = private)

        function attemptDisconnect(app, ~)
            app.CONNECTButton.ValueChangedFcn =...
                createCallbackFcn(app, @attemptConnection, true);
            app.CONNECTButton.BackgroundColor = [0,1,0];
            app.CONNECTButton.Text = 'CONNECT';
            instrreset;
        end

        function readMachinePROPERTIES(app,fID)
            while ~feof(fID)
                tempLine = fgetl(fID);
                if ~isempty(tempLine) && ~strcmp(tempLine(1),'#') %If not a Comment
                    temp = strsplit(tempLine,'=');
                    key = strtrim(temp{1});
                    value = strtrim(temp{2});
                    switch key
                        case 'Motor Set'
```

```
                            motorSet = value;
                        case 'Number of Steps'
                            property = 'Steps';
                        case 'Direction Pin'
                            property = 'Dir';
                        case 'Enable Pin'
                            property = 'En';
                        case 'Step Pin'
                            property = 'Pul';
                    end
                    if ~strcmp(key,'Motor Set')
                        app.motorsetData.(motorSet).(property) = str2double(value);
                    end
                end
            end
        end

    end


    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            try
                fID = fopen('Machine.PROPERTIES','r');
                readMachinePROPERTIES(app,fID);
                fclose(fID);
            catch ME
                app.errorCatch = ME;
                MAGIC.generateMachinePROPERTIES();
                fID = fopen('Machine.PROPERTIES','r');
                readMachinePROPERTIES(app,fID);
                fclose(fID);
            end
        end

        % Value changed function: CONNECTButton
        function attemptConnection(app, event)
            if ~strcmpi(app.COMPORTDropDown.Value,'-SELECT PORT-')
                if ~strcmpi(app.COMPORTDropDown.Value,'Bypass')
                    try
                        app.comPort = str2double(app.COMPORTDropDown.Value(4:end));
                        app.serverObject = Arduino_Server(...
                            app.comPort,...
                            app.baudRate);
                        app.CONNECTIONDisplay.String = '*** CONNECTED ***';
                        app.CONNECTIONDisplay.ForegroundColor = [0,1,0];
                        app.CONNECTButton.ValueChangedFcn =...
                            createCallbackFcn(app, @attemptDisconnect, true);
                        app.CONNECTButton.BackgroundColor = [1,0,0];
                        app.CONNECTButton.Text = 'DISCONNECT';
                    catch ME
                        app.errorCatch = ME;
                        app.CONNECTButton.Value = false;
                        for n = 1:3
                            app.CONNECTIONDisplay.Value = '*** ERROR CONNECTING ***';
                            tic;while(toc<0.75);end
                            app.CONNECTIONDisplay.Value = '*** PLEASE TRY AGAIN ***';
                            tic;while(toc<0.75);end
                            app.CONNECTIONDisplay.Value = '*** NOT CONNECTED ***';
                        end
                    end
                else
                    app.CONNECTButton.ValueChangedFcn =...
                        createCallbackFcn(app, @attemptDisconnect, true);
                    app.CONNECTButton.BackgroundColor = [1,0,0];
                    app.CONNECTButton.Text = 'DISCONNECT';
                end
            else
                app.CONNECTButton.Value = false;
            end
        end

        % Button pushed function: StartButton
        function startBraiding(app, event)
            % Create Even Stepper Motor Object and set Speed and Direction
            tSteps = app.motorsetData.even.Steps;
            tPul   = app.motorsetData.even.Pul;
            tDir   = app.motorsetData.even.Dir;
            tEn    = app.motorsetData.even.En;

            app.serverObject.createStepper(0,tSteps,tPul,tDir,tEn);
            setSpeed(app.serverObject.Stepper{app.evenMotor},'30RPM');
            setDirection(app.serverObject.Stepper{app.evenMotor},'CW');


            % Create Odd Stepper Motor Object and set Speed and Direction
            tSteps = app.motorsetData.odd.Steps;
            tPul   = app.motorsetData.odd.Pul;
            tDir   = app.motorsetData.odd.Dir;
            tEn    = app.motorsetData.odd.En;

            app.serverObject.createStepper(1,tSteps,tPul,tDir,tEn);
            setSpeed(app.serverObject.Stepper{app.oddMotor},'30RPM');
            setDirection(app.serverObject.Stepper{app.oddMotor},'CCW');

            n = 0;
            while(n < 100)
                n = n + 1;

                % Rotate Even Motors by 90 Degrees
                rotateMotor(app.serverObject.Stepper{app.evenMotor},90);
                tic;while(toc < 0.25);end % Wait 0.25 sec before next command

                % Rotate Odd Motors by 90 Degrees
                rotateMotor(app.serverObject.Stepper{app.oddMotor},90);
                tic;while(toc < 0.25);end % Wait 0.25 sec before next command
            end
        end
    end

    % App initialization and construction
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'UI Figure';

            % Create PROGRESSGaugeLabel
            app.PROGRESSGaugeLabel = uilabel(app.UIFigure);
            app.PROGRESSGaugeLabel.HorizontalAlignment = 'center';
            app.PROGRESSGaugeLabel.FontName = 'Consolas';
            app.PROGRESSGaugeLabel.Position = [292 352 58 15];
            app.PROGRESSGaugeLabel.Text = 'PROGRESS';

            % Create PROGRESSGauge
            app.PROGRESSGauge = uigauge(app.UIFigure, 'linear');
            app.PROGRESSGauge.MajorTicks =...
                [0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100];
            app.PROGRESSGauge.FontName = 'Consolas';
            app.PROGRESSGauge.Position = [18 382 606.203125 40];
            % Create BraidCyclesSpinnerLabel
            app.BraidCyclesSpinnerLabel = uilabel(app.UIFigure);
            app.BraidCyclesSpinnerLabel.HorizontalAlignment = 'right';
            app.BraidCyclesSpinnerLabel.FontName = 'Consolas';
            app.BraidCyclesSpinnerLabel.Position = [387 300 85 15];
            app.BraidCyclesSpinnerLabel.Text = 'Braid Cycles';

            % Create BraidCyclesSpinner
            app.BraidCyclesSpinner = uispinner(app.UIFigure);
            app.BraidCyclesSpinner.LowerLimitInclusive = 'off';
            app.BraidCyclesSpinner.Limits = [0 100];
            app.BraidCyclesSpinner.RoundFractionalValues = 'on';
            app.BraidCyclesSpinner.HorizontalAlignment = 'center';
            app.BraidCyclesSpinner.FontName = 'Consolas';
            app.BraidCyclesSpinner.Position = [487 296 100 22];
            app.BraidCyclesSpinner.Value = 1;

            % Create CONNECTIONPanel
            app.CONNECTIONPanel = uipanel(app.UIFigure);
            app.CONNECTIONPanel.TitlePosition = 'centertop';
            app.CONNECTIONPanel.Title = 'CONNECTION';
            app.CONNECTIONPanel.FontName = 'Consolas';
            app.CONNECTIONPanel.Position = [33 179 260 146];

            % Create CONNECTButton
            app.CONNECTButton = uibutton(app.CONNECTIONPanel, 'state');
            app.CONNECTButton.ValueChangedFcn = createCallbackFcn(app,...
                @attemptConnection, true);
            app.CONNECTButton.Text = 'CONNECT';
            app.CONNECTButton.BackgroundColor = [0 1 0];
            app.CONNECTButton.FontName = 'Consolas';
            app.CONNECTButton.Position = [81 13 100 22];

            % Create COMPORTDropDownLabel
            app.COMPORTDropDownLabel = uilabel(app.CONNECTIONPanel);
            app.COMPORTDropDownLabel.HorizontalAlignment = 'right';
            app.COMPORTDropDownLabel.FontName = 'Consolas';
            app.COMPORTDropDownLabel.Position = [30 60 58 15];
            app.COMPORTDropDownLabel.Text = 'COM PORT';

            % Create COMPORTDropDown
            app.COMPORTDropDown = uidropdown(app.CONNECTIONPanel);
            app.COMPORTDropDown.Items = {'-SELECT PORT-', 'COM0', 'COM1',...
                'COM2', 'COM3', 'COM4', 'COM5', 'COM6', 'COM7', 'COM8',...
                'COM9', 'Bypass'};
            app.COMPORTDropDown.FontName = 'Consolas';
            app.COMPORTDropDown.Position = [103 56 130 22];
            app.COMPORTDropDown.Value = '-SELECT PORT-';

            % Create CONNECTIONDisplay
            app.CONNECTIONDisplay = uieditfield(app.CONNECTIONPanel, 'text');
            app.CONNECTIONDisplay.Editable = 'off';
            app.CONNECTIONDisplay.HorizontalAlignment = 'center';
            app.CONNECTIONDisplay.FontName = 'Consolas';
            app.CONNECTIONDisplay.FontColor = [1 0 0];
            app.CONNECTIONDisplay.Position = [35 96 191 22];
            app.CONNECTIONDisplay.Value = '*** NOT CONNECTED ***';

            % Create StartButton
            app.StartButton = uibutton(app.UIFigure, 'push');
            app.StartButton.ButtonPushedFcn = createCallbackFcn(app,...
                @startBraiding, true);
            app.StartButton.Position = [272 42 100 22];
            app.StartButton.Text = 'Start';
        end
    end
    methods (Access = public)

        % Construct app
        function app = MAGIC

            % Create and configure components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

B.1 Matlab Code

Listing B.2: This is a Test

```
%+------------------------------------------------------------------------+
%|                                                                        |
%| FILENAME : Arduino_Server_m                            VERSION : 0.0.0 |
%|                                                                        |
%| TITLE : Arduino Server Object                  AUTHOR : Daniel Aldrich |
%|                                                                        |
%+------------------------------------------------------------------------+
%|                                                                        |
%| DEPENDENT FILES :                                                      |
%|       Arduino.m                                                        |
%|       Custom.m                                                         |
%|       Pololu.m                                                         |
%|       stepperMotor.m                                                   |
%|                                                                        |
%| DESCRIPTION :                                                          |
%|       Arduino Server Object acts as a container object for the provided|
%|         Libraries.                                                     |
%|                                                                        |
%| PUBLIC FUNCTIONS :                                                     |
%|       <various>                                                        |
%|                                                                        |
%| NOTES :                                                                |
%|       <note>                                                           |
%|                                                                        |
%| COPYRIGHT :                                                            |
%|       Copyright (c) 2017 Daniel Aldrich                                |
%{
%|       Permission is hereby granted, free of charge, to any person      |
%|       obtaining a copy of this software and associated documentation   |
%|       files (the "Software"), to deal in the Software without          |
%|       restriction, including without limitation the rights to use,     |
%|       copy, modify, merge, publish, distribute, sublicense, and/or     |
%|        sell copies of the Software, and to permit persons to whom the   |
%|        Software is furnished to do so, subject to the following         |
%|        conditions:                                                      |
%|                                                                         |
%|        The above copyright notice and this permission notice shall be   |
%|        included in all copies or substantial portions of the Software.  |
%|                                                                         |
%|        THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  |
%|        EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES  |
%|        OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND         |
%|        NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT      |
%|        HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,     |
%|        WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING     |
%|        FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR    |
%|        OTHER DEALINGS IN THE SOFTWARE.                                  |
%}
%|                                                                         |
%| CHANGES :                                                               |
%|        <none>                                                           |
%|                                                                         |
%+-------------------------------------------------------------------------+

classdef Arduino_Server < handle
    %% Properties
    properties(SetAccess = public)
        Stepper;
    Actuator;
    end

    properties(SetAccess = private)
        port = 'COM8';
        baud = 115000;
        arduino = [];
        Data = [0 0 0 0 0 0 0 0];
    end

    properties(SetAccess = private, Hidden)

    end

    %% Constructor
    methods(Access = public)
        function [this] = Arduino_Server(portNumber,baudRate)
            switch nargin
                case 2
                    this.port = ['COM',num2str(portNumber)];
                    this.baud = baudRate;
                case 1
                    this.port = ['COM',num2str(portNumber)];
                otherwise
                    % port = 'COM8';
                    % baud = 115000;
            end
            %instrreset;
            this.arduino = serial(this.port,...
                'Baudrate',this.baud,...
                'Timeout',0.001);
            this.arduino.BytesAvailableFcnCount = 1;
            this.arduino.BytesAvailableFcnMode = 'byte';
            fopen(this.arduino);
            this.arduino.BytesAvailableFcn = ...
                {@(~,~,x)fprintf(fscanf(x,'%c',1)),this.arduino};
        end
    end

    %% Methods (Custom)
    methods(Access = public)
        function callHome(this)
```

```
            this.Data = Custom.callHome();
            this.serialWrite();
        end
        function saveVar(this,Variable,Data)
            this.Data = Custom.saveVariable(Variable,Data);
            this.serialWrite();
        end
        function doMath(this,Mode,Var0,Var1)
            this.Data = Custom.Arithmetic(Mode,Var0,Var1);
            this.serialWrite();
        end
    end

    %% Methods (Arduino)
    methods(Access = public)
        function pinMode(this,pin,mode)
            this.Data = Arduino.pinMode(pin,mode);
            this.serialWrite();
        end
        function digitalRead(this,pin)
            this.Data = Arduino.digitalRead(pin);
            this.serialWrite();
        end
        function digitalWrite(this,pin,state)
            this.Data = Arduino.digitalWrite(pin,state);
            this.serialWrite();
        end
        function analogRead(this,pin)
            this.Data = Arduino.analogRead(pin);
            this.serialWrite();
        end
        function analogWrite(this,pin,value)
            this.Data = Arduino.analogWrite(pin,value);
            this.serialWrite();
        end
    end

    %% Methods (stepperMotor)
    methods(Access = public)
        function [] = createStepper(this, ID, Steps, stepPin,...
                dirPin, enablePin, Direction, Speed, Enable)
            switch nargin
                case 9
                    % Do Nothing
                case 8
                    Enable = false;
                case 7
                    Enable = false;
                    Speed = '60RPM';
                case 6
                    Enable = false;
                    Speed = '60RPM';
                    Direction = 'CW';
                case 5
                    Enable = false;
                    Speed = '60RPM';
                    Direction = 'CW';
                    stepPin = 2;
                case 4
                    Enable = false;
                    Speed = '60RPM';
                    Direction = 'CW';
                    stepPin = 2;
                    dirPin = 3;
                case 3
                    Enable = false;
                    Speed = '60RPM';
                    Direction = 'CW';
                    stepPin = 2;
                    dirPin = 3;
                    enablePin = 4;
                case 2
                    Enable = false;
                    Speed = '60RPM';
                    Direction = 'CW';
                    stepPin = 2;
                    dirPin = 3;
                    enablePin = 4;
                    Steps = 200;
                otherwise
                    Enable = false;
                    Speed = '60RPM';
                    Direction = 'CW';
                    stepPin = 2;
                    dirPin = 3;
                    enablePin = 4;
                    Steps = 200;
                    ID = 0;
            end
            this.Stepper{ID+1} = stepperMotor(this, ID, Steps, stepPin,...
                dirPin, enablePin, Direction, Speed, Enable);

        end
    end

    %% Methods (Pololu)
    methods(Access = public)
        function createMotor(this,Motor,Steps,stepPin,dirPin,enablePin)
            this.Data = ...
                Pololu.createMotor(Motor,Steps,stepPin,dirPin,enablePin);
            this.serialWrite;
        end
        function setSpeed(this,Motor,varargin)
            this.Data = Pololu.setSpeed(Motor,varargin{:});
            this.serialWrite();
        end
        function setDirection(this,Motor,Direction)
            this.Data = Pololu.setDirection(Motor,Direction);
            this.serialWrite();
        end
        function enableMotor(this,Motor)
            this.Data = Pololu.enableMotor(Motor);
            this.serialWrite();
        end
        function disableMotor(this,Motor)
            this.Data = Pololu.disableMotor(Motor);
            this.serialWrite();
        end
        function stepMotor(this,Motor,Steps)
            this.Data = Pololu.stepMotor(Motor,Steps);
            this.serialWrite();
        end
        function rotateMotor(this,Motor,Angle)
            this.Data = Pololu.rotateMotor(Motor,Angle);
            this.serialWrite();
        end
    end

    %% Methods Braider
    methods(Access = public)
        function [] = setupBraider()

        end
        function [] = setupXAxis() % Even

        end
        function [] = setupYAxis() % Odd

        end
        function [] = setupZAxis()

        end
        function [] = advanceX()

        end
        function [] = advanceY()

        end
        function [] = advanceZ()

        end
    end

    %% Methods (Private)
    methods(Access = private)
        function [] = serialWrite(this)
            if sum(this.Data)
                fwrite(this.arduino,this.Data,'uchar');
                Arduino_Server.serialDebounce();
            else
                % No Data to Send
            end
            this.Data = [0 0 0 0 0 0 0 0];
        end
        function [] = serialRead(~,~,this)
            display(fscanf(this.arduino,'%c'));
        end
    end

    %% Methods (Static)
    methods(Static)
        function [] = serialDebounce()
            tic;while(toc<0.25);end
        end
    end

    %% Methods (Private & Hidden)
    methods(Access = private, Hidden)

    end

    %% Methods (Hidden)
    methods(Access = public, Hidden)
        function bypassWrite(this,Data)
            % Allows other Libraries to update the Data vector to be written
            %   to the server, and initiate a write.
            this.Data = Data;
            this.serialWrite();
        end
        function bypassSetData(this,Data)
            this.Data = Data;
        end
        function bypassSendData(this)
            this.serialWrite();
        end
    end

    %% Events
    events

    end
end
```

B.1.1 Libraries

Listing B.3: This is a Test

```
%+------------------------------------------------------------------------+
%|                                                                        |
%| FILENAME : Arduino_m                                   VERSION : 0.0.0 |
%|                                                                        |
%| TITLE : Arduino Library                        AUTHOR : Daniel Aldrich |
%|                                                                        |
%+------------------------------------------------------------------------+
%|                                                                        |
%| DEPENDENT FILES :                                                      |
%|       <none>                                                           |
%|                                                                        |
%| DESCRIPTION :                                                          |
%|       Library containing basic function for controlling an Arduino.   |
%|                                                                        |
%| PUBLIC FUNCTIONS :                                                     |
%|       pinMode                                                          |
%|       digitalRead                                                      |
%|       digitalWrite                                                     |
%|       analogRead                                                       |
%|       analogWrite                                                      |
%|                                                                        |
%| NOTES :                                                                |
%|       <note>                                                           |
%|                                                                        |
%| COPYRIGHT :                                                            |
%|       Copyright (c) 2017 Daniel Aldrich                                |
%{
%|       Permission is hereby granted, free of charge, to any person      |
%|       obtaining a copy of this software and associated documentation   |
%|       files (the "Software"), to deal in the Software without          |
%|       restriction, including without limitation the rights to use,    |
%|       copy, modify, merge, publish, distribute, sublicense, and/or     |
%|       sell copies of the Software, and to permit persons to whom the   |
%|       Software is furnished to do so, subject to the following         |
%|       conditions:                                                      |
%|                                                                        |
%|       The above copyright notice and this permission notice shall be   |
%|       included in all copies or substantial portions of the Software.  |
%|                                                                        |
%|       THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  |
%|       EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES  |
%|       OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND         |
%|       NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT      |
%|       HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,     |
%|       WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING     |
%|       FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR    |
%|       OTHER DEALINGS IN THE SOFTWARE.                                  |
%}
%|                                                                        |
%| CHANGES :                                                              |
%|       <none>                                                           |
%|                                                                        |
%+------------------------------------------------------------------------+

classdef Arduino < handle
    %% Properties
    properties(Constant, Hidden)
        Library = 1
        HIGH = 1
        LOW = 0
        PULLUP = 2
        OUTPUT = 1
        INPUT = 0
    end

    %% Static Functions
    methods(Static)
        function [Data] = pinMode(pin,mode)
            if ischar(mode)
                mode = Arduino.(upper(mode));
            else
                mode = mod(mode,3);
            end
            Function = 0;
            Data = uint8([...
                Arduino.Library,...
                Function,...
                pin,...
                mode,...
                0,0,0,0]);
        end
        function [Data] = digitalRead(pin)
            Function = 1;
            Data = uint8([...
                Arduino.Library,...
                Function,...
                pin,...
                0,0,0,0,0]);
        end
        function [Data] = digitalWrite(pin,state)
            if ischar(state)
                state = Arduino.(state);
            else
                state = mod(state,2);
            end
            Function = 2;
            Data = uint8([...
                Arduino.Library,...
                Function,...
                pin,...
                state,...
                0,0,0,0]);
        end
        function [Data] = analogRead(pin)
            Function = 3;
            Data = uint8([...
                Arduino.Library,...
                Function,...
                pin,...
                0,0,0,0,0]);
        end
        function [Data] = analogWrite(pin,value)
            if ischar(value)
                value = str2double(value(1:end-1))*255/100;
            end
            Function = 4;
            Data = uint8([...
                Arduino.Library,...
                Function,...
                pin,...
                value,...
                0,0,0,0]);
        end
    end
end
```

Listing B.4: This is a Test

```
%+-------------------------------------------------------------------------+
%|                                                                         |
%| FILENAME : Custom_m                                     VERSION : 0.0.0 |
%|                                                                         |
%| TITLE : Custom Library                            AUTHOR : Daniel Aldrich |
%|                                                                         |
%+-------------------------------------------------------------------------+
%|                                                                         |
%| DEPENDENT FILES :                                                       |
%|       <none>                                                            |
%|                                                                         |
%| DESCRIPTION :                                                           |
%|       Library containing custom functions, meant for testing Server.    |
%|                                                                         |
%| PUBLIC FUNCTIONS :                                                      |
%|       saveVariable                                                      |
%|       Arithmetic                                                        |
%|                                                                         |
%| NOTES :                                                                 |
%|       <note>                                                            |
%|                                                                         |
%| COPYRIGHT :                                                             |
%|       Copyright (c) 2017 Daniel Aldrich                                 |
%{
%|       Permission is hereby granted, free of charge, to any person       |
%|       obtaining a copy of this software and associated documentation    |
%|       files (the "Software"), to deal in the Software without           |
%|       restriction, including without limitation the rights to use,      |
%|       copy, modify, merge, publish, distribute, sublicense, and/or      |
%|       sell copies of the Software, and to permit persons to whom the    |
%|       Software is furnished to do so, subject to the following          |
%|       conditions:                                                       |
%|                                                                         |
%|       The above copyright notice and this permission notice shall be    |
%|       included in all copies or substantial portions of the Software.   |
%|                                                                         |
%|       THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,   |
%|       EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES   |
%|       OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND          |
%|       NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT       |
%|       HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,      |
%|       WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING      |
%|       FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR     |
%|       OTHER DEALINGS IN THE SOFTWARE.                                   |
%}
%|                                                                         |
%| CHANGES :                                                               |
%|       <none>                                                            |
%|                                                                         |
%+-------------------------------------------------------------------------+

classdef Custom < handle
    %% Properties
    properties(Constant, Hidden)
        Library = 0;
    end

    %% Static Functions
    methods(Static)
        function [Data] = callHome()
            Function = 0;
            Data = uint8([...
                Custom.Library,...
                Function,...
                0,...
                0,...
                127,0,0,1]);
        end
        function [Data] = saveVariable(Variable,Data)
            Function = 1;
            Data = uint8([...
                Custom.Library,...
                Function,...
                Variable,...
                Data,...
                0,0,0,0]);
        end
        function [Data] = Arithmetic(Mode,Var0,Var1)
            switch lower(Mode)
                case {'+','addition','add'}
                    Mode = 0;
                case {'-','subtraction','minus'}
                    Mode = 1;
                case {'*','multiply','times'}
                    Mode = 2;
                case {'/','divide'}
                    Mode = 3;
                otherwise
                    Mode = mod(Mode,4);
            end

            Function = 2;
            Data = uint8([...
                Custom.Library,...
                Function,...
                Mode,...
                Var0,...
                Var1,...
                0,0,0]);
        end
    end
end
```

Listing B.5: This is a Test

```
%+------------------------------------------------------------------------+
%|                                                                        |
%| FILENAME : Pololu_m                                    VERSION : 0.0.0 |
%|                                                                        |
%| TITLE : Pololu Library                         AUTHOR : Daniel Aldrich |
%|                                                                        |
%+------------------------------------------------------------------------+
%|                                                                        |
%| DEPENDENT FILES :                                                      |
%|       <none>                                                           |
%|                                                                        |
%| DESCRIPTION :                                                          |
%|       Library for controlling step/direction based motor drivers.      |
%|                                                                        |
%| PUBLIC FUNCTIONS :                                                     |
%|       createMotor                                                      |
%|       setSpeed                                                         |
%|       setDirection                                                     |
%|       enableMotor                                                      |
%|       disableMotor                                                     |
%|       stepMotor                                                        |
%|       rotateMotor                                                      |
%|       setState                                                         |
%|                                                                        |
%| NOTES :                                                                |
%|       <note>                                                           |
%|                                                                        |
%| COPYRIGHT :                                                            |
%|       Copyright (c) 2017 Daniel Aldrich                                |
%{
%|       Permission is hereby granted, free of charge, to any person      |
%|       obtaining a copy of this software and associated documentation  |
%|       files (the "Software"), to deal in the Software without          |
%|       restriction, including without limitation the rights to use,     |
%|       copy, modify, merge, publish, distribute, sublicense, and/or     |
%|       sell copies of the Software, and to permit persons to whom the   |
%|       Software is furnished to do so, subject to the following         |
%|       conditions:                                                      |
%|                                                                        |
%|       The above copyright notice and this permission notice shall be   |
%|       included in all copies or substantial portions of the Software.  |
%|                                                                        |
%|       THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  |
%|       EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES  |
%|       OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND         |
%|       NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT      |
%|       HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,     |
%|       WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING     |
%|       FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR    |
%|       OTHER DEALINGS IN THE SOFTWARE.                                  |
%}
%|                                                                        |
%| CHANGES :                                                              |
%|       <none>                                                           |
%|                                                                        |
%+------------------------------------------------------------------------+

classdef Pololu < handle
    %% Properties
    properties(Constant, Hidden)
        Library = 2;
        CW = 1;
        CCW = 0;
    end

    %% Stactic Functions
    methods(Static)
        function [Data] = createMotor(Motor, Steps, stepPin, dirPin, enablePin)
            Function = 0;
      addSteps = floor(Steps/256);
            Data = uint8([...
                Pololu.Library,...
                Function,...
                Motor,...
        addSteps,...
                mod(Steps,256),...
                stepPin,...
                dirPin,...
                enablePin]);
        end
        function [Data] = setSpeed(Motor,varargin)
            if length(varargin)>=2
                Mode = varargin{2};
                Speed = varargin{1};
            elseif ischar(varargin{1})
                Mode = varargin{1}(end-2:end);
                Speed = str2double(varargin{1}(1:end-3));
            else
                Mode = 'RPM';
                Speed = varargin{1};
            end

            switch upper(Mode)
                case {'RPM'}
                    Mode = 0;
                case {'RPS'}
                    Mode = 1;
                case {'RAD'}
                    Mode = 2;
                case {'MRPM'}
```

```
                    Mode = 3;
                otherwise
                    Mode = mod(Mode,4);
            end

            Function = 1;
            Data = uint8([...
                Pololu.Library,...
                Function,...
                Mode,...
                Motor,...
                Speed,...
                0,0,0]);
        end
        function [Data] = setDirection(Motor,Direction)
            switch upper(Direction)
                case {'CW','CLOCKWISE'}
                    Direction = Pololu.CW;
                case {'CCW','COUNTERCLOCKWISE'}
                    Direction = Pololu.CCW;
                otherwise
                    Direction = mod(Direction,2);
            end
            Function = 2;
            Data = uint8([...
                Pololu.Library,...
                Function,...
                Motor,...
                Direction,...
                0,0,0,0]);
        end
        function [Data] = enableMotor(Motor)
            Function = 3;
            Data = uint8([...
                Pololu.Library,...
                Function,...
                Motor,...
                0,0,0,0,0]);
        end
        function [Data] = disableMotor(Motor)
            Function = 4;
            Data = uint8([...
                Pololu.Library,...
                Function,...
                Motor,...
                0,0,0,0,0]);
        end
        function [Data] = stepMotor(Motor,Steps)
            Function = 5;
            if Steps>255
                binSteps = dec2bin(Steps,16);
                Steps = bin2dec(binSteps(9:end));
                extraSteps = bin2dec(binSteps(1:8));
            else
                extraSteps = 0;
            end
            Data = uint8([...
                Pololu.Library,...
                Function,...
                Motor,...
                Steps,...
                extraSteps,0,0,0]);
        end
        function [Data] = rotateMotor(Motor, Angle)
            Function = 6;
            if Angle>255
                binAngle = dec2bin(Angle,16);
                Angle = bin2dec(binAngle(9:end));
                extraAngle = bin2dec(binAngle(1:8));
            else
                extraAngle = 0;
            end
            Data = uint8([...
                Pololu.Library,...
                Function,...
                Motor,...
                Angle,...
                extraAngle,0,0,0]);
        end
        function [Data] = setState(Motor, State)
            switch upper(State)
                case {'ENABLE',1,true,'ON','POWERED','ENABLED'}
                    Data = Pololu.enableMotor(Motor);
                case {'DISABLE',0,false,'OFF','DISABLED','UNPOWERED'}
                    Data = Pololu.disableMotor(Motor);
            end
        end
    end
end
```

Listing B.6: This is a Test

```
%+------------------------------------------------------------------------+
%|                                                                        |
%| FILENAME : stepperMotor_m                              VERSION : 0.0.0 |
%|                                                                        |
%| TITLE : Stepper Motor Object                   AUTHOR : Daniel Aldrich |
%|                                                                        |
%+------------------------------------------------------------------------+
%|                                                                        |
%| DEPENDENT FILES :                                                      |
%|       Pololu_m                                                         |
%|                                                                        |
%| DESCRIPTION :                                                          |
%|       Stepper Motor Object acts as a container for the Pololu Library, |
%|         while storing individual properties on a motor.                |
%|                                                                        |
%| PUBLIC FUNCTIONS :                                                     |
%|       createMotor                                                      |
%|       setSpeed                                                         |
%|       setDirection                                                     |
%|       enableMotor                                                      |
%|       disableMotor                                                     |
%|       stepMotor                                                        |
%|       rotateMotor                                                      |
%|       setState                                                         |
%|       toggleState                                                      |
%|       stepMotorHP                                                      |
%|       rotateMotorHP                                                    |
%|       enableMotorHP                                                    |
%|       disableMotorHP                                                   |
%|                                                                        |
%| NOTES :                                                                |
%|       <note>                                                           |
%|                                                                        |
%| COPYRIGHT :                                                            |
%|       Copyright (c) 2017 Daniel Aldrich                                |
%{
%|       Permission is hereby granted, free of charge, to any person      |
%|       obtaining a copy of this software and associated documentation   |
%|       files (the "Software"), to deal in the Software without          |
%|       restriction, including without limitation the rights to use,    |
%|       copy, modify, merge, publish, distribute, sublicense, and/or     |
%|       sell copies of the Software, and to permit persons to whom the   |
%|       Software is furnished to do so, subject to the following         |
%|       conditions:                                                   |
%|                                                                     |
%|       The above copyright notice and this permission notice shall be  |
%|       included in all copies or substantial portions of the Software.  |
%|                                                                     |
%|       THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  |
%|       EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES  |
%|       OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND         |
%|       NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT      |
%|       HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,     |
%|       WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING     |
%|       FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR    |
%|       OTHER DEALINGS IN THE SOFTWARE.                                  |
%}
%|                                                                     |
%| CHANGES :                                                           |
%|       <none>                                                        |
%|                                                                     |
%+---------------------------------------------------------------------+

classdef stepperMotor < Pololu
    %% Properties
    properties(SetAccess = private)
        Server
        ID
        Steps
        Direction
        Speed
        State
    end
    properties(SetAccess = private, Hidden)
        stepDelay
    end

    %% Constructor
    methods
        function[this] = stepperMotor(Server, ID, Steps, stepPin,...
                dirPin, enablePin, Direction, Speed, Enable)
            switch nargin
                case 8
                    ID = 0;
                case 7
                    ID = 0;
                    Steps = 200;
                case 6
                    ID = 0;
                    Steps = 200;
                    stepPin = 13;
                case 5
                    ID = 0;
                    Steps = 200;
                    stepPin = 13;
                    dirPin = 13;
                case 4
                    ID = 0;
                    Steps = 200;
                    stepPin = 13;
                    dirPin = 13;
                    enablePin = 13;
                case 3
                    ID = 0;
                    Steps = 200;
                    stepPin = 13;
                    dirPin = 13;
                    enablePin = 13;
                    Direction = 'CW';
                case 2
                    ID = 0;
                    Steps = 200;
                    stepPin = 13;
                    dirPin = 13;
                    enablePin = 13;
                    Direction = 'CW';
                    Speed = '60RPM';
                case 1
                    ID = 0;
                    Steps = 200;
                    stepPin = 13;
                    dirPin = 13;
                    enablePin = 13;
                    Direction = 'CW';
                    Speed = '60RPM';
                    Enable = 0;
                otherwise

            end

            this.Server = Server;
            this.ID = ID;
            this.Steps = Steps;
            this.Direction = Direction;
            this.Speed = Speed;
      if enablePin<0
        enablePin = abs(enablePin) + 128;
      end
            Mode = Speed(end-2:end);
            Vel = str2double(Speed(1:end-3));
            switch upper(Mode)
                case {'RPM'}
                    Factor = 60;
                case {'RPS'}
                    Factor = 1;
                case {'RAD'}
                    Factor= 2*pi();
                otherwise
            end
            this.stepDelay = inv(this.Steps*Vel/Factor);
            this.State = Enable;

            Data = this.createMotor(ID, Steps, stepPin, dirPin, enablePin);
            this.Server.bypassWrite(Data);

            this.setState(Enable);
            this.setSpeed(Speed);
            this.setDirection(Direction);
        end
    end

    %% Methods
    methods
        function [] = enableMotor(this)
            if ~this.State
                [Data] = enableMotor@Pololu(this.ID);
                this.Server.bypassWrite(Data);
            end
        end
        function [] = disableMotor(this)
            if this.State
                [Data] = disableMotor@Pololu(this.ID);
                this.Server.bypassWrite(Data);
            end
        end
        function [] = setSpeed(this, varargin)
            [Data] = setSpeed@Pololu(this.ID, varargin{:});
            if length(varargin)>=2
                Mode = varargin{2};
```

```
                Vel = varargin{1};
            elseif ischar(varargin{1})
                Mode = varargin{1}(end-2:end);
                Vel = str2double(varargin{1}(1:end-3));
            else
                Mode = 'RPM';
                Vel = varargin{1};
            end
            this.Speed = [num2str(Vel),Mode];
            switch upper(Mode)
                case {'RPM'}
                    Factor = 60;
                case {'RPS'}
                    Factor = 1;
                case {'RAD'}
                    Factor = 2*pi();
                case {'MRPM'}
                    Factor = 60000;
                otherwise
                    ind = mod(Mode,4);
                    Factor = [60 1 2*pi() 60000];
                    Factor = Factor(ind);
            end
            this.stepDelay = inv(this.Steps*Vel/Factor);
            this.Server.bypassWrite(Data);
        end
        function [] = setDirection(this, Direction)
            [Data] = setDirection@Pololu(this.ID, Direction);
            this.Direction = Direction;
            this.Server.bypassWrite(Data);
        end
        function [] = stepMotor(this, Steps)
            if ~this.State
                [Data] = Pololu.enableMotor(this.ID);
                this.Server.bypassWrite(Data);
            end
            [Data] = stepMotor@Pololu(this.ID, Steps);
            this.Server.bypassWrite(Data);
            delay = this.stepDelay*Steps;
            tic;while(toc<delay);end
            if ~this.State
                [Data] = Pololu.disableMotor(this.ID);
                this.Server.bypassWrite(Data);
            end
        end
        function [] = rotateMotor(this, Angle)
            if ~this.State
                [Data] = Pololu.enableMotor(this.ID);
                this.Server.bypassWrite(Data);
            end
            [Data] = rotateMotor@Pololu(this.ID, Angle);
            numofSteps = Angle*this.Steps/360;
            this.Server.bypassWrite(Data);
            delay = this.stepDelay*numofSteps;
            tic;while(toc<delay);end
            if ~this.State
                [Data] = Pololu.disableMotor(this.ID);
                this.Server.bypassWrite(Data);
            end
        end
        function [] = setState(this, State)
            [Data] = setState@Pololu(this.ID, State);
            this.State = ~(Data(2)-3);
            this.Server.bypassWrite(Data);
        end
        function [] = toggleState(this)
            this.State = ~this.State;
            [Data] = Pololu.setState(this.ID, this.State);
            this.Server.bypassWrite(Data);
        end
    end
end
```

B.2 Arduino Code

Listing B.7: This is a Test

```
//+-------------------------------------------------------------------------+
//|                                                                         |
//| FILENAME : Arduino_Server_ino                           VERSION : b.0.2 |
//|                                                                         |
//| TITLE : Arduino Server Protocol                 AUTHOR : Daniel Aldrich |
//|                                                                         |
//+-------------------------------------------------------------------------+
//|                                                                         |
//| DEPENDENT FILES :                                                       |
//|       Pololu_h                                                          |
//|       Pololu_cpp                                                        |
//|                                                                         |
//| DESCRIPTION :                                                           |
//|       Server function to be run on an Arduino board. Command are sent  |
//|         as 8 8bit instructions. The instructions consist of the         |
//|         Library, Function, Option, and Additional Data. Because each    |
//|         the Library and Function Consist of an 8bit number there are    |
//|         256 available Libraries each with 265 available Funcitons. The  |
//|         instructions are to be sent in the following order:             |
//|                                                                         |
//|         Library, Function, Option, Data0, Data1, Data2, Data3, Data4    |
//|                                                                         |
//| PUBLIC FUNCTIONS :                                                      |
//|       <none>                                                            |
//|                                                                         |
//| NOTES :                                                                 |
//|       <none>                                                            |
//|                                                                         |
//| COPYRIGHT :                                                             |
//|       Copyright (c) 2017 Daniel Aldrich                                 |
/**
**|       Permission is hereby granted, free of charge, to any person       |
**|       obtaining a copy of this software and associated documentation    |
**|       files (the "Software"), to deal in the Software without           |
**|       restriction, including without limitation the rights to use,      |
**|       copy, modify, merge, publish, distribute, sublicense, and/or      |
**|       sell copies of the Software, and to permit persons to whom the    |
**|       Software is furnished to do so, subject to the following          |
**|       conditions:                                                       |
**|                                                                         |
**|       The above copyright notice and this permission notice shall be    |
**|       included in all copies or substantial portions of the Software.   |
**|                                                                         |
**|       THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,   |
**|       EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES   |
**|       OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND          |
**|       NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT       |
**|       HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,      |
**|       WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING      |
**|       FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR     |
**|       OTHER DEALINGS IN THE SOFTWARE.                                   |
**/
//|                                                                         |
//| CHANGES :                                                               |
//|       <none>                                                            |
//|                                                                            |
//+----------------------------------------------------------------------------+
#include <Pololu.h>

float tempArray[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

Pololu motorsPololu[5];

float Number0;
float Number1;

byte pin;
byte pwm;
boolean state;

float ans;

char  tempChar;
String  tempStr;
unsigned int tempUint;

byte  Library;
byte  Function;
byte  Option;
byte  Data0;
byte  Data1;
byte  Data2;
byte  Data3;
byte  Data4;

unsigned long timer;
unsigned long timeout = 500000; // 0.5 second timeout for data
boolean timedout = false;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115000);
  Serial.println("Starting Server");
  Serial.println("Connection Established!");
  PORTK = B11111111;
  PORTF = B11111111;
}

void loop() {
  timedout = false;
  if (Serial.available() > 0){
    Library = Serial.read();
    timer = micros();
    while (Serial.available() <= 0)
    {
      if ( (micros() - timer) >= timeout )
      {
        timedout = true;
        break;
      };
    };
    if (!timedout)
    {
      Function = Serial.read();
      timer = micros();
      while (Serial.available() <= 0)
      {
        if ( (micros() - timer) >= timeout )
        {
          timedout = true;
          break;
        };
      };
    };
    if (!timedout)
    {
      Option = Serial.read();
      timer = micros();
      while (Serial.available() <= 0)
      {
        if ( (micros() - timer) >= timeout )
        {
          timedout = true;
          break;
        };
      };
    };
    if (!timedout)
    {
      Data0 = Serial.read();
      timer = micros();
      while (Serial.available() <= 0)
      {
        if ( (micros() - timer) >= timeout )
        {
          timedout = true;
          break;
        };
      };
    };
    if (!timedout)
    {
      Data1 = Serial.read();
      timer = micros();
      while (Serial.available() <= 0)
      {
        if ( (micros() - timer) >= timeout )
        {
          timedout = true;
          break;
        };
      };
    };
    if (!timedout)
    {
      Data2 = Serial.read();
      timer = micros();
      while (Serial.available() <= 0)
      {
        if ( (micros() - timer) >= timeout )
        {
          timedout = true;
          break;
        };
      };
    };
    if (!timedout)
    {
      Data3 = Serial.read();
      timer = micros();
      while (Serial.available() <= 0)
      {
        if ( (micros() - timer) >= timeout )
        {
          timedout = true;
          break;
        };
      };
    };
    if (!timedout)
    {
```

```
Data4 = Serial.read();
switch (Library)
{
  case 0x00:
    /* This Library is reserved for server testing functions.      */
    switch (Function){
      case 0x00:
        // Null Function
        if(!Option && !Data0){
          if (Data1==127 && Data2==0 && Data3==0 && Data4==1){
            // Call Home
            Serial.println("localhost");
          };
        };
      break;
      case 0x01:
        // Store Variable
        tempArray[Option] = Data0;

        Serial.print("var");
        Serial.print(Option);
        Serial.print(" = ");
        Serial.println(Data0);
        break;

      case 0x02:
        // Arithmetic
        Number0 = tempArray[Data0];
        Number1 = tempArray[Data1];
        switch (Option){
          case 0x00:
            ans = Number0 + Number1;
            tempChar = '+';
          break;
          case 0x01:
            ans = Number0 - Number1;
            tempChar = '-';
          break;
          case 0x02:
            ans = Number0 * Number1;
            tempChar = '*';
          break;
          case 0x03:
            ans = Number0 / Number1;
            tempChar = '/';
          break;
        }
        Serial.print(Number0);
        Serial.print(" ");
        Serial.print(tempChar);
        Serial.print(" ");
        Serial.print(Number1);
        Serial.println(" =");
        Serial.print("    ");
        Serial.println(ans);
        break;
    }
  break;
  case 0x01:
    /* This Library is for performing standard arduino functions.
     *  This includes: Changing pins, Setting inputs, Setting outputs
     */
    switch (Function){
      case 0x00:
        //pinMode
        /*
         * Option is the pin Number
         * Option: 1 to n (number of pins, where analog pins
         *   are a continuation of the digital pins)
         * Data is the state of the pin (Input or Output)
         * Data: 0 (Input) or 1 (Output)
         */
        pin = Option;
        state = Data0%2;
        pinMode(pin,state);

        if (state){
          tempStr = "Output";
        }else{
          tempStr = "Input";
        }

        Serial.print("pin ");
        Serial.print(pin);
        Serial.print(" was set to ");
        Serial.println(tempStr);
      break;
      case 0x01:
        // digitalRead
        pin = Option;
        ans = digitalRead(pin);

        Serial.print("pin ");
        Serial.print(pin);
        Serial.print(" reads ");
        Serial.println(ans);
      break;
      case 0x02:
        // digitalWrite
        pin = Option;
        state = Data0%2;
        digitalWrite(pin,state);

        if (state){
          tempStr = "High";
        }else{
          tempStr = "Low";
        }

        Serial.print("pin ");
        Serial.print(pin);
        Serial.print(" was set to ");
        Serial.println(tempStr);
      break;
      case 0x03:
        // analogRead
        pin = Option;
        ans = analogRead(pin);

        Serial.println(ans);
      break;
      case 0x04:
        // analogWrite
        pin = Option;
        pwm = Data0%256;
        analogWrite(pin,pwm);

        Serial.print("pin ");
        Serial.print(pin);
        Serial.print(" was set to ");
        Serial.print(100.0*pwm/255.0);
        Serial.println(" %");
      break;
    }
  break;
  case 0x02:
        /* This is a Library for the use of Pololu stepper drivers */
        switch (Function){
          case 0x00:
            /* Create a Motor Object */
            motorsPololu[Option] =
              Pololu(uint16_t(Data0), uint16_t(Data1), int(Data2), int(Data3), int(Data4));
            // (number_of_steps_0, number_of_steps_1, pin_step, pin_direction, pin_enable)
          break;
          case 0x01:
            /* Set the speed of the Motor */
            switch (Option){
              case 0x00:
                // Set using Rev per Min
                motorsPololu[int(Data0)].setRPM(Data1);
              break;
              case 0x01:
                // Set Using Rev per Sec
                motorsPololu[int(Data0)].setRPS(Data1);
              break;
              case 0x02:
                // Set Using Rad per Sec
                motorsPololu[int(Data0)].setRad(Data1);
              break;
              case 0x03:
                // Set Using Rad per Sec
                motorsPololu[int(Data0)].setmRPM(Data1);
              break;
            }
          break;
          case 0x02:
            /* Set the Direction of the Motor */
            motorsPololu[int(Option)].setDirection(Data0);
          break;
          case 0x03:
            /* Enable the Motor for Use */
            motorsPololu[int(Option)].enable();
          break;
          case 0x04:
            /* Disable the Motor to save Power */
            motorsPololu[int(Option)].disable();
          break;
          case 0x05:
            /* Steps the motor by the number of steps in Data0 */
            tempUint = (int(Data1)<<8) + int(Data0);
            motorsPololu[int(Option)].stepper(tempUint);
          break;
          case 0x06:
            /* Degrees to rotate the motor in it's preset Direction */
            tempUint = (int(Data1)<<8) + int(Data0);
            motorsPololu[int(Option)].rotate(tempUint);
          break;
        }
      break;
      case 0x03:
        /* */
        switch (Function){
            case 0x00:
            /* */

            break;
        }
      break;
    }
  }else{
    Serial.println("Error: Timedout waiting for Data");
  }
}
}
```

B.2.1 Libraries

Listing B.8: This is a Test

```
//+------------------------------------------------------------------------+
//|                                                                        |
//| FILENAME : Pololu_h                                    VERSION : 1.0.0 |
//|                                                                        |
//| TITLE : Pololu Library Header                  AUTHOR : Daniel Aldrich |
//|                                                                        |
//+------------------------------------------------------------------------+
//|                                                                        |
//| DEPENDENT FILES :                                                      |
//|       Pololu_cpp                                                       |
//|                                                                        |
//| DESCRIPTION :                                                          |
//|       <none>                                                           |
//|                                                                        |
//| PUBLIC FUNCTIONS :                                                     |
//|       setRPM                                                           |
//|       setRPS                                                           |
//|       setRad                                                           |
//|       setDirection                                                     |
//|       enable                                                           |
//|       disable                                                          |
//|       stepper                                                          |
//|       rotate                                                           |
//|       enableHP                                                         |
//|       disableHP                                                        |
//|       stepperHP                                                        |
//|       rotateHP                                                         |
//|                                                                        |
//| NOTES :                                                                |
//|       <none>                                                           |
//|                                                                        |
//| COPYRIGHT :                                                            |
//|       Copyright (c) 2017 Daniel Aldrich                                |
/**
**|       Permission is hereby granted, free of charge, to any person     |
**|       obtaining a copy of this software and associated documentation   |
**|       files (the "Software"), to deal in the Software without          |
**|       restriction, including without limitation the rights to use,     |
**|       copy, modify, merge, publish, distribute, sublicense, and/or     |
**|       sell copies of the Software, and to permit persons to whom the   |
**|       Software is furnished to do so, subject to the following         |
**|       conditions:                                                      |
**|                                                                        |
**|       The above copyright notice and this permission notice shall be   |
**|       included in all copies or substantial portions of the Software.  |
**|                                                                        |
**|       THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  |
**|       EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES  |
**|       OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND         |
**|       NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT      |
**|       HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,     |
**|       WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING     |
**|       FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR    |
**|       OTHER DEALINGS IN THE SOFTWARE.                                  |
**/
//|                                                                        |
//| CHANGES :                                                              |
//|       <none>                                                           |
//|                                                                        |
//+-------------------------------------------------------------------------+

// Ensure Library is only included once
#ifndef Pololu_h
#define Pololu_h

#define CW  1
#define CCW 0

class Pololu
{
  private:
    // Properties:
      unsigned long number_of_steps;
      int direction;

      unsigned long step_delay;
      unsigned long timer;

    // Hardware:
      int  pin_steps;
      int  pin_direction;
      int  pin_enable;
      bool reversePolarity;

  public:
    // Constructor:
      Pololu(uint16_t number_of_steps_0, uint16_t number_of_steps_1, int pin_step,
      int pin_direction, int pin_enable);
      Pololu();
    // Methods:
      void setRPM(long rev_per_min);
      void setRPS(long rev_per_sec);
      void setRad(long rad_per_sec);
      void setmRPM(long mrev_per_min);

      void setDirection(bool direction);

      void enable(void);
      void disable(void);
      void stepper(unsigned int steps_to_take);
      void rotate(unsigned int degrees_to_rotate);
  private:
    // Methods:
      void stepMotor(void);
      void stepMotorHP(void);
};

#endif
```

Listing B.9: This is a Test

```
//+-------------------------------------------------------------------------+
//|                                                                         |
//| FILENAME : Pololu_cpp                                   VERSION : 1.0.0 |
//|                                                                         |
//| TITLE : Pololu Library Source                   AUTHOR : Daniel Aldrich |
//|                                                                         |
//+-------------------------------------------------------------------------+
//|                                                                         |
//| DEPENDENT FILES :                                                       |
//|       Pololu_h                                                          |
//|                                                                         |
//| DESCRIPTION :                                                           |
//|       <none>                                                            |
//|                                                                         |
//| PUBLIC FUNCTIONS :                                                      |
//|       setRPM                                                           |
//|       setRPS                                                           |
//|       setRad                                                           |
//|       setDirection                                                     |
//|       enable                                                           |
//|       disable                                                          |
//|       stepper                                                          |
//|       rotate                                                           |
//|       enableHP                                                         |
//|       disableHP                                                        |
//|       stepperHP                                                        |
//|       rotateHP                                                         |
//|                                                                        |
//| NOTES :                                                                |
//|       <none>                                                           |
//|                                                                        |
//| COPYRIGHT :                                                            |
//|       Copyright (c) 2017 Daniel Aldrich                                |
/**
**|       Permission is hereby granted, free of charge, to any person      |
**|       obtaining a copy of this software and associated documentation   |
**|       files (the "Software"), to deal in the Software without          |
**|       restriction, including without limitation the rights to use,     |
**|       copy, modify, merge, publish, distribute, sublicense, and/or     |
**|       sell copies of the Software, and to permit persons to whom the   |
**|       Software is furnished to do so, subject to the following         |
**|       conditions:                                                      |
**|                                                                        |
**|       The above copyright notice and this permission notice shall be   |
**|       included in all copies or substantial portions of the Software.  |
**|                                                                        |
**|       THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  |
**|       EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES  |
**|       OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND         |
**|       NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT      |
**|       HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,     |
**|       WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING     |
**|       FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR    |
**|       OTHER DEALINGS IN THE SOFTWARE.                                  |
**/
//|                                                                        |
//| CHANGES :                                                              |
//|       <none>                                                           |
//|                                                                        |
//+------------------------------------------------------------------------+

#include ”Arduino.h”
#include ”Pololu.h”

// Constructors
Pololu::Pololu(uint16_t number_of_steps_0, uint16_t number_of_steps_1, int pin_steps,
    int pin_direction, int pin_enable)
{
  this->number_of_steps = (long(number_of_steps_0)<<8) + number_of_steps_1;
  this->direction = 1;
  this->step_delay = 0;
  this->pin_steps = pin_steps;
  this->pin_direction = pin_direction;
  this->pin_enable = 0x7F & pin_enable;
  this-> reversePolarity = bool(pin_enable>>7);
  pinMode(this->pin_steps, OUTPUT);
  pinMode(this->pin_direction, OUTPUT);
  pinMode(this->pin_enable, OUTPUT);
};
Pololu::Pololu()
{
  this->number_of_steps = 200;
  this->direction = 1;
  this->step_delay = 0;
  this->pin_steps = 2;
  this->pin_direction = 3;
  this->pin_enable = 4;

  pinMode(this->pin_steps, OUTPUT);
  pinMode(this->pin_direction, OUTPUT);
  pinMode(this->pin_enable, OUTPUT);
  digitalWrite(this->pin_enable,HIGH);
};

//Methods
void Pololu::setRPM(long rev_per_min)
{
  this->step_delay = 60000000L/(this->number_of_steps * rev_per_min);
};

void Pololu::setRPS(long rev_per_sec)
{
  this->step_delay = 1000000L/(this->number_of_steps * rev_per_sec);
};

void Pololu::setRad(long rad_per_sec)
{
  this->step_delay = 1000000L/(this->number_of_steps*rad_per_sec/6.283185L);
};
void Pololu::setmRPM(long mrev_per_min)
{
  this->step_delay = 60000000000L/(this->number_of_steps*mrev_per_min);
};
void Pololu::setDirection(bool direction)
{
  this->direction = direction;
};

void Pololu::enable(void)
{
  if(this->reversePolarity)
  {
    digitalWrite(this->pin_enable, HIGH);
  }else{
    digitalWrite(this->pin_enable, LOW);
  }
};

void Pololu::disable(void)
{
  if(this->reversePolarity)
  {
    digitalWrite(this->pin_enable, LOW);
  }else{
    digitalWrite(this->pin_enable, HIGH);
  }
};

void Pololu::stepper(unsigned int steps_to_take)
{
  digitalWrite(this->pin_direction,this->direction);
  while (steps_to_take > 0)
  {
    this->stepMotor();
    --steps_to_take;
  }
};

void Pololu::rotate(unsigned int degrees_to_rotate)
{
  this->stepper((unsigned int)((float)degrees_to_rotate*(float)this->number_of_steps/360.0));
```

```
};

void Pololu::stepMotor(void)
{
  this->timer = micros();

  digitalWrite(this->pin_steps, HIGH);
  while ( (micros() - this->timer) <= this->step_delay/2 ){};

  digitalWrite(this->pin_steps, LOW);
  while ( (micros() - this->timer) <= this->step_delay){};
};
```

Appendix C: Path Generation and Property Estimation

C.1 Material Library

Listing C.1: .

```
<!-- This file is designed to store material properties for the various  -->
<!--   materials used in 3D braiding.                                     -->
<!--                                                                      -->
<!-- There are two types of materials that can be declared using the     -->
<!--   Keywords 'Fiber' or 'Matrix'. To start a material declaration use -->
<!--   '<Keyword>' and to end a declaration use '</Keyword>'.            -->
<!--                                                                      -->
<!-- When specifing a material the first line following the declaration  -->
<!--   must be the name of the material:                                 -->
<!--    e.g. name=Name of Material                                       -->
<!--                                                                      -->
<!-- For a Fiber, following the name of the material, the next lines     -->
<!--   should contain (in any order):                                    -->
<!--   E1       ~  Logitudinal Young's Modulus [GPa]                     -->
<!--   E2       ~  Transverse Young's Modulus [GPa]                      -->
<!--   v        ~  Possion's Ratio                                       -->
<!--   G        ~  Logitudinal Shear Modulus [GPa]                       -->
<!--   Denier   ~  Linear Density [g/9000m]                              -->
<!--   Density  ~  Density [s.g.] or [g/cm3]                             -->
<!--                                                                      -->
<!-- For a Matrix, following the name of the material, the next lines    -->
<!--   should contain (in any order):                                    -->
<!--   E        ~  Young's Modulus [GPa]                                 -->
<!--   v        ~  Possion's Ratio                                       -->

<Materials>
  <Fiber>
    name=Aramid 29
    E1=61
    E2=4.2
    v=0.35
    G=2.9
    Denier=1625
    Density=1.44
  </Fiber>
  <Fiber>
    name=Aramid 49
    E1=154
    E2=4.2
    v=0.35
    G=2.9
    Denier=1140
    Density=1.44
  </Fiber>
  <Fiber>
```

```
    name=Graphite PITCH
    E1=385
    E2=6.3
    v=0.2
    G=7.7
    Denier=1500
    Density=1.99
  </Fiber>
  <Fiber>
    name=Graphite PAN
    E1=224
    E2=14
    v=0.2
    G=14
    Denier=1500
    Density=1.78
  </Fiber>
  <Fiber>
    name=Glass
    E1=71
    E2=71
    v=0.22
    G=29.1
    Denier=400
    Density=2.55
  </Fiber>
  <Matrix>
    name=Epoxy
    E=3.4
    v=0.3
  </Matrix>
  <Matrix>
    name=Polyester
    E=2.5
    v=0.33
  </Matrix>
  <Matrix>
    name=Aluminium
    E=69
    v=0.3
  </Matrix>
  <Matrix>
    name=Polyamide NYLON
    E=3.5
    v=0.35
  </Matrix>
</Materials>
```

C.2 Path Generation Code

Listing C.2: Program interface for generating braid paths and for initiating the generation of the braid model.

```
classdef braidGenerator < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        LabelNumericEditField2          matlab.ui.control.Label
        etPathAngle                     matlab.ui.control.NumericEditField
        LabelNumericEditField3          matlab.ui.control.Label
        etBraidWidth                    matlab.ui.control.NumericEditField
        LabelNumericEditField4          matlab.ui.control.Label
        etKnotsinSpline                 matlab.ui.control.NumericEditField
        LabelNumericEditField5          matlab.ui.control.Label
        etBraidDepth                    matlab.ui.control.NumericEditField
        LabelNumericEditField6          matlab.ui.control.Label
        etSpacingFactor                 matlab.ui.control.NumericEditField
        btGeneratePlots                 matlab.ui.control.Button
        btGenerateModel                 matlab.ui.control.Button
        LabelNumericEditField7          matlab.ui.control.Label
        etNumofUnitCells                matlab.ui.control.NumericEditField
        cbSmoothing                     matlab.ui.control.CheckBox
        cbTriAxial                      matlab.ui.control.CheckBox
        btGeneratePaths                 matlab.ui.control.Button
        LabelEditField                  matlab.ui.control.Label
        etFolderName                    matlab.ui.control.EditField
        btClosePlots                    matlab.ui.control.Button
        cbType2Paths                    matlab.ui.control.CheckBox
        LabelDropDown                   matlab.ui.control.Label
        ddFiberSelect                   matlab.ui.control.DropDown
        LabelDropDown2                  matlab.ui.control.Label
        ddMatrixSelect                  matlab.ui.control.DropDown
        LabelNumericEditField9          matlab.ui.control.Label
        etFiberE1                       matlab.ui.control.NumericEditField
        Label                           matlab.ui.control.Label
        etFiberv                        matlab.ui.control.NumericEditField
        Label2                          matlab.ui.control.Label
        etFiberG                        matlab.ui.control.NumericEditField
        Label3                          matlab.ui.control.Label
        etMatrixE                       matlab.ui.control.NumericEditField
        Label4                          matlab.ui.control.Label
        etMatrixv                       matlab.ui.control.NumericEditField
        Label5                          matlab.ui.control.Label
        etMatrixG                       matlab.ui.control.NumericEditField
        Label33                         matlab.ui.control.Label
        etFiberE2                       matlab.ui.control.NumericEditField
        cbDenier                        matlab.ui.control.CheckBox
        DenierEditFieldLabel            matlab.ui.control.Label
        etDenier                        matlab.ui.control.NumericEditField
        UnitCellHeightPitchLabel        matlab.ui.control.Label
        etZHeight                       matlab.ui.control.NumericEditField
        LowerBoundPanel                 matlab.ui.container.Panel
        tbAngleLamina                   matlab.ui.control.EditField
        Label24                         matlab.ui.control.Label
        etAngleLaminaEx                 matlab.ui.control.NumericEditField
        Label25                         matlab.ui.control.Label
        etAngleLaminavxy                matlab.ui.control.NumericEditField
        Label26                         matlab.ui.control.Label
        etAngleLaminaGxy                matlab.ui.control.NumericEditField
        Label27                         matlab.ui.control.Label
        etAngleLaminaEy                 matlab.ui.control.NumericEditField
        Label28                         matlab.ui.control.Label
        etAngleLaminavxz                matlab.ui.control.NumericEditField
        Label29                         matlab.ui.control.Label
        etAngleLaminaGxz                matlab.ui.control.NumericEditField
        Label30                         matlab.ui.control.Label
        etAngleLaminaEz                 matlab.ui.control.NumericEditField
        Label31                         matlab.ui.control.Label
        etAngleLaminavyz                matlab.ui.control.NumericEditField
        Label32                         matlab.ui.control.Label
        etAngleLaminaGyz                matlab.ui.control.NumericEditField
        UpperBoundPanel                 matlab.ui.container.Panel
        tbLamina                        matlab.ui.control.EditField
        Label15                         matlab.ui.control.Label
        etLaminaE1                      matlab.ui.control.NumericEditField
        Label16                         matlab.ui.control.Label
        etLaminav12                     matlab.ui.control.NumericEditField
        Label17                         matlab.ui.control.Label
        etLaminaG12                     matlab.ui.control.NumericEditField
        Label18                         matlab.ui.control.Label
        etLaminaE2                      matlab.ui.control.NumericEditField
```

```
Label19                         matlab.ui.control.Label
etLaminav13                     matlab.ui.control.NumericEditField
Label20                         matlab.ui.control.Label
etLaminaG13                     matlab.ui.control.NumericEditField
Label21                         matlab.ui.control.Label
etLaminaE3                      matlab.ui.control.NumericEditField
Label22                         matlab.ui.control.Label
etLaminav23                     matlab.ui.control.NumericEditField
Label23                         matlab.ui.control.Label
etLaminaG23                     matlab.ui.control.NumericEditField
YarnPropertiesPanel             matlab.ui.container.Panel
Label6                          matlab.ui.control.Label
etYarnE1                        matlab.ui.control.NumericEditField
Label7                          matlab.ui.control.Label
etYarnv12                       matlab.ui.control.NumericEditField
Label8                          matlab.ui.control.Label
etYarnG12                       matlab.ui.control.NumericEditField
Label9                          matlab.ui.control.Label
etYarnE2                        matlab.ui.control.NumericEditField
Label10                         matlab.ui.control.Label
etYarnv13                       matlab.ui.control.NumericEditField
Label11                         matlab.ui.control.Label
etYarnG13                       matlab.ui.control.NumericEditField
Label12                         matlab.ui.control.Label
etYarnE3                        matlab.ui.control.NumericEditField
Label13                         matlab.ui.control.Label
etYarnv23                       matlab.ui.control.NumericEditField
Label14                         matlab.ui.control.Label
etYarnG23                       matlab.ui.control.NumericEditField
YarnPackingFractionLabel        matlab.ui.control.Label
etYarnPackingFraction           matlab.ui.control.NumericEditField
YarnRadiusLabel                 matlab.ui.control.Label
etYarnRadius                    matlab.ui.control.NumericEditField
BraidProperties                 matlab.ui.container.Panel
BraidCrossSectionalDimsEditFieldLabel   matlab.ui.control.Label
BraidCrossSectionalDimsEditField  matlab.ui.control.EditField
UnitCellHeigthEditFieldLabel    matlab.ui.control.Label
UnitCellHeigthEditField         matlab.ui.control.EditField
AverageBraidingAngleEditFieldLabel  matlab.ui.control.Label
AverageBraidingAngleEditField   matlab.ui.control.EditField
BraidingTightnessEditFieldLabel  matlab.ui.control.Label
BraidingTightnessEditField      matlab.ui.control.EditField
FiberVolumeFractionEditFieldLabel  matlab.ui.control.Label
FiberVolumeFractionEditField    matlab.ui.control.EditField
NumberofYarnsEditFieldLabel     matlab.ui.control.Label
NumberofYarnsEditField          matlab.ui.control.EditField
YoungsModulusRangeEditFieldLabel  matlab.ui.control.Label
YoungsModulusRangeEditField     matlab.ui.control.EditField
InteriorBraidingAngleEditFieldLabel  matlab.ui.control.Label
InteriorBraidingAngleEditField  matlab.ui.control.EditField
SurfaceBraidingAngleEditFieldLabel  matlab.ui.control.Label
SurfaceBraidingAngleEditField   matlab.ui.control.EditField
FGMPanel                        matlab.ui.container.Panel
Label24_2                       matlab.ui.control.Label
etFGMEx                         matlab.ui.control.NumericEditField
Label25_2                       matlab.ui.control.Label
etFGMvxy                        matlab.ui.control.NumericEditField
Label26_2                       matlab.ui.control.Label
etFGMGxy                        matlab.ui.control.NumericEditField
Label27_2                       matlab.ui.control.Label
etFGMEy                         matlab.ui.control.NumericEditField
Label28_2                       matlab.ui.control.Label
etFGMvxz                        matlab.ui.control.NumericEditField
Label29_2                       matlab.ui.control.Label
etFGMGxz                        matlab.ui.control.NumericEditField
Label30_2                       matlab.ui.control.Label
etFGMEz                         matlab.ui.control.NumericEditField
Label31_2                       matlab.ui.control.Label
```

```
        etFGMvyz                        matlab.ui.control.NumericEditField
        Label32_2                       matlab.ui.control.Label
        etFGMGyz                        matlab.ui.control.NumericEditField
        BraidAngleAverageLabel          matlab.ui.control.Label
        etTAngle                        matlab.ui.control.NumericEditField
        AnglesfromGeometryPanel         matlab.ui.container.Panel
        ApproxAverageAngleLabel         matlab.ui.control.Label
        etAverageAngle                  matlab.ui.control.NumericEditField
        ApproxInteriorAngleLabel        matlab.ui.control.Label
        etInteriorAngle                 matlab.ui.control.NumericEditField
        ApproxSurfaceAngleLabel         matlab.ui.control.Label
        etSurfaceAngle                  matlab.ui.control.NumericEditField
        ButtonGroup                     matlab.ui.container.ButtonGroup
        enablePitch                     matlab.ui.control.RadioButton
        enableAngle                     matlab.ui.control.RadioButton
    end


    properties (Access = private)
        Knots = 10; % Description
        Path = 0.0;% Description
        Denier = 1500;
        Density = 1.32629;
        Radius = 0.2;
        braidLength = 3; % Description
        braidWidth = 3;% Description
        unitCells = 3;% Description
        Spacing_Factor = 1;% Description
        enableSmoothing = true;% Description
        enableYarns = false;% Description
        folderName = 'default';
        Vf = 1;
        yarnAngle = 0;
        curDir
        Materials
        FGMData
    end

    methods (Access = private)

        function [] = updateLamina(app)

            Em = app.etMatrixE.Value;
            vm = app.etMatrixv.Value;
            Gm = app.etMatrixG.Value;

            E1 = app.etYarnE1.Value;
            E2 = app.etYarnE2.Value;
            v12 = app.etYarnv12.Value;
            G12 = app.etYarnG12.Value;

            theta = app.yarnAngle;
            Vm = 1-app.Vf;

            kf = E1/(2*(1+v12)*(1-2*v12));
            km = Em/(2*(1+vm)*(1-2*vm));
            k  = (km*(kf+Gm)*Vm + kf*(km+Gm)*app.Vf)/...
                ((kf+Gm)*Vm + (km+Gm)*app.Vf);

            E1 = app.Vf*E1 + Vm*Em;
            E2 = 1/(app.Vf/E2 + Vm/Em);
            E3 = E2;

            v12 = app.Vf*v12 + Vm*vm;
            v13 = v12;

            m = 1+4*k*v12^2/E1;
```

```
v23 = @(G23) (k - m*G23)/(k + m*G23);

G12 = 1/(app.Vf/G12 + Vm/Gm);
G13 = G12;
G23 = max(double(solve(@(G23) 2*(1-v23(G23))*G23 - E2)));

v23 = v23(G23);


app.etLaminaE1.Value = E1;
app.etLaminav12.Value = v12;
app.etLaminaG12.Value = G12;
app.etLaminaE2.Value = E2;
app.etLaminav13.Value = v13;
app.etLaminaG13.Value = G13;
app.etLaminaE3.Value = E3;
app.etLaminav23.Value = v23;
app.etLaminaG23.Value = G23;

E1 = app.etYarnE1.Value;
E2 = app.etYarnE2.Value;
v12 = app.etYarnv12.Value;
G12 = app.etYarnG12.Value;
Vf = app.Vf*cosd(theta);
Vm = 1-Vf;

kf = E1/(2*(1+v12)*(1-2*v12));
km = Em/(2*(1+vm)*(1-2*vm));
k  = (km*(kf+Gm)*Vm + kf*(km+Gm)*Vf)/...
     ((kf+Gm)*Vm + (km+Gm)*Vf);

E1 = Vf*E1 + Vm*Em;
E2 = 1/(Vf/E2 + Vm/Em);
E3 = E2;

v12 = Vf*v12 + Vm*vm;
v13 = v12;

m = 1+4*k*v12^2/E1;

v23 = @(G23) (k - m*G23)/(k + m*G23);

G12 = 1/(Vf/G12 + Vm/Gm);
G13 = G12;
G23 = max(double(solve(@(G23) 2*(1-v23(G23))*G23 - E2)));

v23 = v23(G23);

T = @(angle)[...
     cosd(angle)^2 sind(angle)^2 0 0 0 2*sind(angle)*cosd(angle)
     sind(angle)^2 cosd(angle)^2 0 0 0 -2*sind(angle)*cosd(angle)
     0 0 1 0 0 0
     0 0 0 cosd(angle) sind(angle) 0
     0 0 0 -sind(angle) cosd(angle) 0
     -sind(angle)*cosd(angle) sind(angle)*cosd(angle) 0 0 0...
     cosd(angle)^2-sind(angle)^2];

R = diag([1 1 1 2 2 2]);

S = [...
     1/E1     -v12/E1   -v13/E1      0         0         0
     -v12/E1     1/E2    -v23/E2      0         0         0
     -v13/E1  -v23/E2      1/E3       0         0         0
     0           0          0       1/G23       0         0
     0           0          0         0       1/G13       0
     0           0          0         0         0       1/G12];

Sb = T(theta)'*S*T(theta);
```

```
            E.x = 1/Sb(1,1);
            E.y = 1/Sb(2,2);
            E.z = 1/Sb(3,3);
            v.xy = -Sb(1,2)*E.x;
            v.yx = -Sb(2,1)*E.y;
            v.xz = -Sb(1,3)*E.x;
            v.zx = -Sb(3,1)*E.z;
            v.yz = -Sb(2,3)*E.y;
            v.zy = -Sb(3,2)*E.z;
            G.xy = 1/Sb(6,6);
            G.xz = 1/Sb(5,5);
            G.yz = 1/Sb(4,4);

            app.etAngleLaminaEx.Value = E.x;
            app.etAngleLaminavxy.Value = v.xy;
            app.etAngleLaminaGxy.Value = G.xy;
            app.etAngleLaminaEy.Value = E.y;
            app.etAngleLaminavxz.Value = v.xz;
            app.etAngleLaminaGxz.Value = G.xz;
            app.etAngleLaminaEz.Value = E.z;
            app.etAngleLaminavyz.Value = v.yz;
            app.etAngleLaminaGyz.Value = G.yz;

            app.tbLamina.Value = sprintf('%s/%s %.2f (Vy: %0.3f, PF: %0.3f)',...
                app.ddFiberSelect.Value,...
                app.ddMatrixSelect.Value,...
                0,...
                app.Vf,...
                app.etYarnPackingFraction.Value);
            app.tbAngleLamina.Value = sprintf('%s/%s %.2f (Vy: %0.3f, PF: %0.3f)',...
                app.ddFiberSelect.Value,...
                app.ddMatrixSelect.Value,...
                theta,...
                Vf,...
                app.etYarnPackingFraction.Value);
        end

        function [] = printResults(app,enablePlot)
            if nargin==1
                enablePlot = false;
            else
                enablePlot = true;
            end
            filename = [app.folderName,'\BRAID.PROPERTIES'];
            fID = fopen(filename,'W','native','UTF-8');
            Ef1= app.etFiberE1.Value;
            Ef2= app.etFiberE2.Value;
            vf = app.etFiberv.Value;
            Gf = app.etFiberG.Value;
            Em = app.etMatrixE.Value;
            vm = app.etMatrixv.Value;
            Gm = app.etMatrixG.Value;

            Ey1 = app.etYarnE1.Value;
            vy12 = app.etYarnv12.Value;
            Gy12 = app.etYarnG12.Value;
            Ey2 = app.etYarnE2.Value;
            vy13 = app.etYarnv13.Value;
            Gy13 = app.etYarnG13.Value;
            Ey3 = app.etYarnE3.Value;
            vy23 = app.etYarnv23.Value;
            Gy23 = app.etYarnG23.Value;

            E1 = app.etLaminaE1.Value;
            v12 = app.etLaminav12.Value;
            G12 = app.etLaminaG12.Value;
            E2 = app.etLaminaE2.Value;
```

```
v13 = app.etLaminav13.Value;
G13 = app.etLaminaG13.Value;
E3 = app.etLaminaE3.Value;
v23 = app.etLaminav23.Value;
G23 = app.etLaminaG23.Value;

E.x = app.etAngleLaminaEx.Value;
v.xy = app.etAngleLaminavxy.Value;
G.xy = app.etAngleLaminaGxy.Value;
E.y = app.etAngleLaminaEy.Value;
v.xz = app.etAngleLaminavxz.Value;
G.xz = app.etAngleLaminaGxz.Value;
E.z = app.etAngleLaminaEz.Value;
v.yz = app.etAngleLaminavyz.Value;
G.yz = app.etAngleLaminaGyz.Value;

theta = app.yarnAngle;
bn = app.braidWidth;
bm = app.braidLength;
SF = app.Spacing_Factor+app.enableYarns*...
    (1/sind(45+app.Path)-1.0+app.Path/180);
Pf = app.etYarnPackingFraction.Value;
Fiber  = upper(app.ddFiberSelect.Value);
Matrix = upper(app.ddMatrixSelect.Value);
yarnRadius = app.Radius;

Sx = 2*yarnRadius*(2*bn*SF +1.5);
Sy = 2*yarnRadius*(2*bm*SF +1.5);
Ny = (2+app.enableYarns)*bn*bm+bn+bm;
eta = Ny*pi*yarnRadius^2/(Sx*Sy);

app.BraidCrossSectionalDimsEditField.Value = ...
    sprintf('%.3f X %.3f',Sx,Sy);
app.UnitCellHeigthEditField.Value = num2str(app.etZHeight.Value);
app.AverageBraidingAngleEditField.Value = num2str(app.yarnAngle);
app.InteriorBraidingAngleEditField.Value =...
    num2str(atand(4*sqrt(2)*app.Radius*...
    app.Spacing_Factor/app.etZHeight.Value));
app.SurfaceBraidingAngleEditField.Value =...
    num2str(atand(2*sqrt(2)*app.Radius*...
    app.Spacing_Factor/app.etZHeight.Value));
app.BraidingTightnessEditField.Value = num2str(eta);
app.FiberVolumeFractionEditField.Value = num2str(app.Vf*Pf);
app.NumberofYarnsEditField.Value = num2str(Ny);
app.YoungsModulusRangeEditField.Value = ...
    sprintf('%.3f - %.3f',E.x,E1);

app.FGM(Ny, Sx*Sy)

printHeader(fID)

printSummary(fID,Fiber,Matrix,Pf,app.Vf,theta,E.x,E1,...
    app.etFGMEx.Value,bn,bm,SF,yarnRadius,app.Knots,...
    app.enableSmoothing,app.enableYarns,app.etZHeight.Value)

printFiberProps(fID,Ef1,Ef2,vf,Gf,['Fiber Properties: ',Fiber],...
    app.Density,app.Denier,yarnRadius)

printIso(fID, Em,vm,Gm,['Matrix Properties: ' ,Matrix])

printOrtho(fID, Ey1,Ey2,Ey3,vy12,vy13,vy23,Gy12,Gy13,Gy23,...
    sprintf('Yarn Properties: %s/%s (PF: %.3f)',Fiber,Matrix,Pf))

printOrtho(fID, E1,E2,E3,v12,v13,v23,G12,G13,G23,...
    sprintf('%s/%s 0.00 deg (Vy: %.3f, PF: %.3f)',...
    Fiber,Matrix,app.Vf,Pf))

printOrtho(fID,E.x,E.y,E.z,v.xy,v.xz,v.yz,G.xy,G.xz,G.yz,...
        sprintf('%s/%s %.2f deg (Vy: %.3f, PF: %.3f)',...
        Fiber,Matrix,theta,app.Vf,Pf))

    printOrtho(fID,...
        app.etFGMEx.Value,...
        app.etFGMEy.Value,...
        app.etFGMEz.Value,...
        app.etFGMvxy.Value,...
        app.etFGMvxz.Value,...
        app.etFGMvyz.Value,...
        app.etFGMGxy.Value,...
        app.etFGMGxz.Value,...
        app.etFGMGyz.Value,...
        'Fabric Geometery Model')

    fclose(fID);

    if enablePlot
        figure;
        fplot(Pf,[0,90]);
        hold on
        Sx = 2*yarnRadius*(2*bn*SF +1.5);
        Sy = 2*yarnRadius*(2*bm*SF +1.5);
        Ny = (2+app.enableYarns)*bn*bm+bn+bm;
        eta = Ny*pi*yarnRadius^2/(Sx*Sy);
        fplot(@(t)Pf.*eta./cosd(t),[0,acosd(sqrt(eta/(pi/2-eta)))]);
        fplot(@(t)pi./2.*Pf.*cosd(t)./(1+cosd(t).^2),[0,90]);

        plot(theta,Pf*eta/cosd(theta),'ok');
        plot([0,theta],[Pf*eta/cosd(theta),Pf*eta/cosd(theta)],'--k');
        plot([theta,theta],[0,Pf*eta/cosd(theta)],'--k');

        axis([0 90 0 1]);
        xlabel('Braiding Angle, \theta');
        ylabel('Fiber Volume Fraction, V_f');
        text(45,Pf+0.05,sprintf('Yarn Packing Fraction = %.2f',Pf),...
            'HorizontalAlignment','center')
    end

end

function [] = FGM(app,Ny, Area)
    E1 = app.etYarnE1.Value;
    v12 = app.etYarnv12.Value;
    G12 = app.etYarnG12.Value;
    E2 = app.etYarnE2.Value;
    v13 = app.etYarnv13.Value;
    G13 = app.etYarnG13.Value;
    E3 = app.etYarnE3.Value;
    v23 = app.etYarnv23.Value;
    G23 = app.etYarnG23.Value;

    Em = app.etMatrixE.Value;
    vm = app.etMatrixv.Value;
    Gm = app.etMatrixG.Value;

    S  = [...
        1/E1    -v12/E1  -v13/E1     0       0       0
        -v12/E1    1/E2   -v23/E2     0       0       0
        -v13/E1  -v23/E2    1/E3      0       0       0
        0         0         0      1/G23     0       0
        0         0         0        0     1/G13     0
        0         0         0        0       0     1/G12];
    Sm = [...
        1/Em     -vm/Em  -vm/Em     0       0       0
        -vm/Em     1/Em    -vm/Em     0       0       0
        -vm/Em    -vm/Em    1/Em      0       0       0
        0         0         0      1/Gm     0       0
```

```
                0           0           0          0      1/Gm      0
                0           0           0          0        0     1/Gm];

            A = @(theta,beta) [1 0 0; 0 cosd(beta) sind(beta);...
                0 -sind(beta) cosd(beta)]*...
                [cosd(theta) -sind(theta) 0;sind(theta) cosd(theta) 0; 0 0 1];

            Ta = @(theta,beta)A(theta,beta).^2;
            Tb = @(theta,beta)2*circshift(A(theta,beta),1,2).*...
                circshift(A(theta,beta),2,2);
            Tc = @(theta,beta)circshift(A(theta,beta),1,1).*...
                circshift(A(theta,beta),2,1);
            Td = @(theta,beta)circshift(A(theta,beta),[1,1]).*...
                circshift(A(theta,beta),[2,2]) +...
                circshift(A(theta,beta),[1,2]).*circshift(A(theta,beta),[2,1]);
            T  = @(theta,beta) [Ta(theta,beta),Tb(theta,beta);...
                Tc(theta,beta),Td(theta,beta)];

            for m = 1:2
                C = zeros(size(S));
                for n = 1:Ny
                    tFGM = app.FGMData(n+(m-1)*Ny,2);
                    bFGM = app.FGMData(n+(m-1)*Ny,1);
                    C = C + inv(T(tFGM,bFGM)'*S*T(tFGM,bFGM))*...
                        pi*app.Radius^2/cosd(tFGM)/Area;

                end
                C = C + inv(Sm)*(Area - sum(app.Radius./...
                    cosd(app.FGMData((m-1)*Ny+1:m*Ny,2))))/Area;
                switch m
                    case 1
                        C1 = C;
                    case 2
                        C2 = C;
                end
            end
            Snew = inv(C1/2+C2/2);
            Snew(abs(Snew)<1e-5) = 0;

            app.etFGMEx.Value = 1/Snew(1,1);
            app.etFGMEy.Value   = 1/Snew(2,2);
            app.etFGMEz.Value   = 1/Snew(3,3);
            app.etFGMvxy.Value  = -Snew(1,2)*app.etFGMEx.Value;
            app.etFGMGxy.Value  = 1/Snew(6,6);
            app.etFGMvxz.Value  = -Snew(1,3)*app.etFGMEx.Value;
            app.etFGMGxz.Value  = 1/Snew(5,5);
            app.etFGMvyz.Value  = -Snew(2,3)*app.etFGMEy.Value;
            app.etFGMGyz.Value  = 1/Snew(4,4);
        end

    end


    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            app.curDir = cd;

            app.Materials = xmlParse('Materials.lib');

            app.ddFiberSelect.Items = fieldnames(app.Materials.Fiber);

            app.ddMatrixSelect.Items = fieldnames(app.Materials.Matrix);
            value = app.ddFiberSelect.Value;
            E1 = app.Materials.Fiber.(value).E1;
            E2 = app.Materials.Fiber.(value).E2;
            v  = app.Materials.Fiber.(value).v;
            G  = app.Materials.Fiber.(value).G;
            app.Denier = app.Materials.Fiber.(value).Denier;
            app.etDenier.Value = app.Denier;
            app.Density = app.Materials.Fiber.(value).Density;
            app.Radius = sqrt(app.Denier/(9000*pi()*app.Density));
            app.etYarnRadius.Value = app.Radius;
            n = app.braidLength;
            m = app.braidWidth;
            R = app.Radius;
            SF = app.Spacing_Factor;
            z = app.etZHeight.Value;
            app.etAverageAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(0.5*z))+...
                (2*(n+m))/(2*n*m+n+m)*atand((2*sqrt(2)*R*SF)/(z));
            app.etInteriorAngle.Value = atand((2*sqrt(2)*R*SF)/(0.5*z));
            app.etTAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((4*sqrt(2)*R*SF)/(z)) + (2*(n+m))/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(z));
            app.etSurfaceAngle.Value  = atand((2*sqrt(2)*R*SF)/(z));
            app.etFiberE1.Value = E1;
            app.etFiberE2.Value = E2;
            app.etFiberv.Value = v;
            app.etFiberG.Value = G;
            value = app.ddMatrixSelect.Value;
            E = app.Materials.Matrix.(value).E;
            v = app.Materials.Matrix.(value).v;
            G = E/(2*(1+v));

            app.etMatrixE.Value = E;
            app.etMatrixv.Value = v;
            app.etMatrixG.Value = G;
            app.PF()
        end

        % Value changed function: cbDenier
        function DenierChange(app, event)
            value = app.cbDenier.Value;

            if value
                app.etDenier.Editable = 'on';
            else
                app.etDenier.Editable = 'off';
                value = app.ddFiberSelect.Value;
                app.Denier  = app.Materials.Fiber.(value).Denier;
                app.etDenier.Value = app.Materials.Fiber.(value).Denier;
                app.Radius = sqrt(app.Denier/(9000*pi()*app.Density));
                app.etYarnRadius.Value = app.Radius;
                n = app.braidLength;
                m = app.braidWidth;
                R = app.Radius;
                SF = app.Spacing_Factor;
                z = app.etZHeight.Value;
                app.etAverageAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                    atand((2*sqrt(2)*R*SF)/(0.5*z))+...
                    (2*(n+m))/(2*n*m+n+m)*atand((2*sqrt(2)*R*SF)/(z));
                app.etInteriorAngle.Value = atand((2*sqrt(2)*R*SF)/(0.5*z));
                app.etTAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                    atand((4*sqrt(2)*R*SF)/(z)) + (2*(n+m))/(2*n*m+n+m)*...
                    atand((2*sqrt(2)*R*SF)/(z));
                app.etSurfaceAngle.Value  = atand((2*sqrt(2)*R*SF)/(z));
            end

        end

        % Value changed function: etDenier
        function DenierValue(app, event)
            value = round(app.etDenier.Value);
            app.etDenier.Value = value;
            app.Denier = value;
            app.Radius = sqrt(app.Denier/(9000*pi()*app.Density));
            app.etYarnRadius.Value = app.Radius;
            n = app.braidLength;
            m = app.braidWidth;
            R = app.Radius;
            SF = app.Spacing_Factor;
            z = app.etZHeight.Value;
            app.etAverageAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(0.5*z))+...
                (2*(n+m))/(2*n*m+n+m)*atand((2*sqrt(2)*R*SF)/(z));
            app.etInteriorAngle.Value = atand((2*sqrt(2)*R*SF)/(0.5*z));
            app.etTAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((4*sqrt(2)*R*SF)/(z)) + (2*(n+m))/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(z));
            app.etSurfaceAngle.Value  = atand((2*sqrt(2)*R*SF)/(z));
        end

        % Value changed function: etYarnPackingFraction
        function PF(app, event)
            value = app.etYarnPackingFraction.Value;

            Ef1= app.etFiberE1.Value;
            Ef2= app.etFiberE2.Value;
            vf = app.etFiberv.Value;
            Gf = app.etFiberG.Value;
            Em = app.etMatrixE.Value;
            vm = app.etMatrixv.Value;
            Gm = app.etMatrixG.Value;

            Pf = value;

            Vm = 1-Pf;

            kf = Ef1/(2*(1+vf)*(1-2*vf));
            km = Em/(2*(1+vm)*(1-2*vm));
            k  = (km*(kf+Gm)*Vm + kf*(km+Gm)*Pf)/...
                ((kf+Gm)*Vm + (km+Gm)*Pf);

            E1 = Pf*Ef1 + Vm*Em;
            E2 = 1/(Pf/Ef2 + Vm/Em);
            E3 = E2;

            v12 = Pf*vf + Vm*vm;
            v13 = v12;

            m = 1+4*k*v12^2/E1;

            v23 = @(G23) (k - m*G23)/(k + m*G23);

            G12 = 1/(Pf/Gf + Vm/Gm);
            G13 = G12;
            G23 = max(double(solve(@(G23) 2*(1-v23(G23))*G23 - E2)));

            v23 = v23(G23);
            app.etYarnE1.Value = E1;
            app.etYarnv12.Value = v12;
            app.etYarnG12.Value = G12;
            app.etYarnE2.Value = E2;
            app.etYarnv13.Value = v13;
            app.etYarnG13.Value = G13;
            app.etYarnE3.Value = E3;
            app.etYarnv23.Value = v23;
            app.etYarnG23.Value = G23;
            app.updateLamina()
        end

        % Value changed function: etTAngle
```

```
        function TAngleChange(app, event)
            value = app.etTAngle.Value;
            n = app.braidLength;
            m = app.braidWidth;
            R = app.Radius;
            SF = app.Spacing_Factor;
            z = 4*sqrt(2)*R*SF/tand(value);
            app.etZHeight.Value = z;
            app.UnitCellHeigthEditField.Value = num2str(app.etZHeight.Value);
            app.etAverageAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(0.5*z))+...
                (2*(n+m))/(2*n*m+n+m)*atand((2*sqrt(2)*R*SF)/(z));
            app.etInteriorAngle.Value = atand((2*sqrt(2)*R*SF)/(0.5*z));
            app.etSurfaceAngle.Value  = atand((2*sqrt(2)*R*SF)/(z));
        end

        % Value changed function: etBraidDepth
        function braidL(app, event)
            value = app.etBraidDepth.Value;
            app.braidLength = round(value);
            bm = app.braidLength;
            bn = app.braidWidth;
            SF = app.Spacing_Factor;
            Sx = 2*app.Radius*(2*bn*SF +1.5);
            Sy = 2*app.Radius*(2*bm*SF +1.5);
            app.BraidCrossSectionalDimsEditField.Value = ...
                sprintf('%.3f X %.3f',Sx,Sy);
            app.NumberofYarnsEditField.Value = ...
                num2str((2+app.enableYarns)*bn*bm+bn+bm);

            n = app.braidLength;
            m = app.braidWidth;
            R = app.Radius;
            SF = app.Spacing_Factor;
            z = app.etZHeight.Value;
            app.etAverageAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(0.5*z))+...
                (2*(n+m))/(2*n*m+n+m)*atand((2*sqrt(2)*R*SF)/(z));
            app.etInteriorAngle.Value = atand((2*sqrt(2)*R*SF)/(0.5*z));
            app.etTAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((4*sqrt(2)*R*SF)/(z)) + (2*(n+m))/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(z));
            app.etSurfaceAngle.Value  = atand((2*sqrt(2)*R*SF)/(z));
        end

        % Value changed function: etBraidWidth
        function braidW(app, event)
            value = app.etBraidWidth.Value;
            app.braidWidth = round(value);
            bm = app.braidLength;
            bn = app.braidWidth;
            SF = app.Spacing_Factor;
            Sx = 2*app.Radius*(2*bn*SF +1.5);
            Sy = 2*app.Radius*(2*bm*SF +1.5);
            app.BraidCrossSectionalDimsEditField.Value =...
                sprintf('%.3f X %.3f',Sx,Sy);
            app.NumberofYarnsEditField.Value = ...
                num2str((2+app.enableYarns)*bn*bm+bn+bm);

            n = app.braidLength;
            m = app.braidWidth;
            R = app.Radius;
            SF = app.Spacing_Factor;
            z = app.etZHeight.Value;
            app.etAverageAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(0.5*z))+...
                (2*(n+m))/(2*n*m+n+m)*atand((2*sqrt(2)*R*SF)/(z));
            app.etInteriorAngle.Value = atand((2*sqrt(2)*R*SF)/(0.5*z));
            app.etTAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((4*sqrt(2)*R*SF)/(z)) + (2*(n+m))/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(z));
            app.etSurfaceAngle.Value  = atand((2*sqrt(2)*R*SF)/(z));
        end

        % Button pushed function: btClosePlots
        function closePlots(app, event)
            close all;
        end

        % Value changed function: cbSmoothing
        function enSmoothing(app, event)
            value = app.cbSmoothing.Value;
            app.enableSmoothing = value;
        end

        % Value changed function: cbTriAxial
        function enTriAxial(app, event)
            value = app.cbTriAxial.Value;
            app.enableYarns = value;
            bm = app.braidLength;
            bn = app.braidWidth;
            SF = app.Spacing_Factor;
            Sx = 2*app.Radius*(2*bn*SF +1.5);
            Sy = 2*app.Radius*(2*bm*SF +1.5);
            app.BraidCrossSectionalDimsEditField.Value =...
                sprintf('%.3f X %.3f',Sx,Sy);
            app.NumberofYarnsEditField.Value = ...
                num2str((2+app.enableYarns)*bn*bm+bn+bm);
        end

        % Value changed function: ddFiberSelect
        function fiberChange(app, event)
            value = app.ddFiberSelect.Value;
            E1 = app.Materials.Fiber.(value).E1;
            E2 = app.Materials.Fiber.(value).E2;
            v  = app.Materials.Fiber.(value).v;
            G  = app.Materials.Fiber.(value).G;
            D  = app.Materials.Fiber.(value).Denier;
            app.etDenier.Value = D;
            rho= app.Materials.Fiber.(value).Density;
            app.etFiberE1.Value  = E1;
            app.etFiberE2.Value = E2;
            app.etFiberv.Value = v;
            app.etFiberG.Value = G;
            app.Density = rho;
            app.Denier = D;
            app.Radius = sqrt(app.Denier/(9000*pi()*app.Density));
            app.etYarnRadius.Value = app.Radius;
            bm = app.braidLength;
            bn = app.braidWidth;
            SF = app.Spacing_Factor;
            Sx = 2*app.Radius*(2*bn*SF +1.5);
            Sy = 2*app.Radius*(2*bm*SF +1.5);
            app.BraidCrossSectionalDimsEditField.Value = ...
                sprintf('%.3f X %.3f',Sx,Sy);
            n = app.braidLength;
            m = app.braidWidth;
            R = app.Radius;
            SF = app.Spacing_Factor;
            z = app.etZHeight.Value;
            app.etAverageAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(0.5*z))+(2*(n+m))/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(z));
            app.etInteriorAngle.Value = atand((2*sqrt(2)*R*SF)/(0.5*z));
            app.etTAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((4*sqrt(2)*R*SF)/(z)) + (2*(n+m))/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(z));
            app.etSurfaceAngle.Value  = atand((2*sqrt(2)*R*SF)/(z));
            app.PF()
        end

        % Value changed function: etFolderName
        function folder(app, event)
            value = app.etFolderName.Value;
            if isempty(value)
                app.etFolderName.Value = app.folderName;
            else
                app.folderName = value;
            end
        end

        % Button pushed function: btGenerateModel
        function genModel(app, event)
            if app.cbType2Paths.Value
                [~,app.Vf,app.yarnAngle,app.FGMData]  =...
                    generateBraidPaths(app.Knots,app.Path,app.braidLength,...
                    app.braidWidth,app.Radius,app.Spacing_Factor,...
                    app.unitCells,app.folderName,app.enableSmoothing,...
                    app.enableYarns,false,false,true,app.etZHeight.Value);
            else
                [~,app.Vf,app.yarnAngle,app.FGMData]  =...
                    generateBraidPaths(app.Knots,app.Path,app.braidLength,...
                    app.braidWidth,app.Radius,app.Spacing_Factor,...
                    app.unitCells,app.folderName,app.enableSmoothing,...
                    app.enableYarns,false,false,false,app.etZHeight.Value);
            end
            app.updateLamina();
            app.printResults();
            system(['copy "',app.curDir,'\Import_Curves.swp" "',...
                app.curDir,'\',app.folderName,'\Import_Curves.swp"'])
            system(['"C:\Program Files\SOLIDWORKS Corp\',...
                'SOLIDWORKS (2)\SLDWORKS.exe" -m "',...
                app.curDir,'\',app.folderName,'\Import_Curves.swp"'])
        end

        % Button pushed function: btGeneratePaths
        function genPaths(app, event)
            if app.cbType2Paths.Value
                [~,app.Vf,app.yarnAngle,app.FGMData]  =...
                    generateBraidPaths(app.Knots,app.Path,app.braidLength,...
                    app.braidWidth,app.Radius,app.Spacing_Factor,...
                    app.unitCells,app.folderName,app.enableSmoothing,...
                    app.enableYarns,false,false,true,app.etZHeight.Value);
            else
                [~,app.Vf,app.yarnAngle,app.FGMData]  =...
                    generateBraidPaths(app.Knots,app.Path,app.braidLength,...
                    app.braidWidth,app.Radius,app.Spacing_Factor,...
                    app.unitCells,app.folderName,app.enableSmoothing,...
                    app.enableYarns,false,false,false,app.etZHeight.Value);
            end
            app.updateLamina();
            app.printResults();
        end

        % Button pushed function: btGeneratePlots
        function genPlots(app, event)
            close all;
            if app.cbType2Paths.Value
                [~,app.Vf,app.yarnAngle,app.FGMData]  =...
                    generateBraidPaths(app.Knots,app.Path,app.braidLength,...
                    app.braidWidth,app.Radius,app.Spacing_Factor,...
                    app.unitCells,app.folderName,app.enableSmoothing,...
                    app.enableYarns,false,true,true,app.etZHeight.Value);
            else
                [~,app.Vf,app.yarnAngle,app.FGMData]  =...
                    generateBraidPaths(app.Knots,app.Path,app.braidLength,...
                    app.braidWidth,app.Radius,app.Spacing_Factor,...
                    app.unitCells,app.folderName,app.enableSmoothing,...
                    app.enableYarns,false,true,false,app.etZHeight.Value);
            end
            app.updateLamina();
            app.printResults(1);
        end

        % Value changed function: ddMatrixSelect
        function matrixChange(app, event)
            value = app.ddMatrixSelect.Value;
            E = app.Materials.Matrix.(value).E;
            v = app.Materials.Matrix.(value).v;
            G = E/(2*(1+v));

            app.etMatrixE.Value = E;
            app.etMatrixv.Value = v;
            app.etMatrixG.Value = G;
            app.PF()
        end

        % Selection changed function: ButtonGroup
        function modeChange(app, event)
            selectedButton = app.ButtonGroup.SelectedObject;
            switch selectedButton
                case app.enablePitch
                    app.etZHeight.Editable = 'on';
                    app.etTAngle.Editable  = 'off';
                case app.enableAngle
                    app.etZHeight.Editable = 'off';
                    app.etTAngle.Editable  = 'on';
            end
        end

        % Value changed function: etPathAngle
        function pathAngle(app, event)
            value = app.etPathAngle.Value;
            app.Path = value;
        end

        % Value changed function: etSpacingFactor
        function sf(app, event)
            value = app.etSpacingFactor.Value;
            app.Spacing_Factor = value;
            bm = app.braidLength;
            bn = app.braidWidth;
            SF = app.Spacing_Factor;
            Sx = 2*app.Radius*(2*bn*SF +1.5);
            Sy = 2*app.Radius*(2*bm*SF +1.5);
            app.BraidCrossSectionalDimsEditField.Value = ...
                sprintf('%.3f X %.3f',Sx,Sy);
            n = app.braidLength;
            m = app.braidWidth;
            R = app.Radius;
            SF = app.Spacing_Factor;
            z = app.etZHeight.Value;
            app.etAverageAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(0.5*z))+...
                (2*(n+m))/(2*n*m+n+m)*atand((2*sqrt(2)*R*SF)/(z));
            app.etInteriorAngle.Value = atand((2*sqrt(2)*R*SF)/(0.5*z));
            app.etTAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((4*sqrt(2)*R*SF)/(z)) + (2*(n+m))/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(z));
            app.etSurfaceAngle.Value  = atand((2*sqrt(2)*R*SF)/(z));
        end
```

```
        % Value changed function: etKnotsinSpline
        function splineKnots(app, event)
            value = app.etKnotsinSpline.Value;
            app.Knots = round(value);
        end

        % Value changed function: etNumofUnitCells
        function uc(app, event)
            value = app.etNumofUnitCells.Value;
            app.unitCells = round(value);
        end

        % Value changed function: etZHeight
        function zChange(app, event)
            value = app.etZHeight.Value;
            n = app.braidLength;
            m = app.braidWidth;
            R = app.Radius;
            SF = app.Spacing_Factor;
            z = app.etZHeight.Value;
            app.UnitCellHeigthEditField.Value = num2str(app.etZHeight.Value);
            app.etAverageAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(0.5*z))+...
                (2*(n+m))/(2*n*m+n+m)*atand((2*sqrt(2)*R*SF)/(z));
            app.etInteriorAngle.Value = atand((4*sqrt(2)*R*SF)/(z));
            app.etTAngle.Value = (2*n*m-n-m)/(2*n*m+n+m)*...
                atand((4*sqrt(2)*R*SF)/(z)) + (2*(n+m))/(2*n*m+n+m)*...
                atand((2*sqrt(2)*R*SF)/(z));
            app.etSurfaceAngle.Value  = atand((2*sqrt(2)*R*SF)/(z));
        end
    end

    % App initialization and construction
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [101 101 1425 871];
            app.UIFigure.Name = 'Braid Generator';

            % Create LabelNumericEditField2
            app.LabelNumericEditField2 = uilabel(app.UIFigure);
            app.LabelNumericEditField2.HorizontalAlignment = 'right';
            app.LabelNumericEditField2.Position = [95 745 60 15];
            app.LabelNumericEditField2.Text = 'Path Angle';

            % Create etPathAngle
            app.etPathAngle = uieditfield(app.UIFigure, 'numeric');
            app.etPathAngle.ValueChangedFcn = ...
                createCallbackFcn(app, @pathAngle, true);
            app.etPathAngle.Limits = [0 45];
            app.etPathAngle.ValueDisplayFormat = '%.1f';
            app.etPathAngle.HorizontalAlignment = 'center';
            app.etPathAngle.Position = [170 741 100 22];

            % Create LabelNumericEditField3
            app.LabelNumericEditField3 = uilabel(app.UIFigure);
            app.LabelNumericEditField3.HorizontalAlignment = 'right';
            app.LabelNumericEditField3.Position = [92 702 63 15];
            app.LabelNumericEditField3.Text = 'Braid Width';

            % Create etBraidWidth
            app.etBraidWidth = uieditfield(app.UIFigure, 'numeric');
            app.etBraidWidth.ValueChangedFcn = ...
                createCallbackFcn(app, @braidW, true);
```

```
app.etBraidWidth.Limits = [1 20];
app.etBraidWidth.ValueDisplayFormat = ’%.0f’;
app.etBraidWidth.HorizontalAlignment = ’center’;
app.etBraidWidth.Position = [170 698 100 22];
app.etBraidWidth.Value = 3;

% Create LabelNumericEditField4
app.LabelNumericEditField4 = uilabel(app.UIFigure);
app.LabelNumericEditField4.HorizontalAlignment = ’right’;
app.LabelNumericEditField4.Position = [73 788 82 15];
app.LabelNumericEditField4.Text = ’Knots in Spline’;

% Create etKnotsinSpline
app.etKnotsinSpline = uieditfield(app.UIFigure, ’numeric’);
app.etKnotsinSpline.ValueChangedFcn =...
    createCallbackFcn(app, @splineKnots, true);
app.etKnotsinSpline.Limits = [5 1000];
app.etKnotsinSpline.ValueDisplayFormat = ’%.0f’;
app.etKnotsinSpline.HorizontalAlignment = ’center’;
app.etKnotsinSpline.Position = [170 784 100 22];
app.etKnotsinSpline.Value = 10;

% Create LabelNumericEditField5
app.LabelNumericEditField5 = uilabel(app.UIFigure);
app.LabelNumericEditField5.HorizontalAlignment = ’right’;
app.LabelNumericEditField5.Position = [90 659 65 15];
app.LabelNumericEditField5.Text = ’Braid Depth’;

% Create etBraidDepth
app.etBraidDepth = uieditfield(app.UIFigure, ’numeric’);
app.etBraidDepth.ValueChangedFcn = ...
    createCallbackFcn(app, @braidL, true);
app.etBraidDepth.Limits = [1 20];
app.etBraidDepth.ValueDisplayFormat = ’%.0f’;
app.etBraidDepth.HorizontalAlignment = ’center’;
app.etBraidDepth.Position = [170 655 100 22];
app.etBraidDepth.Value = 3;

% Create LabelNumericEditField6
app.LabelNumericEditField6 = uilabel(app.UIFigure);
app.LabelNumericEditField6.HorizontalAlignment = ’right’;
app.LabelNumericEditField6.Position = [73 617 82 15];
app.LabelNumericEditField6.Text = ’Spacing Factor’;

% Create etSpacingFactor
app.etSpacingFactor = uieditfield(app.UIFigure, ’numeric’);
app.etSpacingFactor.ValueChangedFcn = ...
    createCallbackFcn(app, @sf, true);
app.etSpacingFactor.Limits = [0.5 50];
app.etSpacingFactor.ValueDisplayFormat = ’%.2f’;
app.etSpacingFactor.HorizontalAlignment = ’center’;
app.etSpacingFactor.Position = [170 613 100 22];
app.etSpacingFactor.Value = 1;

% Create btGeneratePlots
app.btGeneratePlots = uibutton(app.UIFigure, ’push’);
app.btGeneratePlots.ButtonPushedFcn = ...
    createCallbackFcn(app, @genPlots, true);
app.btGeneratePlots.Position = [163 416 120 42];
app.btGeneratePlots.Text = ’Generate Plots’;

% Create btGenerateModel
app.btGenerateModel = uibutton(app.UIFigure, ’push’);
app.btGenerateModel.ButtonPushedFcn = ...
    createCallbackFcn(app, @genModel, true);
app.btGenerateModel.Position = [307 416 120 42];
app.btGenerateModel.Text = ’Generate Model’;
```

```
% Create LabelNumericEditField7
app.LabelNumericEditField7 = uilabel(app.UIFigure);
app.LabelNumericEditField7.HorizontalAlignment = 'right';
app.LabelNumericEditField7.Position = [42 575 113 15];
app.LabelNumericEditField7.Text = 'Number of Unit Cells';

% Create etNumofUnitCells
app.etNumofUnitCells = uieditfield(app.UIFigure, 'numeric');
app.etNumofUnitCells.ValueChangedFcn = ...
    createCallbackFcn(app, @uc, true);
app.etNumofUnitCells.Limits = [1 20];
app.etNumofUnitCells.ValueDisplayFormat = '%.0f';
app.etNumofUnitCells.HorizontalAlignment = 'center';
app.etNumofUnitCells.Position = [170 571 100 22];
app.etNumofUnitCells.Value = 3;

% Create cbSmoothing
app.cbSmoothing = uicheckbox(app.UIFigure);
app.cbSmoothing.ValueChangedFcn = ...
    createCallbackFcn(app, @enSmoothing, true);
app.cbSmoothing.Text = 'Enable Smoothing';
app.cbSmoothing.Position = [392 768 120 16];
app.cbSmoothing.Value = true;

% Create cbTriAxial
app.cbTriAxial = uicheckbox(app.UIFigure);
app.cbTriAxial.ValueChangedFcn = ...
    createCallbackFcn(app, @enTriAxial, true);
app.cbTriAxial.Text = 'Enable Tri-Axial Yarns ';
app.cbTriAxial.Position = [392 728 144 16];

% Create btGeneratePaths
app.btGeneratePaths = uibutton(app.UIFigure, 'push');
app.btGeneratePaths.ButtonPushedFcn = ...
    createCallbackFcn(app, @genPaths, true);
app.btGeneratePaths.Position = [18 416 120 42];
app.btGeneratePaths.Text = 'Generate Paths';

% Create LabelEditField
app.LabelEditField = uilabel(app.UIFigure);
app.LabelEditField.HorizontalAlignment = 'right';
app.LabelEditField.Position = [357 634 74 15];
app.LabelEditField.Text = 'Folder Output';

% Create etFolderName
app.etFolderName = uieditfield(app.UIFigure, 'text');
app.etFolderName.ValueChangedFcn = ...
    createCallbackFcn(app, @folder, true);
app.etFolderName.HorizontalAlignment = 'center';
app.etFolderName.Position = [446 630 100 22];
app.etFolderName.Value = 'default';

% Create btClosePlots
app.btClosePlots = uibutton(app.UIFigure, 'push');
app.btClosePlots.ButtonPushedFcn = ...
    createCallbackFcn(app, @closePlots, true);
app.btClosePlots.Position = [451 416 120 42];
app.btClosePlots.Text = 'Close Plots';

% Create cbType2Paths
app.cbType2Paths = uicheckbox(app.UIFigure);
app.cbType2Paths.Enable = 'off';
app.cbType2Paths.Text = 'Enable Type II';
app.cbType2Paths.Position = [392 686 97 16];

% Create LabelDropDown
app.LabelDropDown = uilabel(app.UIFigure);
app.LabelDropDown.Position = [624 762 28 15];
app.LabelDropDown.Text = 'Fiber';

% Create ddFiberSelect
app.ddFiberSelect = uidropdown(app.UIFigure);
app.ddFiberSelect.Items = {'Option 1'};
app.ddFiberSelect.ValueChangedFcn = ...
    createCallbackFcn(app, @fiberChange, true);
app.ddFiberSelect.Position = [667 758 100 22];

% Create LabelDropDown2
app.LabelDropDown2 = uilabel(app.UIFigure);
app.LabelDropDown2.HorizontalAlignment = 'right';
app.LabelDropDown2.Position = [800 763 33 15];
app.LabelDropDown2.Text = 'Matrix';

% Create ddMatrixSelect
app.ddMatrixSelect = uidropdown(app.UIFigure);
app.ddMatrixSelect.Items = {'Option 1'};
app.ddMatrixSelect.ValueChangedFcn = ...
    createCallbackFcn(app, @matrixChange, true);
app.ddMatrixSelect.Position = [848 759 100 22];

% Create LabelNumericEditField9
app.LabelNumericEditField9 = uilabel(app.UIFigure);
app.LabelNumericEditField9.HorizontalAlignment = 'center';
app.LabelNumericEditField9.Position = [661.5 712 111 15];
app.LabelNumericEditField9.Text = 'Logitudinal Modulus';

% Create etFiberE1
app.etFiberE1 = uieditfield(app.UIFigure, 'numeric');
app.etFiberE1.Editable = 'off';
app.etFiberE1.HorizontalAlignment = 'center';
app.etFiberE1.Position = [667 680 100 22];

% Create Label
app.Label = uilabel(app.UIFigure);
app.Label.HorizontalAlignment = 'right';
app.Label.Position = [675 600 84 15];
app.Label.Text = 'Poisson''s Ratio';

% Create etFiberv
app.etFiberv = uieditfield(app.UIFigure, 'numeric');
app.etFiberv.Editable = 'off';
app.etFiberv.HorizontalAlignment = 'center';
app.etFiberv.Position = [667 568 100 22];

% Create Label2
app.Label2 = uilabel(app.UIFigure);
app.Label2.HorizontalAlignment = 'right';
app.Label2.Position = [675 542 83 15];
app.Label2.Text = 'Shear Modulus';

% Create etFiberG
app.etFiberG = uieditfield(app.UIFigure, 'numeric');
app.etFiberG.Editable = 'off';
app.etFiberG.HorizontalAlignment = 'center';
app.etFiberG.Position = [667 510 100 22];

% Create Label3
app.Label3 = uilabel(app.UIFigure);
app.Label3.HorizontalAlignment = 'right';
app.Label3.Position = [851 713 94 15];
app.Label3.Text = 'Young''s Modulus';

% Create etMatrixE
app.etMatrixE = uieditfield(app.UIFigure, 'numeric');
app.etMatrixE.Editable = 'off';
app.etMatrixE.HorizontalAlignment = 'center';
app.etMatrixE.Position = [848 681 100 22];

% Create Label4
app.Label4 = uilabel(app.UIFigure);
app.Label4.HorizontalAlignment = 'right';
app.Label4.Position = [856 630 84 15];
app.Label4.Text = 'Poisson''s Ratio';

% Create etMatrixv
app.etMatrixv = uieditfield(app.UIFigure, 'numeric');
app.etMatrixv.Editable = 'off';
app.etMatrixv.HorizontalAlignment = 'center';
app.etMatrixv.Position = [848 598 100 22];

% Create Label5
app.Label5 = uilabel(app.UIFigure);
app.Label5.HorizontalAlignment = 'right';
app.Label5.Position = [856 543 83 15];
app.Label5.Text = 'Shear Modulus';

% Create etMatrixG
app.etMatrixG = uieditfield(app.UIFigure, 'numeric');
app.etMatrixG.Editable = 'off';
app.etMatrixG.HorizontalAlignment = 'center';
app.etMatrixG.Position = [848 511 100 22];

% Create Label33
app.Label33 = uilabel(app.UIFigure);
app.Label33.HorizontalAlignment = 'center';
app.Label33.Position = [661.5 654 111 15];
app.Label33.Text = 'Transverse Modulus';

% Create etFiberE2
app.etFiberE2 = uieditfield(app.UIFigure, 'numeric');
app.etFiberE2.Editable = 'off';
app.etFiberE2.HorizontalAlignment = 'center';
app.etFiberE2.Position = [667 622 100 22];

% Create cbDenier
app.cbDenier = uicheckbox(app.UIFigure);
app.cbDenier.ValueChangedFcn = ...
    createCallbackFcn(app, @DenierChange, true);
app.cbDenier.Text = 'Override Denier';
app.cbDenier.Position = [662 426 109 15];

% Create DenierEditFieldLabel
app.DenierEditFieldLabel = uilabel(app.UIFigure);
app.DenierEditFieldLabel.HorizontalAlignment = 'center';
app.DenierEditFieldLabel.Position = [696 486 42 15];
app.DenierEditFieldLabel.Text = 'Denier';

% Create etDenier
app.etDenier = uieditfield(app.UIFigure, 'numeric');
app.etDenier.ValueChangedFcn = ...
    createCallbackFcn(app, @DenierValue, true);
app.etDenier.Limits = [1 Inf];
app.etDenier.Editable = 'off';
app.etDenier.HorizontalAlignment = 'center';
app.etDenier.Position = [667 454 100 22];
app.etDenier.Value = 1000;

% Create UnitCellHeightPitchLabel
app.UnitCellHeightPitchLabel = uilabel(app.UIFigure);
app.UnitCellHeightPitchLabel.HorizontalAlignment = 'right';
app.UnitCellHeightPitchLabel.Position = [26 533 129 15];
app.UnitCellHeightPitchLabel.Text = 'Unit Cell Height (Pitch)';

% Create etZHeight
```

```
app.etZHeight = uieditfield(app.UIFigure, 'numeric');
app.etZHeight.ValueChangedFcn = ...
    createCallbackFcn(app, @zChange, true);
app.etZHeight.Limits = [1e-06 1000];
app.etZHeight.HorizontalAlignment = 'center';
app.etZHeight.Position = [170 529 100 22];
app.etZHeight.Value = 2;

% Create LowerBoundPanel
app.LowerBoundPanel = uipanel(app.UIFigure);
app.LowerBoundPanel.Title = 'Lower Bound';
app.LowerBoundPanel.Position = [367 33 291 325];

% Create tbAngleLamina
app.tbAngleLamina = uieditfield(app.LowerBoundPanel, 'text');
app.tbAngleLamina.Editable = 'off';
app.tbAngleLamina.HorizontalAlignment = 'center';
app.tbAngleLamina.BackgroundColor = [0.9373 0.9373 0.9373];
app.tbAngleLamina.Position = [9 257 276 35];

% Create Label24
app.Label24 = uilabel(app.LowerBoundPanel);
app.Label24.HorizontalAlignment = 'center';
app.Label24.VerticalAlignment = 'center';
app.Label24.Position = [47 224 20 15];
app.Label24.Text = 'Ex';

% Create etAngleLaminaEx
app.etAngleLaminaEx = uieditfield(app.LowerBoundPanel, 'numeric');
app.etAngleLaminaEx.Editable = 'off';
app.etAngleLaminaEx.HorizontalAlignment = 'center';
app.etAngleLaminaEx.Position = [25 192 64 22];

% Create Label25
app.Label25 = uilabel(app.LowerBoundPanel);
app.Label25.HorizontalAlignment = 'center';
app.Label25.Position = [47 141 20 15];
app.Label25.Text = 'vxy';

% Create etAngleLaminavxy
app.etAngleLaminavxy = uieditfield(app.LowerBoundPanel, 'numeric');
app.etAngleLaminavxy.Editable = 'off';
app.etAngleLaminavxy.HorizontalAlignment = 'center';
app.etAngleLaminavxy.Position = [25 109 64 22];

% Create Label26
app.Label26 = uilabel(app.LowerBoundPanel);
app.Label26.HorizontalAlignment = 'center';
app.Label26.Position = [46 54 21 15];
app.Label26.Text = 'Gxy';

% Create etAngleLaminaGxy
app.etAngleLaminaGxy = uieditfield(app.LowerBoundPanel, 'numeric');
app.etAngleLaminaGxy.Editable = 'off';
app.etAngleLaminaGxy.HorizontalAlignment = 'center';
app.etAngleLaminaGxy.Position = [25 22 64 22];

% Create Label27
app.Label27 = uilabel(app.LowerBoundPanel);
app.Label27.HorizontalAlignment = 'center';
app.Label27.VerticalAlignment = 'center';
app.Label27.Position = [135 223 20 15];
app.Label27.Text = 'Ey';

% Create etAngleLaminaEy
app.etAngleLaminaEy = uieditfield(app.LowerBoundPanel, 'numeric');
app.etAngleLaminaEy.Editable = 'off';
app.etAngleLaminaEy.HorizontalAlignment = 'center';
```

```
            app.etAngleLaminaEy.Position = [113 191 64 22];

            % Create Label28
            app.Label28 = uilabel(app.LowerBoundPanel);
            app.Label28.HorizontalAlignment = 'center';
            app.Label28.Position = [135 140 20 15];
            app.Label28.Text = 'vxz';

            % Create etAngleLaminavxz
            app.etAngleLaminavxz = uieditfield(app.LowerBoundPanel, 'numeric');
            app.etAngleLaminavxz.Editable = 'off';
            app.etAngleLaminavxz.HorizontalAlignment = 'center';
            app.etAngleLaminavxz.Position = [113 108 64 22];

            % Create Label29
            app.Label29 = uilabel(app.LowerBoundPanel);
            app.Label29.HorizontalAlignment = 'center';
            app.Label29.Position = [134 53 21 15];
            app.Label29.Text = 'Gxz';

            % Create etAngleLaminaGxz
            app.etAngleLaminaGxz = uieditfield(app.LowerBoundPanel, 'numeric');
            app.etAngleLaminaGxz.Editable = 'off';
            app.etAngleLaminaGxz.HorizontalAlignment = 'center';
            app.etAngleLaminaGxz.Position = [113 21 64 22];

            % Create Label30
            app.Label30 = uilabel(app.LowerBoundPanel);
            app.Label30.HorizontalAlignment = 'center';
            app.Label30.VerticalAlignment = 'center';
            app.Label30.Position = [223 223 20 15];
            app.Label30.Text = 'Ez';

            % Create etAngleLaminaEz
            app.etAngleLaminaEz = uieditfield(app.LowerBoundPanel, 'numeric');
            app.etAngleLaminaEz.Editable = 'off';
            app.etAngleLaminaEz.HorizontalAlignment = 'center';
            app.etAngleLaminaEz.Position = [201 191 64 22];

            % Create Label31
            app.Label31 = uilabel(app.LowerBoundPanel);
            app.Label31.HorizontalAlignment = 'center';
            app.Label31.Position = [223 140 20 15];
            app.Label31.Text = 'vyz';

            % Create etAngleLaminavyz
            app.etAngleLaminavyz = uieditfield(app.LowerBoundPanel, 'numeric');
            app.etAngleLaminavyz.Editable = 'off';
            app.etAngleLaminavyz.HorizontalAlignment = 'center';
            app.etAngleLaminavyz.Position = [201 108 64 22];

            % Create Label32
            app.Label32 = uilabel(app.LowerBoundPanel);
            app.Label32.HorizontalAlignment = 'center';
            app.Label32.Position = [222 53 21 15];
            app.Label32.Text = 'Gyz';

            % Create etAngleLaminaGyz
            app.etAngleLaminaGyz = uieditfield(app.LowerBoundPanel, 'numeric');
            app.etAngleLaminaGyz.Editable = 'off';
            app.etAngleLaminaGyz.HorizontalAlignment = 'center';
            app.etAngleLaminaGyz.Position = [201 21 64 22];

            % Create UpperBoundPanel
            app.UpperBoundPanel = uipanel(app.UIFigure);
            app.UpperBoundPanel.Title = 'Upper Bound';
            app.UpperBoundPanel.Position = [26 33 297 325];
            % Create tbLamina
            app.tbLamina = uieditfield(app.UpperBoundPanel, 'text');
            app.tbLamina.Editable = 'off';
            app.tbLamina.HorizontalAlignment = 'center';
            app.tbLamina.BackgroundColor = [0.9373 0.9373 0.9373];
            app.tbLamina.Position = [11 260 280 35];

            % Create Label15
            app.Label15 = uilabel(app.UpperBoundPanel);
            app.Label15.HorizontalAlignment = 'center';
            app.Label15.VerticalAlignment = 'center';
            app.Label15.Position = [48 224 20 15];
            app.Label15.Text = 'E1';

            % Create etLaminaE1
            app.etLaminaE1 = uieditfield(app.UpperBoundPanel, 'numeric');
            app.etLaminaE1.Editable = 'off';
            app.etLaminaE1.HorizontalAlignment = 'center';
            app.etLaminaE1.Position = [26 192 64 22];

            % Create Label16
            app.Label16 = uilabel(app.UpperBoundPanel);
            app.Label16.HorizontalAlignment = 'center';
            app.Label16.Position = [48 141 20 15];
            app.Label16.Text = 'v12';

            % Create etLaminav12
            app.etLaminav12 = uieditfield(app.UpperBoundPanel, 'numeric');
            app.etLaminav12.Editable = 'off';
            app.etLaminav12.HorizontalAlignment = 'center';
            app.etLaminav12.Position = [26 109 64 22];

            % Create Label17
            app.Label17 = uilabel(app.UpperBoundPanel);
            app.Label17.HorizontalAlignment = 'center';
            app.Label17.Position = [46 54 23 15];
            app.Label17.Text = 'G12';

            % Create etLaminaG12
            app.etLaminaG12 = uieditfield(app.UpperBoundPanel, 'numeric');
            app.etLaminaG12.Editable = 'off';
            app.etLaminaG12.HorizontalAlignment = 'center';
            app.etLaminaG12.Position = [26 22 64 22];

            % Create Label18
            app.Label18 = uilabel(app.UpperBoundPanel);
            app.Label18.HorizontalAlignment = 'center';
            app.Label18.VerticalAlignment = 'center';
            app.Label18.Position = [139 224 20 15];
            app.Label18.Text = 'E2';

            % Create etLaminaE2
            app.etLaminaE2 = uieditfield(app.UpperBoundPanel, 'numeric');
            app.etLaminaE2.Editable = 'off';
            app.etLaminaE2.HorizontalAlignment = 'center';
            app.etLaminaE2.Position = [117 192 64 22];

            % Create Label19
            app.Label19 = uilabel(app.UpperBoundPanel);
            app.Label19.HorizontalAlignment = 'center';
            app.Label19.Position = [139 141 20 15];
            app.Label19.Text = 'v13';

            % Create etLaminav13
            app.etLaminav13 = uieditfield(app.UpperBoundPanel, 'numeric');
            app.etLaminav13.Editable = 'off';
            app.etLaminav13.HorizontalAlignment = 'center';
            app.etLaminav13.Position = [117 109 64 22];
            % Create Label20
            app.Label20 = uilabel(app.UpperBoundPanel);
            app.Label20.HorizontalAlignment = 'center';
            app.Label20.Position = [137 54 23 15];
            app.Label20.Text = 'G13';

            % Create etLaminaG13
            app.etLaminaG13 = uieditfield(app.UpperBoundPanel, 'numeric');
            app.etLaminaG13.Editable = 'off';
            app.etLaminaG13.HorizontalAlignment = 'center';
            app.etLaminaG13.Position = [117 22 64 22];

            % Create Label21
            app.Label21 = uilabel(app.UpperBoundPanel);
            app.Label21.HorizontalAlignment = 'center';
            app.Label21.VerticalAlignment = 'center';
            app.Label21.Position = [230 223 20 15];
            app.Label21.Text = 'E3';

            % Create etLaminaE3
            app.etLaminaE3 = uieditfield(app.UpperBoundPanel, 'numeric');
            app.etLaminaE3.Editable = 'off';
            app.etLaminaE3.HorizontalAlignment = 'center';
            app.etLaminaE3.Position = [208 191 64 22];

            % Create Label22
            app.Label22 = uilabel(app.UpperBoundPanel);
            app.Label22.HorizontalAlignment = 'center';
            app.Label22.Position = [230 140 20 15];
            app.Label22.Text = 'v23';

            % Create etLaminav23
            app.etLaminav23 = uieditfield(app.UpperBoundPanel, 'numeric');
            app.etLaminav23.Editable = 'off';
            app.etLaminav23.HorizontalAlignment = 'center';
            app.etLaminav23.Position = [208 108 64 22];

            % Create Label23
            app.Label23 = uilabel(app.UpperBoundPanel);
            app.Label23.HorizontalAlignment = 'center';
            app.Label23.Position = [228 53 23 15];
            app.Label23.Text = 'G23';

            % Create etLaminaG23
            app.etLaminaG23 = uieditfield(app.UpperBoundPanel, 'numeric');
            app.etLaminaG23.Editable = 'off';
            app.etLaminaG23.HorizontalAlignment = 'center';
            app.etLaminaG23.Position = [208 21 64 22];

            % Create YarnPropertiesPanel
            app.YarnPropertiesPanel = uipanel(app.UIFigure);
            app.YarnPropertiesPanel.Title = 'Yarn Properties';
            app.YarnPropertiesPanel.Position = [978 459 403 325];

            % Create Label6
            app.Label6 = uilabel(app.YarnPropertiesPanel);
            app.Label6.HorizontalAlignment = 'center';
            app.Label6.VerticalAlignment = 'center';
            app.Label6.Position = [66 218 20 15];
            app.Label6.Text = 'E1';

            % Create etYarnE1
            app.etYarnE1 = uieditfield(app.YarnPropertiesPanel, 'numeric');
            app.etYarnE1.Editable = 'off';
            app.etYarnE1.HorizontalAlignment = 'center';
            app.etYarnE1.Position = [26 186 100 22];
            % Create Label7
            app.Label7 = uilabel(app.YarnPropertiesPanel);
            app.Label7.HorizontalAlignment = 'center';
            app.Label7.Position = [66 135 20 15];
            app.Label7.Text = 'v12';

            % Create etYarnv12
            app.etYarnv12 = uieditfield(app.YarnPropertiesPanel, 'numeric');
            app.etYarnv12.Editable = 'off';
            app.etYarnv12.HorizontalAlignment = 'center';
            app.etYarnv12.Position = [26 103 100 22];

            % Create Label8
            app.Label8 = uilabel(app.YarnPropertiesPanel);
            app.Label8.HorizontalAlignment = 'center';
            app.Label8.Position = [64 48 23 15];
            app.Label8.Text = 'G12';

            % Create etYarnG12
            app.etYarnG12 = uieditfield(app.YarnPropertiesPanel, 'numeric');
            app.etYarnG12.Editable = 'off';
            app.etYarnG12.HorizontalAlignment = 'center';
            app.etYarnG12.Position = [26 16 100 22];

            % Create Label9
            app.Label9 = uilabel(app.YarnPropertiesPanel);
            app.Label9.HorizontalAlignment = 'center';
            app.Label9.VerticalAlignment = 'center';
            app.Label9.Position = [192 218 20 15];
            app.Label9.Text = 'E2';

            % Create etYarnE2
            app.etYarnE2 = uieditfield(app.YarnPropertiesPanel, 'numeric');
            app.etYarnE2.Editable = 'off';
            app.etYarnE2.HorizontalAlignment = 'center';
            app.etYarnE2.Position = [152 186 100 22];

            % Create Label10
            app.Label10 = uilabel(app.YarnPropertiesPanel);
            app.Label10.HorizontalAlignment = 'center';
            app.Label10.Position = [192 135 20 15];
            app.Label10.Text = 'v13';

            % Create etYarnv13
            app.etYarnv13 = uieditfield(app.YarnPropertiesPanel, 'numeric');
            app.etYarnv13.Editable = 'off';
            app.etYarnv13.HorizontalAlignment = 'center';
            app.etYarnv13.Position = [152 103 100 22];

            % Create Label11
            app.Label11 = uilabel(app.YarnPropertiesPanel);
            app.Label11.HorizontalAlignment = 'center';
            app.Label11.Position = [190 48 23 15];
            app.Label11.Text = 'G13';

            % Create etYarnG13
            app.etYarnG13 = uieditfield(app.YarnPropertiesPanel, 'numeric');
            app.etYarnG13.Editable = 'off';
            app.etYarnG13.HorizontalAlignment = 'center';
            app.etYarnG13.Position = [152 16 100 22];

            % Create Label12
            app.Label12 = uilabel(app.YarnPropertiesPanel);
            app.Label12.HorizontalAlignment = 'center';
            app.Label12.VerticalAlignment = 'center';
            app.Label12.Position = [318 218 20 15];
            app.Label12.Text = 'E3';
```

```
% Create etYarnE3
app.etYarnE3 = uieditfield(app.YarnPropertiesPanel, 'numeric');
app.etYarnE3.Editable = 'off';
app.etYarnE3.HorizontalAlignment = 'center';
app.etYarnE3.Position = [278 186 100 22];

% Create Label13
app.Label13 = uilabel(app.YarnPropertiesPanel);
app.Label13.HorizontalAlignment = 'center';
app.Label13.Position = [318 135 20 15];
app.Label13.Text = 'v23';

% Create etYarnv23
app.etYarnv23 = uieditfield(app.YarnPropertiesPanel, 'numeric');
app.etYarnv23.Editable = 'off';
app.etYarnv23.HorizontalAlignment = 'center';
app.etYarnv23.Position = [278 103 100 22];

% Create Label14
app.Label14 = uilabel(app.YarnPropertiesPanel);
app.Label14.HorizontalAlignment = 'center';
app.Label14.Position = [316 48 23 15];
app.Label14.Text = 'G23';

% Create etYarnG23
app.etYarnG23 = uieditfield(app.YarnPropertiesPanel, 'numeric');
app.etYarnG23.Editable = 'off';
app.etYarnG23.HorizontalAlignment = 'center';
app.etYarnG23.Position = [278 16 100 22];

% Create YarnPackingFractionLabel
app.YarnPackingFractionLabel = uilabel(app.YarnPropertiesPanel);
app.YarnPackingFractionLabel.HorizontalAlignment = 'right';
app.YarnPackingFractionLabel.Position = [82 275 125 15];
app.YarnPackingFractionLabel.Text = 'Yarn Packing Fraction';

% Create etYarnPackingFraction
app.etYarnPackingFraction = ...
    uieditfield(app.YarnPropertiesPanel, 'numeric');
app.etYarnPackingFraction.ValueChangedFcn =...
    createCallbackFcn(app, @PF, true);
app.etYarnPackingFraction.Limits = [0.05 1];
app.etYarnPackingFraction.ValueDisplayFormat = '%.3f';
app.etYarnPackingFraction.HorizontalAlignment = 'center';
app.etYarnPackingFraction.Position = [222 271 100 22];
app.etYarnPackingFraction.Value = 0.75;

% Create YarnRadiusLabel
app.YarnRadiusLabel = uilabel(app.YarnPropertiesPanel);
app.YarnRadiusLabel.HorizontalAlignment = 'right';
app.YarnRadiusLabel.Position = [134 242 73 15];
app.YarnRadiusLabel.Text = 'Yarn Radius';

% Create etYarnRadius
app.etYarnRadius = uieditfield(app.YarnPropertiesPanel, 'numeric');
app.etYarnRadius.Limits = [1e-05 1001];
app.etYarnRadius.ValueDisplayFormat = '%.3f';
app.etYarnRadius.Editable = 'off';
app.etYarnRadius.HorizontalAlignment = 'center';
app.etYarnRadius.Position = [222 238 100 22];
app.etYarnRadius.Value = 0.2;

% Create BraidProperties
app.BraidProperties = uipanel(app.UIFigure);
app.BraidProperties.Title = 'Braid Properties';
app.BraidProperties.Position = [1085 33 296 325];

% Create BraidCrossSectionalDimsEditFieldLabel
```

```
            app.BraidCrossSectionalDimsEditFieldLabel = ...
                uilabel(app.BraidProperties);
            app.BraidCrossSectionalDimsEditFieldLabel.HorizontalAlignment = 'right';
            app.BraidCrossSectionalDimsEditFieldLabel.Position = [6 243 158 15];
            app.BraidCrossSectionalDimsEditFieldLabel.Text = ...
                'Braid Cross-Sectional Dims.';

            % Create BraidCrossSectionalDimsEditField
            app.BraidCrossSectionalDimsEditField = ...
                uieditfield(app.BraidProperties, 'text');
            app.BraidCrossSectionalDimsEditField.Editable = 'off';
            app.BraidCrossSectionalDimsEditField.HorizontalAlignment = 'center';
            app.BraidCrossSectionalDimsEditField.Position = [179 239 100 22];

            % Create UnitCellHeigthEditFieldLabel
            app.UnitCellHeigthEditFieldLabel = uilabel(app.BraidProperties);
            app.UnitCellHeigthEditFieldLabel.HorizontalAlignment = 'right';
            app.UnitCellHeigthEditFieldLabel.Position = [73 210 91 15];
            app.UnitCellHeigthEditFieldLabel.Text = 'Unit Cell Heigth';

            % Create UnitCellHeigthEditField
            app.UnitCellHeigthEditField = uieditfield(app.BraidProperties, 'text');
            app.UnitCellHeigthEditField.Editable = 'off';
            app.UnitCellHeigthEditField.HorizontalAlignment = 'center';
            app.UnitCellHeigthEditField.Position = [179 206 100 22];

            % Create AverageBraidingAngleEditFieldLabel
            app.AverageBraidingAngleEditFieldLabel = uilabel(app.BraidProperties);
            app.AverageBraidingAngleEditFieldLabel.HorizontalAlignment = 'right';
            app.AverageBraidingAngleEditFieldLabel.Position = [29 177 135 15];
            app.AverageBraidingAngleEditFieldLabel.Text = 'Average Braiding Angle';

            % Create AverageBraidingAngleEditField
            app.AverageBraidingAngleEditField = ...
                uieditfield(app.BraidProperties, 'text');
            app.AverageBraidingAngleEditField.Editable = 'off';
            app.AverageBraidingAngleEditField.HorizontalAlignment = 'center';
            app.AverageBraidingAngleEditField.Position = [179 173 100 22];

            % Create BraidingTightnessEditFieldLabel
            app.BraidingTightnessEditFieldLabel = uilabel(app.BraidProperties);
            app.BraidingTightnessEditFieldLabel.HorizontalAlignment = 'right';
            app.BraidingTightnessEditFieldLabel.Position = [57 78 107 15];
            app.BraidingTightnessEditFieldLabel.Text = 'Braiding Tightness';

            % Create BraidingTightnessEditField
            app.BraidingTightnessEditField = ...
                uieditfield(app.BraidProperties, 'text');
            app.BraidingTightnessEditField.Editable = 'off';
            app.BraidingTightnessEditField.HorizontalAlignment = 'center';
            app.BraidingTightnessEditField.Position = [179 74 100 22];

            % Create FiberVolumeFractionEditFieldLabel
            app.FiberVolumeFractionEditFieldLabel = uilabel(app.BraidProperties);
            app.FiberVolumeFractionEditFieldLabel.HorizontalAlignment = 'right';
            app.FiberVolumeFractionEditFieldLabel.Position = [39 45 125 15];
            app.FiberVolumeFractionEditFieldLabel.Text = 'Fiber Volume Fraction';

            % Create FiberVolumeFractionEditField
            app.FiberVolumeFractionEditField = ...
                uieditfield(app.BraidProperties, 'text');
            app.FiberVolumeFractionEditField.Editable = 'off';
            app.FiberVolumeFractionEditField.HorizontalAlignment = 'center';
            app.FiberVolumeFractionEditField.Position = [179 41 100 22];

            % Create NumberofYarnsEditFieldLabel
            app.NumberofYarnsEditFieldLabel = uilabel(app.BraidProperties);
            app.NumberofYarnsEditFieldLabel.HorizontalAlignment = 'right';
```

```
app.NumberofYarnsEditFieldLabel.Position = [67 12 97 15];
app.NumberofYarnsEditFieldLabel.Text = 'Number of Yarns';

% Create NumberofYarnsEditField
app.NumberofYarnsEditField = uieditfield(app.BraidProperties, 'text');
app.NumberofYarnsEditField.Editable = 'off';
app.NumberofYarnsEditField.HorizontalAlignment = 'center';
app.NumberofYarnsEditField.Position = [179 8 100 22];

% Create YoungsModulusRangeEditFieldLabel
app.YoungsModulusRangeEditFieldLabel = uilabel(app.BraidProperties);
app.YoungsModulusRangeEditFieldLabel.HorizontalAlignment = 'right';
app.YoungsModulusRangeEditFieldLabel.Position = [26 277 138 15];
app.YoungsModulusRangeEditFieldLabel.Text = 'Young''s Modulus Range';

% Create YoungsModulusRangeEditField
app.YoungsModulusRangeEditField = uieditfield(app.BraidProperties, 'text');
app.YoungsModulusRangeEditField.Editable = 'off';
app.YoungsModulusRangeEditField.HorizontalAlignment = 'center';
app.YoungsModulusRangeEditField.Position = [179 273 100 22];

% Create InteriorBraidingAngleEditFieldLabel
app.InteriorBraidingAngleEditFieldLabel = uilabel(app.BraidProperties);
app.InteriorBraidingAngleEditFieldLabel.HorizontalAlignment = 'right';
app.InteriorBraidingAngleEditFieldLabel.Position = [37 144 127 15];
app.InteriorBraidingAngleEditFieldLabel.Text = 'Interior Braiding Angle';

% Create InteriorBraidingAngleEditField
app.InteriorBraidingAngleEditField = ...
    uieditfield(app.BraidProperties, 'text');
app.InteriorBraidingAngleEditField.Editable = 'off';
app.InteriorBraidingAngleEditField.HorizontalAlignment = 'center';
app.InteriorBraidingAngleEditField.Position = [179 140 100 22];

% Create SurfaceBraidingAngleEditFieldLabel
app.SurfaceBraidingAngleEditFieldLabel = uilabel(app.BraidProperties);
app.SurfaceBraidingAngleEditFieldLabel.HorizontalAlignment = 'right';
app.SurfaceBraidingAngleEditFieldLabel.Position = [32 111 131 15];
app.SurfaceBraidingAngleEditFieldLabel.Text = 'Surface Braiding Angle';

% Create SurfaceBraidingAngleEditField
app.SurfaceBraidingAngleEditField = ...
    uieditfield(app.BraidProperties, 'text');
app.SurfaceBraidingAngleEditField.Editable = 'off';
app.SurfaceBraidingAngleEditField.HorizontalAlignment = 'center';
app.SurfaceBraidingAngleEditField.Position = [178 107 100 22];

% Create FGMPanel
app.FGMPanel = uipanel(app.UIFigure);
app.FGMPanel.Title = 'FGM';
app.FGMPanel.Position = [707 33 336 325];

% Create Label24_2
app.Label24_2 = uilabel(app.FGMPanel);
app.Label24_2.HorizontalAlignment = 'center';
app.Label24_2.VerticalAlignment = 'center';
app.Label24_2.Position = [69 229 20 15];
app.Label24_2.Text = 'Ex';

% Create etFGMEx
app.etFGMEx = uieditfield(app.FGMPanel, 'numeric');
app.etFGMEx.Editable = 'off';
app.etFGMEx.HorizontalAlignment = 'center';
app.etFGMEx.Position = [47 197 64 22];

% Create Label25_2
app.Label25_2 = uilabel(app.FGMPanel);
app.Label25_2.HorizontalAlignment = 'center';
```

```
            app.Label25_2.Position = [69 146 20 15];
            app.Label25_2.Text = 'vxy';

            % Create etFGMvxy
            app.etFGMvxy = uieditfield(app.FGMPanel, 'numeric');
            app.etFGMvxy.Editable = 'off';
            app.etFGMvxy.HorizontalAlignment = 'center';
            app.etFGMvxy.Position = [47 114 64 22];

            % Create Label26_2
            app.Label26_2 = uilabel(app.FGMPanel);
            app.Label26_2.HorizontalAlignment = 'center';
            app.Label26_2.Position = [68 59 21 15];
            app.Label26_2.Text = 'Gxy';

            % Create etFGMGxy
            app.etFGMGxy = uieditfield(app.FGMPanel, 'numeric');
            app.etFGMGxy.Editable = 'off';
            app.etFGMGxy.HorizontalAlignment = 'center';
            app.etFGMGxy.Position = [47 27 64 22];

            % Create Label27_2
            app.Label27_2 = uilabel(app.FGMPanel);
            app.Label27_2.HorizontalAlignment = 'center';
            app.Label27_2.VerticalAlignment = 'center';
            app.Label27_2.Position = [157 228 20 15];
            app.Label27_2.Text = 'Ey';

            % Create etFGMEy
            app.etFGMEy = uieditfield(app.FGMPanel, 'numeric');
            app.etFGMEy.Editable = 'off';
            app.etFGMEy.HorizontalAlignment = 'center';
            app.etFGMEy.Position = [135 196 64 22];

            % Create Label28_2
            app.Label28_2 = uilabel(app.FGMPanel);
            app.Label28_2.HorizontalAlignment = 'center';
            app.Label28_2.Position = [157 145 20 15];
            app.Label28_2.Text = 'vxz';

            % Create etFGMvxz
            app.etFGMvxz = uieditfield(app.FGMPanel, 'numeric');
            app.etFGMvxz.Editable = 'off';
            app.etFGMvxz.HorizontalAlignment = 'center';
            app.etFGMvxz.Position = [135 113 64 22];

            % Create Label29_2
            app.Label29_2 = uilabel(app.FGMPanel);
            app.Label29_2.HorizontalAlignment = 'center';
            app.Label29_2.Position = [156 58 21 15];
            app.Label29_2.Text = 'Gxz';

            % Create etFGMGxz
            app.etFGMGxz = uieditfield(app.FGMPanel, 'numeric');
            app.etFGMGxz.Editable = 'off';
            app.etFGMGxz.HorizontalAlignment = 'center';
            app.etFGMGxz.Position = [135 26 64 22];

            % Create Label30_2
            app.Label30_2 = uilabel(app.FGMPanel);
            app.Label30_2.HorizontalAlignment = 'center';
            app.Label30_2.VerticalAlignment = 'center';
            app.Label30_2.Position = [245 228 20 15];
            app.Label30_2.Text = 'Ez';

            % Create etFGMEz
            app.etFGMEz = uieditfield(app.FGMPanel, 'numeric');
            app.etFGMEz.Editable = 'off';
```

```
app.etFGMEz.HorizontalAlignment = 'center';
app.etFGMEz.Position = [223 196 64 22];

% Create Label31_2
app.Label31_2 = uilabel(app.FGMPanel);
app.Label31_2.HorizontalAlignment = 'center';
app.Label31_2.Position = [245 145 20 15];
app.Label31_2.Text = 'vyz';

% Create etFGMvyz
app.etFGMvyz = uieditfield(app.FGMPanel, 'numeric');
app.etFGMvyz.Editable = 'off';
app.etFGMvyz.HorizontalAlignment = 'center';
app.etFGMvyz.Position = [223 113 64 22];

% Create Label32_2
app.Label32_2 = uilabel(app.FGMPanel);
app.Label32_2.HorizontalAlignment = 'center';
app.Label32_2.Position = [244 58 21 15];
app.Label32_2.Text = 'Gyz';

% Create etFGMGyz
app.etFGMGyz = uieditfield(app.FGMPanel, 'numeric');
app.etFGMGyz.Editable = 'off';
app.etFGMGyz.HorizontalAlignment = 'center';
app.etFGMGyz.Position = [223 26 64 22];

% Create BraidAngleAverageLabel
app.BraidAngleAverageLabel = uilabel(app.UIFigure);
app.BraidAngleAverageLabel.HorizontalAlignment = 'right';
app.BraidAngleAverageLabel.Position = [29 491 126 15];
app.BraidAngleAverageLabel.Text = 'Braid Angle (Average)';

% Create etTAngle
app.etTAngle = uieditfield(app.UIFigure, 'numeric');
app.etTAngle.ValueChangedFcn = ...
    createCallbackFcn(app, @TAngleChange, true);
app.etTAngle.Limits = [1e-15 90];
app.etTAngle.Editable = 'off';
app.etTAngle.HorizontalAlignment = 'center';
app.etTAngle.Position = [170 487 100 22];
app.etTAngle.Value = 2;

% Create AnglesfromGeometryPanel
app.AnglesfromGeometryPanel = uipanel(app.UIFigure);
app.AnglesfromGeometryPanel.TitlePosition = 'centertop';
app.AnglesfromGeometryPanel.Title = 'Angles from Geometry';
app.AnglesfromGeometryPanel.Position = [312 476 279 138];

% Create ApproxAverageAngleLabel
app.ApproxAverageAngleLabel = uilabel(app.AnglesfromGeometryPanel);
app.ApproxAverageAngleLabel.HorizontalAlignment = 'right';
app.ApproxAverageAngleLabel.Position = [11 89 131 15];
app.ApproxAverageAngleLabel.Text = 'Approx. Average Angle';

% Create etAverageAngle
app.etAverageAngle = ...
    uieditfield(app.AnglesfromGeometryPanel, 'numeric');
app.etAverageAngle.Editable = 'off';
app.etAverageAngle.HorizontalAlignment = 'center';
app.etAverageAngle.Position = [157 85 100 22];

% Create ApproxInteriorAngleLabel
app.ApproxInteriorAngleLabel = uilabel(app.AnglesfromGeometryPanel);
app.ApproxInteriorAngleLabel.HorizontalAlignment = 'right';
app.ApproxInteriorAngleLabel.Position = [19 53 123 15];
app.ApproxInteriorAngleLabel.Text = 'Approx. Interior Angle';
```

```
            % Create etInteriorAngle
            app.etInteriorAngle = ...
                uieditfield(app.AnglesfromGeometryPanel, ’numeric’);
            app.etInteriorAngle.Editable = ’off’;
            app.etInteriorAngle.HorizontalAlignment = ’center’;
            app.etInteriorAngle.Position = [157 49 100 22];

            % Create ApproxSurfaceAngleLabel
            app.ApproxSurfaceAngleLabel = uilabel(app.AnglesfromGeometryPanel);
            app.ApproxSurfaceAngleLabel.HorizontalAlignment = ’right’;
            app.ApproxSurfaceAngleLabel.Position = [15 15 127 15];
            app.ApproxSurfaceAngleLabel.Text = ’Approx. Surface Angle’;

            % Create etSurfaceAngle
            app.etSurfaceAngle = ...
                uieditfield(app.AnglesfromGeometryPanel, ’numeric’);
            app.etSurfaceAngle.Editable = ’off’;
            app.etSurfaceAngle.HorizontalAlignment = ’center’;
            app.etSurfaceAngle.Position = [157 11 100 22];

            % Create ButtonGroup
            app.ButtonGroup = uibuttongroup(app.UIFigure);
            app.ButtonGroup.SelectionChangedFcn = ...
                createCallbackFcn(app, @modeChange, true);
            app.ButtonGroup.BorderType = ’none’;
            app.ButtonGroup.Position = [9 486 21 72];

            % Create enablePitch
            app.enablePitch = uiradiobutton(app.ButtonGroup);
            app.enablePitch.Text = ’’;
            app.enablePitch.Position = [3 48 25 15];
            app.enablePitch.Value = true;

            % Create enableAngle
            app.enableAngle = uiradiobutton(app.ButtonGroup);
            app.enableAngle.Text = ’’;
            app.enableAngle.Position = [3 6 25 15];
        end
    end

    methods (Access = public)

        % Construct app
        function app = braidGenerator

            % Create and configure components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

Listing C.3: Main function for generating the braid paths.

```
%+------------------------------------------------------------------------+
%|                                                                        |
%| FILENAME : generateBraidPaths_m                        VERSION : b.1.0 |
%|                                                                        |
%| TITLE : Generate Braid Paths                   AUTHOR : Daniel Aldrich |
%|                                                                        |
%+------------------------------------------------------------------------+
%|                                                                        |
%| DEPENDENT FILES :                                                      |
%|       Machine_Emulation.m                                              |
%|       Plot_Path.m                                                      |
%|                                                                        |
%| DESCRIPTION :                                                          |
%|       <none>                                                           |
%|                                                                        |
%| PUBLIC FUNCTIONS :                                                     |
%|       <none>                                                           |
%|                                                                        |
%| NOTES :                                                                |
%|       <none>                                                           |
%|                                                                        |
%| COPYRIGHT :                                                            |
%|       Copyright (c) 2017 Daniel Aldrich                                |
%{
%|       Permission is hereby granted, free of charge, to any person      |
%|       obtaining a copy of this software and associated documentation  |
%|       files (the "Software"), to deal in the Software without          |
%|       restriction, including without limitation the rights to use,    |
%|       copy, modify, merge, publish, distribute, sublicense, and/or    |
%|       sell copies of the Software, and to permit persons to whom the   |
%|       Software is furnished to do so, subject to the following         |
%|       conditions:                                                      |
%|                                                                        |
%|       The above copyright notice and this permission notice shall be   |
%|       included in all copies or substantial portions of the Software.  |
%|                                                                        |
%|       THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  |
%|       EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES  |
%|       OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND         |
%|       NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT      |
%|       HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,     |
%|       WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING     |
%|       FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR    |
%|       OTHER DEALINGS IN THE SOFTWARE.                                  |
%}
%|                                                                        |
%| CHANGES :                                                              |
%|       <none>                                                           |
%|                                                                        |
%+------------------------------------------------------------------------+

function [time, Vf, yarnAngle,FGM] = generateBraidPaths(Knot,Path,...
    braidLength,braidWidth,Radius,Spacing_Factor,unitCells,folderName,...
    enableSmoothing,enableYarns,enableTimer,enablePlots,enableTypeII,z)

system(['mkdir ',folderName]);
res = 15;
Spacing_Factor = Spacing_Factor+enableYarns*(1/sind(45+Path)-1.0+Path/180);
zScale = z/2;
braidingSteps = unitCells*2;
smoothingFactor = floor(Knot/3)*2+1;
repeats = 1;
replicates = 1;
if enableTimer
    repeats = 3;
    replicates = 100;
```

```
end
for m = 1:repeats
    tic
    for c = 1:replicates
        clear x y z M
        Path = pi()*Path/180;
        Pattern=Machine_Emulation(braidLength,braidWidth,braidingSteps+1);
        Pattern = Pattern(1:braidingSteps+1);
        Size = max(max(Pattern{1}));
        Deadzone = 2;
        number_n = 24;

        Middlex = (braidWidth*2+2)/2;
        Middley = (braidLength*2+2)/2;

        for n = 1:Size
            if enableTypeII
                [x(n,:),y(n,:),z(n,:)] =...
                    Plot_Path_t(Pattern(:),n,1,Knot,Path);
            else
                [x(n,:),y(n,:),z(n,:)] =...
                    Plot_Path(Pattern(:),n,1,Knot,Path);
                [xFGM(n,:),yFGM(n,:),zFGM(n,:)] = ...
                    Plot_Path(Pattern(1:3),n,1,1,0);
            end
        end


        Z_n = z;
        X_n = x-Middlex;
        Y_n = y-Middley;

        xFGM = diff((xFGM-Middlex)*Radius*2*Spacing_Factor,1,2);
        yFGM = diff((yFGM-Middley)*Radius*2*Spacing_Factor,1,2);
        mFGM = sqrt(xFGM.^2 + yFGM.^2);
        zFGM = diff(zFGM*zScale,1,2);

        logiFGM = asind(yFGM./mFGM)>=0;

        bFGM = logiFGM.*(acosd(xFGM./mFGM)) +...
            ~logiFGM.*(360 - acosd(xFGM./mFGM));
        bFGM(isnan(bFGM)) = 0;
        tFGM = atand(abs(mFGM./zFGM));

        FGM = [bFGM(:), tFGM(:)];

        for n = 1:Size
            x = X_n(n,:);
            y = Y_n(n,:);
            z = Z_n(n,:);

            x = [x(1)*[ones(1,20)],x(:)',x(end)*[ones(1,20)]]...
                *Radius*2*Spacing_Factor;
            y = [y(1)*[ones(1,20)],y(:)',y(end)*[ones(1,20)]]...
                *Radius*2*Spacing_Factor;
            if enableSmoothing
                x = filtfilt(ones(1,smoothingFactor)/smoothingFactor,1,x);
                y = filtfilt(ones(1,smoothingFactor)/smoothingFactor,1,y);
            end
            z = [Deadzone/20*[20:-1:1],...
                z(:)'*zScale,...
                -Deadzone/20*[1:20]+z(end)*zScale]...
                -Deadzone;

            if enablePlots
                if n == 1
                    fig1 = figure('Name','Braid Paths',...
                        'units','normalized',...
                        'position',[0,0.55,0.5,0.35]);
                    hold on;
                    fig2 = figure('Name','Braid Geometry Approximation',...
                        'units','normalized',...
                        'position',[0.5,0.55,0.5,0.35]);
                    view([1,1,1])
                    hold on;
                end
                figure(fig1)
                if sum(n == number_n)
                    plot3(x,y,z,'-o','Color','r','LineWidth',2)
                else
                    plot3(x,y,z,':','Color',[0.5 0.5 0.5],'LineWidth',1.8)
                end
                axis equal
                grid on
                figure(fig2)
                tubeplot([x(:),y(:),z(:)]',Radius,res);
                axis equal
                grid on
            end

            M(:,:,n) = [x(:)*100,z(:)*100,y(:)*100];
            filename = sprintf('\\Yarn%i.txt',n);
            csvwrite([folderName,filename],M(:,:,n))
        end

        if enableYarns
            for a = 1:braidWidth
                for b = 1:braidLength
                    n = n + 1;
                    x = [(2*a-Middlex)*ones(1,length(z))]...
                        *Radius*2*Spacing_Factor;
                    y = [(2*b-Middley)*ones(1,length(z))]...
                        *Radius*2*Spacing_Factor;
                    if enablePlots
                        figure(fig1)
                        plot3(x,y,z,':',...
                            'Color',[0.5 0.5 0.5],...
                            'LineWidth',1.8)
                        axis equal
                        grid on
                        figure(fig2)
                        tubeplot([x(:),y(:),z(:)]',Radius,res);
                        axis equal
                        grid on
                        view([1,1,1])
                    end
                    M(:,:,n) = [x(:)*100,z(:)*100,y(:)*100];
                    filename = sprintf('\\Yarn%i.txt',n);
                    csvwrite([folderName,filename],M(:,:,n))
                end
            end
            Size = Size+braidWidth*braidLength;
        end
        csvwrite([folderName,'\Data.txt'],...
            [Size, Radius/10, max(abs(z))/10,...
            braidWidth, braidLength, Spacing_Factor]);
        %axis equal
    end
    time(m) = toc/100;
end


M = M/100;
a(size(M,1)-1,size(M,3)) = 0;
dM = diff(M);
yarnArea(size(M,1)-1) = 0;
```

```
yarnAngle(size(M,1)-1) = 0;
yAngle(size(M,1)-1,size(M,3)) = 0;
yArea(size(M,1)-1,size(M,3)) = 0;
Vf(size(M,1)-1) = 0;
Area = ((4 * Spacing_Factor * braidWidth + 3) * Radius)...
    *((4 * Spacing_Factor * braidLength + 3) * Radius);
c = 0;
for m = 1:size(M,1)-1
    for n = 1:size(M,3)
        c = c + 1;
        dx = dM(m,1,n);
        dz = dM(m,2,n);
        dy = dM(m,3,n);
        a(m,n) = Radius/(abs(dz)/norm([dx,dy,dz]));
        yAngle(m,n) = atand(norm([dx,dy])/abs(dz));
        yArea(m,n) = pi()*Radius*a(m,n);
        yarnAngle(m) = yarnAngle(m) + atand(norm([dx,dy])/abs(dz));
        yarnArea(m) = yarnArea(m) + pi()*Radius*a(m,n);
    end
    yarnAngle(m) = yarnAngle(m)/n;
    Vf(m) = yarnArea(m)/Area;
end
if enablePlots
    figure('Name','Volume Fraction along Braid Length',...
        'units','normalized',...
        'position',[0,0.05,1/3,0.35])
    plot(-M(1:end-1,2,1),Vf)
end
ind = -M(1:end-1,2,1)>2 & -M(1:end-1,2,1)<-M(end,2,1)-2;
Vf = (mean(Vf(ind)));
if enablePlots
    figure('Name','Average Yarn Angle along Braid Length',...
        'units','normalized',...
        'position',[1/3,0.05,1/3,0.35])
    plot(-M(1:end-1,2,1),yarnAngle)
end
yarnAngle = (mean(yarnAngle(ind)));
if enablePlots
    figure('Name','Yarn Angle along Braid Length',...
        'units','normalized',...
        'position',[2/3,0.05,1/3,0.35])
    for n = 1:size(M,3)
        plot3(-M(1:end-1,2,n),ones(1,size(M,1)-1)*n,yAngle(:,n))
        hold on
    end
end
if enablePlots
                if n == 1
                    fig1 = figure('Name','Braid Paths',...
                        'units','normalized',...
                        'position',[0,0.55,0.5,0.35]);
                    hold on;
                    fig2 = figure('Name','Braid Geometry Approximation',...
                        'units','normalized',...
                        'position',[0.5,0.55,0.5,0.35]);
                    view([1,1,1])
                    hold on;
                end
                figure(fig1)
                if sum(n == number_n)
                    plot3(x,y,z,'-o','Color','r','LineWidth',2)
                else
                    plot3(x,y,z,':','Color',[0.5 0.5 0.5],'LineWidth',1.8)
                end
                axis equal
                grid on
                figure(fig2)
                tubeplot([x(:),y(:),z(:)]',Radius,res);
                axis equal
                grid on
            end
```

Listing C.4: Function that performs machine emulation to generate the braid pattern.

```
%+-----------------------------------------------------------------------+
%|                                                                       |
%| FILENAME : Machine_Emulation_m                        VERSION : 1.0.0 |
%|                                                                       |
%| TITLE : Machine Emulation                       AUTHOR : Daniel Aldrich |
%|                                                                       |
%+-----------------------------------------------------------------------+
%|                                                                       |
%| DEPENDENT FILES :                                                     |
%|       <none>                                                          |
%|                                                                       |
%| DESCRIPTION :                                                         |
%|       <none>                                                          |
%|                                                                       |
%| PUBLIC FUNCTIONS :                                                    |
%|       <none>                                                          |
%|                                                                       |
%| NOTES :                                                               |
%|       <none>                                                          |
%|                                                                       |
%| COPYRIGHT :                                                           |
%|       Copyright (c) 2017 Daniel Aldrich                               |
%{
%|       Permission is hereby granted, free of charge, to any person     |
%|       obtaining a copy of this software and associated documentation |
%|       files (the "Software"), to deal in the Software without         |
%|       restriction, including without limitation the rights to use,   |
%|       copy, modify, merge, publish, distribute, sublicense, and/or    |
%|       sell copies of the Software, and to permit persons to whom the  |
%|       Software is furnished to do so, subject to the following        |
%|       conditions:                                                     |
%|                                                                       |
%|       The above copyright notice and this permission notice shall be  |
%|       included in all copies or substantial portions of the Software. |
%|                                                                       |
%|       THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, |
%|       EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES |
%|       OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND        |
%|       NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT     |
%|       HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,    |
%|       WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING    |
%|       FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR   |
%|       OTHER DEALINGS IN THE SOFTWARE.                                 |
%}
%|                                                                       |
%| CHANGES :                                                             |
%|       <none>                                                          |
%|                                                                       |
%+-----------------------------------------------------------------------+

function [Pattern] = Machine_Emulation(varargin)
    if nargin<1
        Cams = [3, 3];
        bSteps = 25;
    elseif nargin == 1
        Cams(1) = varargin{1};
        Cams(2) = varargin{1};
        bSteps = 25;
    elseif nargin == 2
        Cams(1) = varargin{1};
        Cams(2) = varargin{2};
        bSteps = 25;
    elseif nargin == 3
        Cams(1) = varargin{1};
        Cams(2) = varargin{2};
        bSteps = varargin{3};
    end

    Braid_Even = zeros(1,Cams(1)*2+1);
    Braid_Odd = Braid_Even-1;
    Braid_Even(2:2:end) = 1;
    Braid_Odd(1:2:end) = 1;
    Braid_Odd1 = Braid_Odd;
    Braid_Odd2 = Braid_Odd;

    Braid_Odd1(4:4:end) = -2;
    Braid_Odd2(2:4:end) = -2;

    Braid = zeros(Cams*2+1)';

    for n = 1:2*Cams(2)+1
        if mod(n,2)
            Braid(n,:) = Braid_Even;
        elseif mod(n,4)
            Braid(n,:) = Braid_Odd1;
        else
            Braid(n,:) = Braid_Odd2;
        end
    end

    temp = Braid';
    in = find(temp==1);
    temp(in) = 1:length(in);
    Braid = temp';

    Braid_start = Braid;
    Pattern{1} = Braid;
    Braid = zeros(size(Braid));

    n = 1;
    while n<bSteps
        if n == 1
            Braid = Braid_start;
        end
        n = n + 1;
        if ~mod(n,2)
            Braid = RotateO(Braid);

        else
            Braid = RotateE(Braid);
        end
        Pattern{n} = Braid;
    end
end

function [Braid] = RotateO(Braid)
    [y,x] = find(Braid==-1);
    for n = 1:length(x)
        Braid_Section = Braid(y(n)-1:y(n)+1,x(n)-1:x(n)+1);
        Braid_Section = Rotate1(Braid_Section);
        Braid(y(n)-1:y(n)+1,x(n)-1:x(n)+1) = Braid_Section;
    end
end
function [Braid] = RotateE(Braid)
    [y,x] = find(Braid==-2);
    for n = 1:length(x)
        Braid_Section = Braid(y(n)-1:y(n)+1,x(n)-1:x(n)+1);
        Braid_Section = Rotate2(Braid_Section);
        Braid(y(n)-1:y(n)+1,x(n)-1:x(n)+1) = Braid_Section;
    end
end
function [Braid_Section] = Rotate1(Braid_Section)
    Braid_Section([4,6]) = Braid_Section([6,4]);
    Braid_Section = Braid_Section';
end
function [Braid_Section] = Rotate2(Braid_Section)
    Braid_Section([2,8]) = Braid_Section([8,2]);
    Braid_Section = Braid_Section';
end
```

Listing C.5: Function that further subdivides the braid patterns to generate the individual paths of the yarn.

```
%+--------------------------------------------------------------------------+
%|                                                                          |
%| FILENAME : Plot_Path_m                                   VERSION : 1.0.0 |
%|                                                                          |
%| TITLE : Plot Paths from Pattern                   AUTHOR : Daniel Aldrich |
%|                                                                          |
%+--------------------------------------------------------------------------+
%|                                                                          |
%| DEPENDENT FILES :                                                        |
%|       <none>                                                             |
%|                                                                          |
%| DESCRIPTION :                                                            |
%|       <none>                                                             |
%|                                                                          |
%| PUBLIC FUNCTIONS :                                                       |
%|       <none>                                                             |
%|                                                                          |
%| NOTES :                                                                  |
%|       <none>                                                             |
%|                                                                          |
%| COPYRIGHT :                                                              |
%|       Copyright (c) 2017 Daniel Aldrich                                  |
%{
%|       Permission is hereby granted, free of charge, to any person       |
%|       obtaining a copy of this software and associated documentation    |
%|       files (the "Software"), to deal in the Software without           |
%|       restriction, including without limitation the rights to use,      |
%|       copy, modify, merge, publish, distribute, sublicense, and/or      |
%|       sell copies of the Software, and to permit persons to whom the    |
%|       Software is furnished to do so, subject to the following          |
%|       conditions:                                                        |
%|                                                                          |
%|       The above copyright notice and this permission notice shall be    |
%|       included in all copies or substantial portions of the Software.   |
%|                                                                          |
%|       THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,   |
%|       EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES   |
%|       OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND          |
%|       NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT       |
%|       HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,      |
%|       WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING      |
%|       FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR     |
%|       OTHER DEALINGS IN THE SOFTWARE.                                    |
%}
%|                                                                          |
%| CHANGES :                                                                |
%|       <none>                                                             |
%|                                                                          |
%+--------------------------------------------------------------------------+

function [x,y,z] = Plot_Path(Pattern,n,res,theta,maxAngle)
patternLength = length(Pattern);
x(patternLength) = 0;
```

```
y(patternLength) = 0;
z(patternLength) = 0;

[tx1,ty1] = find(Pattern{1}==-1);
[tx2,ty2] = find(Pattern{1}==-2);

for i = 1:patternLength
    temp = Pattern{i};

    [x(i),y(i)] = find(temp==n);

    if i == 1
        z(1) = 0;
    else
        z(i) = z(i-1) - 1;
    end
end

if theta > 2
    X = x(1);
    Y = y(1);
    Z = z(1);
    angles = linspace(0,pi/2,theta);
    angles = angles(2:end-1);
    for n = 2:i
        if x(n)~=x(n-1) && y(n)~=y(n-1)

            if ~mod(n, 2)
                % -1 Rotates CW
                [~,index] = min((tx1-x(n-1)).^2 + (ty1-y(n-1)).^2);
                [t,~] = cart2pol(x(n-1)-tx1(index),y(n-1)-ty1(index));
                Ztheta = linspace(z(n-1),z(n),theta);
                [Xtheta,Ytheta] = pol2cart(t-angles,1);


                temp = Xtheta((angles>=maxAngle)&(angles<=(pi/2-maxAngle)));

                if length(temp)>1
                    x1 = temp(1);
                    x2 = temp(end);

                    temp = Ytheta((angles>=maxAngle)&(angles<=(pi/2-maxAngle)));

                    y1 = temp(1);
                    y2 = temp(end);
                end
                    Xtheta = Xtheta + tx1(index);
                    Ytheta = Ytheta + ty1(index);
                if length(temp)>1
                    m = (x2 - x1)/(y2 - y1);
                    b = y2 - m*x2;

                    r = @(t) (b)./(-cos(t)/m+sin(t));

                    ang = angles(angles>=maxAngle & angles<=(pi/2-maxAngle));
                    ind = find(angles>=maxAngle & angles<=(pi/2-maxAngle));

                    [tempx, tempy] = pol2cart(t-ang,r(t-ang));
                    dx = diff(tempx);
                    dy = diff(tempy);

                    nlen = sqrt(dx.^2 + dy.^2)/norm([y2 - y1,x2 - x1]);
                    zlen = Ztheta(ind(end)+1)-Ztheta(ind(1)+1);
                    tempz = cumsum(zlen*nlen);
                    Ztheta(ind+1) = [0,tempz]+Ztheta(ind(1)+1);
                    Xtheta(ind) = tempx + tx1(index);
                    Ytheta(ind) = tempy + ty1(index);
                end
            else
                % -2 Rotates CCW
                [~,index] = min((tx2-x(n-1)).^2 + (ty2-y(n-1)).^2);
                [t,~] = cart2pol(x(n-1)-tx2(index),y(n-1)-ty2(index));
                Ztheta = linspace(z(n-1),z(n),theta);
                [Xtheta,Ytheta] = pol2cart(t+angles,1);

                temp = Xtheta((angles >= maxAngle) & (angles <=(pi/2-maxAngle)));
                if length(temp)>1
                    x1 = temp(1);
                    x2 = temp(end);

                    temp = Ytheta((angles>=maxAngle)&(angles<=(pi/2-maxAngle)));

                    y1 = temp(1);
                    y2 = temp(end);
                end
                    Xtheta = Xtheta + tx2(index);
                    Ytheta = Ytheta + ty2(index);
                if length(temp)>1
                    m = (x2 - x1)/(y2 - y1);
                    b = y2 - m*x2;

                    r = @(t) (b)./(-cos(t)/m+sin(t));

                    ang = angles(angles>=maxAngle & angles<=(pi/2-maxAngle));
                    ind = find(angles>=maxAngle & angles<=(pi/2-maxAngle));

                    [tempx, tempy] = pol2cart(t+ang,r(t+ang));

                    dx = diff(tempx);
                    dy = diff(tempy);

                    nlen = sqrt(dx.^2 + dy.^2)/norm([y2 - y1,x2 - x1]);
                    zlen = Ztheta(ind(end)+1)-Ztheta(ind(1)+1);
                    tempz = cumsum(zlen*nlen);
                    Ztheta(ind+1) = [0,tempz]+Ztheta(ind(1)+1);

                    Xtheta(ind) = tempx + tx2(index);
                    Ytheta(ind) = tempy + ty2(index);
                end
            end
            X = [X,Xtheta,x(n)];
            Y = [Y,Ytheta,y(n)];
            Z = [Z,Ztheta(2:end)];
        else
            X = [X,ones(1,theta-1)*x(n)];
            Y = [Y,ones(1,theta-1)*y(n)];
            Z = [Z,-[1:theta-1]/(theta-1)+z(n-1)];
        end
    end
else
    X = x;
    Y = y;
    Z = z;
end
z = Z;
if res>1
    Z = z(1):(z(2) - z(1))/res:z(end);
    x = spline(z,X,Z);
    y = spline(z,Y,Z);
else
   x = X(:);
   y = Y(:);
end
z = Z(:);
z = linspace(Z(1),Z(end),length(Z));
```

```
x = x(:);
y = y(:);
x = x';
y = y';
z = z(:)';
```

C.3 Estimated Properties Output

Listing C.6: .

```
+=============================================================================+
|                                                                             |
|                              BRAID PROPERTIES                              |
|                                                                             |
|                            28-Feb-2018 11:40:00                            |
|                                                                             |
+=============================================================================+
+=============================================================================+
|                                                                             |
|                                 MATERIALS                                   |
|                                                                             |
|     Fibers: ARAMID_29                                                       |
|     Matrix: EPOXY                                                           |
|                                                                             |
|                                 PROPERTIES                                  |
|                                                                             |
|     Long. Young's Modulus       :  13.149 - 18.237   [GPa]                  |
|     Young's Modulus        (FGM):  16.364 [GPa]                             |
|     Yarn Packing Fraction  (Pf):   0.750                                   |
|     Ave Braiding Angle (Theta):  11.613 [deg]                              |
|     Interior Braiding Angle    :  15.476 [deg]                              |
|     Surface Braiding Angle     :   7.882 [deg]                              |
|     Braiding Tightness   (eta):   0.335                                    |
|     Fiber Volume Fraction (Vf):   0.258                                    |
|                                                                             |
|                              MODEL PROPERTIES                               |
|                                                                             |
|     Braid Cross-section Dims. :  1.051 x 1.051  [mm]                        |
|     Unit Cell Height (Pitch)  :          1.432  [mm]                        |
|     Number of Active Cams     :         3 by 3  ( 9)                        |
|     Spline Knots per Step     :             10                              |
|     2-Way Conv. Smoothing     :        Enabled                              |
|     Number of Yarns           :             24                              |
|     Spacing Factor            :          1.000                              |
|                                                                             |
+=============================================================================+
+=============================================================================+
|                                                                             |
|                        FIBER PROPERTIES: ARAMID_29                          |
|                                                                             |
|     Density :     1.44  [g/cc]                                              |
|     Denier  :      200  [g/9000m]                                           |
|     Radius  :    0.070  [mm]                                                |
|                                                                             |
+---------------+---------------+---------------------+-----------------------+
|      E1       |      E2       |   Poisson's Ratio   |    Shear Modulus      |
+---------------+---------------+---------------------+-----------------------+
|    61.000     |     4.200     |        0.350        |        2.900          |
+=============================================================================+
+=============================================================================+
|                                                                             |
|                        MATRIX PROPERTIES: EPOXY                             |
|                                                                             |
+-------------------------+-------------------------+-------------------------+
|    Young's Modulus    |    Poisson's Ratio    |     Shear Modulus     |
+-----------------------+-----------------------+-----------------------+
|         3.400         |         0.300         |         1.308         |
+=======================================================================+
+=======================================================================+
|                                                                       |
|         YARN PROPERTIES: ARAMID_29/EPOXY (PF: 0.750)                  |
|                                                                       |
+-----------------------------------------------------------------------+
|                            Young's Modulus                            |
+-----------------------+-----------------------+-----------------------+
|          E1           |          E2           |          E3           |
+-----------------------+-----------------------+-----------------------+
|        46.600         |         3.967         |         3.967         |
+-----------------------+-----------------------+-----------------------+
|                            Poisson's Ratio                            |
+-----------------------+-----------------------+-----------------------+
|          v12          |          v13          |          v23          |
+-----------------------+-----------------------+-----------------------+
|         0.337         |         0.337         |         0.510         |
+-----------------------+-----------------------+-----------------------+
|                             Shear Modulus                             |
+-----------------------+-----------------------+-----------------------+
|          G12          |          G13          |          G23          |
+-----------------------+-----------------------+-----------------------+
|         2.223         |         2.223         |         4.049         |
+=======================================================================+
+=======================================================================+
|                                                                       |
|       ARAMID_29/EPOXY 0.00 DEG (VY: 0.343, PF: 0.750)                 |
|                                                                       |
+-----------------------------------------------------------------------+
|                            Young's Modulus                            |
+-----------------------+-----------------------+-----------------------+
|          E1           |          E2           |          E3           |
+-----------------------+-----------------------+-----------------------+
|        18.237         |         3.575         |         3.575         |
+-----------------------+-----------------------+-----------------------+
|                            Poisson's Ratio                            |
+-----------------------+-----------------------+-----------------------+
|          v12          |          v13          |          v23          |
+-----------------------+-----------------------+-----------------------+
|         0.313         |         0.313         |         0.304         |
+-----------------------+-----------------------+-----------------------+
|                             Shear Modulus                             |
+-----------------------+-----------------------+-----------------------+
|          G12          |          G13          |          G23          |
+-----------------------+-----------------------+-----------------------+
|         1.523         |         1.523         |         2.569         |
+=======================================================================+
+=======================================================================+
|                                                                       |
|       ARAMID_29/EPOXY 11.61 DEG (VY: 0.343, PF: 0.750)                |
|                                                                       |
+-----------------------------------------------------------------------+
|                            Young's Modulus                            |
+-----------------------+-----------------------+-----------------------+
|          E1           |          E2           |          E3           |
+-----------------------+-----------------------+-----------------------+
|        13.149         |         3.545         |         3.572         |
+-----------------------+-----------------------+-----------------------+
|                            Poisson's Ratio                            |
+-----------------------+-----------------------+-----------------------+
|          v12          |          v13          |          v23          |
+-----------------------+-----------------------+-----------------------+
|         0.377         |         0.265         |         0.289         |
+-----------------------+-----------------------+-----------------------+
|                             Shear Modulus                             |
+---------------------+---------------------+---------------------+
|         G12         |         G13         |         G23         |
+---------------------+---------------------+---------------------+
|        1.629        |        1.543        |        2.487        |
+=================================================================+
+=================================================================+
|                                                                 |
|                     FABRIC GEOMETERY MODEL                      |
|                                                                 |
+-----------------------------------------------------------------+
|                         Young's Modulus                         |
+---------------------+---------------------+---------------------+
|         E1          |         E2          |         E3          |
+---------------------+---------------------+---------------------+
|       16.364        |        4.937        |        4.937        |
+---------------------+---------------------+---------------------+
|                         Poisson's Ratio                         |
+---------------------+---------------------+---------------------+
|         v12         |         v13         |         v23         |
+---------------------+---------------------+---------------------+
|        0.360        |        0.360        |        0.241        |
+---------------------+---------------------+---------------------+
|                          Shear Modulus                          |
+---------------------+---------------------+---------------------+
|         G12         |         G13         |         G23         |
+---------------------+---------------------+---------------------+
|        1.980        |        1.980        |        1.569        |
+=================================================================+
```

Appendix D: Model Generation Code

Listing D.1: This is a Test

```
Dim swApp As Object
Dim Part As Object
Dim PartExt As SldWorks.ModelDocExtension
Dim doc As Object
Dim boolstatus As Boolean
Dim longstatus As Long, longwarnings As Long
Dim myFeature As Object
Dim skSegment As Object
Dim i As Integer
Dim t As Long
Dim CWD As String
Dim bRet As Boolean
Dim partYarns As String
Dim partResin As String
Dim partBraid As String
Dim filename As String

Public Declare PtrSafe Function GetTickCount Lib "kernel32.dll" () As Long

Sub main()
    Set swApp = Application.SldWorks

'Part 1

    Set Part = swApp.NewDocument( _
      "C:\ProgramData\SolidWorks\SOLIDWORKS 2015\templates\Part.prtdot", _
      0, 0, 0)
    Set PartExt = Part.Extension
    partYarns = Part.GetTitle
    CWD = swApp.GetCurrentMacroPathFolder()
    swApp.SetCurrentWorkingDirectory (CWD)
    t = GetTickCount
    Open CWD & "\Data.txt" For Input As #1
    Input #1, Count, Radius, Length, bW, bL, SF
    Close #1

    Set Part = swApp.ActiveDoc
    Dim myModelView As Object
    Set myModelView = Part.ActiveView
    myModelView.FrameState = swWindowState_e.swWindowMaximized
    For i = 1 To Count
        boolstatus = Part.InsertCurveFile(CWD & "\Yarn" & i & ".txt")

        Part.ClearSelection2 True
    Next i

    For i = 1 To Count
        boolstatus = Part.Extension.SelectByID2("Top Plane", "PLANE", _
          0, 0, 0, False, 0, Nothing, 0)
        Part.SketchManager.InsertSketch True

        Part.ClearSelection2 True

        Open CWD & "\Yarn" & i & ".txt" For Input As #1
        Input #1, X, Y, Z
        Close #1

        Set skSegment = Part.SketchManager.CreateCircleByRadius( _
          X / 1000, -Z / 1000, 0#, Radius)
        Part.ClearSelection2 True
        Part.SketchManager.InsertSketch True

        Part.ClearSelection2 True

        boolstatus = Part.Extension.SelectByID2( _
          "Sketch" & i, "SKETCH", 0, 0, 0, False, 1, Nothing, 0)
        boolstatus = Part.Extension.SelectByID2( _
          "Curve" & i, "REFERENCECURVES", 0, 0, 0, True, 4, Nothing, 0)


        Set myFeature = Part.FeatureManager.InsertProtrusionSwept3( _
          False, False, swTwistControlType_e.swTwistControlFollowPath, _
          False, False, swTangencyType_e.swTangencyNone, _
          swTangencyType_e.swTangencyNone, False, 0, 0, _
          swThinWallType_e.swThinWallOneDirection, 10, _
          True, True, True, 0, True)
        Part.ClearSelection2 True
    Next i

    Part.ShowNamedView2 "*Isometric", 7
    Part.ViewZoomtofit2
    Part.SetMaterialPropertyName2 "Default", _
      "C:/ProgramData/SolidWorks/SOLIDWORKS 2015/"& _
      "Custom Materials/Custom Materials.sldmat", "Aramid"
    filename = CWD & "\Yarns.SLDPRT"
    boolstatus = PartExt.SaveAs(filename, swSaveAsCurrentVersion, _
      swSaveAsOptions_Silent, Nothing, longstatus, longwarnings)

'Part 2

    Set Part = swApp.NewDocument( _
      "C:\ProgramData\SolidWorks\SOLIDWORKS 2015\templates\Part.prtdot", _
      0, 0, 0)
    Set PartExt = Part.Extension
    partResin = Part.GetTitle
    swApp.ActivateDoc2 "Part2", False, longstatus

    boolstatus = Part.Extension.SelectByID2("Top Plane", "PLANE", 0, 0, 0, _
      False, 0, Nothing, 0)
    Part.SketchManager.InsertSketch True
    Part.ClearSelection2 True
    Part.ShowNamedView2 "*Top", 5
    boolstatus = Part.Extension.SetUserPreferenceToggle( _
      swUserPreferenceToggle_e.swSketchAddConstToRectEntity, _
      swUserPreferenceOption_e.swDetailingNoOptionSpecified, True)
    boolstatus = Part.Extension.SetUserPreferenceToggle( _
      swUserPreferenceToggle_e.swSketchAddConstLineDiagonalType, _
      swUserPreferenceOption_e.swDetailingNoOptionSpecified, True)
    Dim vSkLines As Variant
    vSkLines = Part.SketchManager.CreateCenterRectangle(0, 0, 0, _
      (4 * SF * bW + 3) * Radius / 2, (4 * SF * bL + 3) * Radius / 2, 0)
    Part.ClearSelection2 True

    Part.SketchManager.InsertSketch True
    Part.ClearSelection2 True
    boolstatus = Part.Extension.SelectByID2("Sketch1", "SKETCH", _
      0, 0, 0, False, 4, Nothing, 0)
    Set myFeature = Part.FeatureManager.FeatureExtrusion2(True, False, _
      True, 0, 0, Length, 0.01, False, False, False, False, 0, 0, _
      False, False, False, False, True, True, True, 0, 0, False)
    Part.SelectionManager.EnableContourSelection = False

    Part.ClearSelection2 True

    Part.SetMaterialPropertyName2 "Default", _
      "C:/Program Files/SOLIDWORKS Corp/SOLIDWORKS (2)/"& _
      "lang/english/sldmaterials/SOLIDWORKS Materials.sldmat", _
      "Epoxy, Unfilled"
    filename = CWD & "\Resin.SLDPRT"
    boolstatus = PartExt.SaveAs(filename, swSaveAsCurrentVersion, _
      swSaveAsOptions_Silent, Nothing, longstatus, longwarnings)

'Assembly

    Set Part = swApp.NewDocument("C:\ProgramData\SolidWorks\"& _
      "SOLIDWORKS 2015\templates\Assembly.asmdot", 0, 0, 0)
    Set PartExt = Part.Extension
    swApp.ActivateDoc2 "Assem#", False, longstatus
    partBraid = Part.GetTitle
    Set Part = swApp.ActiveDoc

    boolstatus = Part.AddComponent(CWD & "\Yarns.SLDPRT", 0, 0, 0)
    Set Part = swApp.ActiveDoc
    boolstatus = Part.AddComponent(CWD & "\Resin.SLDPRT", 0, 0, 0)
    Set Part = swApp.ActiveDoc

    boolstatus = Part.Extension.SelectByID2("Point1@Origin@Yarns-1@" & _
      partBraid, "EXTSKETCHPOINT", 0, 0, 0, True, 1, Nothing, 0)
    boolstatus = Part.Extension.SelectByID2("Point1@Origin@Resin-1@" & _
      partBraid, "EXTSKETCHPOINT", 0, 0, 0, True, 1, Nothing, 0)
    Dim myMate As Object
    Set myMate = Part.AddMate5(20, -1, False, 0, 0, 0, 0, 0, 0, 0, 0, _
      False, False, 0, longstatus)
    Part.ClearSelection2 True
    Part.EditRebuild3

    Part.ShowNamedView2 "*Isometric", 7
    Part.ViewZoomtofit2

    filename = CWD & "\Braid.SLDASM"
    boolstatus = PartExt.SaveAs(filename, swSaveAsCurrentVersion, _
      swSaveAsOptions_Silent, Nothing, longstatus, longwarnings)

    boolstatus = Part.Extension.SelectByID2("Resin-1@Braid", "COMPONENT", _
      0, 0, 0, False, 0, Nothing, 0)
    Part.AssemblyPartToggle
    Part.EditPart
    Part.ClearSelection2 True
    boolstatus = Part.Extension.SelectByID2("Yarns-1@Braid", "COMPONENT", _
      0, 0, 0, True, 0, Nothing, 0)
    Part.InsertCavity4 0, 0, 0, True, 1, -1
    Part.AssemblyPartToggle
    Part.EditAssembly

    boolstatus = PartExt.SaveAs(filename, swSaveAsCurrentVersion, _
      swSaveAsOptions_Silent, Nothing, longstatus, longwarnings)

    swApp.ActivateDoc2 "Resin.SLDPRT", False, longstatus
    Set Part = swApp.ActiveDoc
    Set PartExt = Part.Extension
    filename = CWD & "\Resin.SLDPRT"
    boolstatus = PartExt.SaveAs(filename, swSaveAsCurrentVersion, _
      swSaveAsOptions_Silent, Nothing, longstatus, longwarnings)
    swApp.ActivateDoc2 "Braid.SLDASM", False, longstatus
    Set Part = swApp.ActiveDoc
    Set PartExt = Part.Extension
    filename = CWD & "\Braid.SLDASM"
    boolstatus = PartExt.SaveAs(filename, swSaveAsCurrentVersion, _
      swSaveAsOptions_Silent, Nothing, longstatus, longwarnings)

    swApp.CloseDoc "Resin.SLDPRT"
    swApp.CloseDoc "Yarns.SLDPRT"

    boolstatus = Part.Extension.SelectByID2("Resin-1@Braid", "COMPONENT", _
      0, 0, 0, False, 0, Nothing, 0)
    boolstatus = Part.SetComponentTransparent(True)

    Part.ClearSelection2 True

    MsgBox GetTickCount - t, , "Milliseconds"
End Sub
```