# GENERATION MILLING OF CYLINDRICAL INVOLUTE GEARS

by

Jesse Michael Groover

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Mechanical Engineering

Charlotte

2018

Approved by:

______________________________

Dr. Gert Goch

______________________________

Dr. Jimmie Miller

______________________________

Dr. Tony Schmitz

## ABSTRACT

JESSE MICHAEL GROOVER. Generation Milling of Cylindrical Involute Gears.
(Under the direction of DR. GERT GOCH)

In the past, gears have largely been manufactured by means of dedicated gear manufacturing machines and tools. These are difficult and expensive to manufacture and maintain, and offer low flexibility and an inability to make anything other than gears. This thesis attempts to offer a method by which internal and external cylindrical involute gears can be machined on a conventional five-axis milling center used for other manufacturing processes, in conjunction with low cost commonly used cylindrical end mills. The algorithms are based on a parametric vectorial model of the involute profile, and the involute flanks are machined in accordance with the generation principle. A machine with a C axis stacked on a B axis is used. Specifically, the rotary table (C axis) of a five axis milling machine is used to rotate the work piece, while the tool is caused to travel within a plane tangent to the base circle. For helical gears, the B axis is rotated to account for the helix angle. External and internal, spur and helical gears have been machined and measured, and flanks with form deviations less than 5 $\mu m$ have been produced.

# DEDICATION

To my parents, for making this degree possible.

And to my future children, for making it necessary.

# ACKNOWLEDGEMENTS

The author would like to thank Professor Dr. Gert Goch for his invaluable support, encouragement, and insight that was vital to the completion of the project. Professor Dr. John Ziegert provided advising and technical expertise related to the project. Ph.D. students Dr. Kang Ni and Yue Peng were very helpful in both thinking through problems, and performing the measurements to asses the quality of the produced gears. Mr. Greg Caskey of the UNC Charlotte Center for Precision Metrology performed the surface roughness measurements. The Mori Seiki NMV5000DCG was provided by the Machine Tool Technologies Research Foundation (MTTRF).

The congregation of Matthews Orthodox Presbyterian Church provided countless meals, stimulating conversations, and true friendship. In particular, the Trice family, the Fawcett family, the Cleveland family, the English family, and the McGowan family, have been true friends, and I would not be the same without them.

And finally, my Lord and Savior Jesus Christ, who died for my sins, and provided all the blessings listed above, among others.

TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

$\alpha_t$ Transverse Pressure Angle

$\alpha_{t,a}$ Transverse Pressure Angle at Tooth Tip

$\alpha_{t,i}$ Transverse Pressure Angle at Some Arbitrary Point $i$ Along Involute

$\beta$ Helix Angle at Refence Cylinder

$\beta_b$ Helix Angle at Base Cylinder

$\eta_b$ Tooth Space Half Width Angle

$\gamma$ Angle for Determining Roll Angle Limits

$\Lambda$ Offset Angle

$\Lambda_0$ Initial Offset Angle

$\Lambda_S$ Offset Angle for Determining Which Flank to Cut

$\Lambda_\beta$ Offset Angle for Taking Helix Angle Into Account

$\Lambda_d$ Offset Angle Corresponding to Some Stock Thickness $d$

$\Lambda_{maxstock}$ Offset Angle Corresponding to Maximum Stock Thickness

$\Lambda_{N,k}$ Offset Angle for Cutting Tooth $k$

$\Lambda_{step}$ Offset Angle due to Successive Axial Steps

$\Lambda_{total}$ Total Offset Angle

$\Lambda_{z_b}$ Offset Angle to Account For Helix Angle

$\mathbf{H}$ Homogeneous Transformation Matrix

$\mathbf{n_G}$ Surface Normal Vector according to [28]

$\mathbf{n_{basic}}$ Basic Representation of Surface Normal Vector

$\mathbf{n_{transverse}}$ Surface Normal Vector in Transverse Plane

$\mathbf{N_{xy}}$ Scaled Transverse Normal Vector

$\mathbf{N}$ Scaled Normal Vector

$\mathbf{n}$ Surface Normal Vector

$\mathbf{P}$ Vectorial Representation of Point on Involute Surface

$\mathbf{Q}$ Tool Center Point Commanded Position

$\mathbf{R}$ $2 \times n$ Matrix of Rotary Positions

$\mathbf{T}$ Tool Orientation Vector

$\mathbf{V_{g,m}}$ Velocity of Gear Relative to Machine

$\mathbf{V_{t,g}}$ Velocity of Tool Relative to Gear

$\mathbf{V_{t,m}}$ Velocity of Tool Relative to Machine

$\mathbf{V}$ Tool Travel Direction Vector

$\omega$ Angular Speed of Workpiece

$\psi_b$ Tooth Thickness Half Width Angle

$\xi$ Roll angle

$\xi_0$ Initial Roll Angle

$\xi_a$ Roll Angle at Tooth Tip

$\xi_f$ Final Roll Angle

$\xi_i$ Roll Angle at Some Arbitrary Point $i$ Along Involute

$\xi_p$ Roll angle at point p

$\xi_{f,lim}$ Limiting Roll Angle for an Internal Helical Gear

$\xi_{inner}$ Roll Angle at Inner Radius

$\xi_{maximum}$ Roll Angle at Outer Radius

$b$ Gear Face Width

$d$ Scaling Factor for Unit Normal Vector

$d_{finalstock}$ Final Stock Thickness

$d_{maxstock}$ Maximum Stock Thickness

$dir$ Direction Switch for Tool Travel Direction Vector

$f$ Direction Switch for Involutes

$f_{cut}$ Involute Direction Switch, Taking Internal or External Gears into Account

$F_{INV}$ Inverse Time Feedrate Value

$F_{IPM}$ Linear Feedrate Value

$i$ Counter for Looping Through Axial Depth Passes

$inv\alpha_t$ The Involute of $\alpha_t$

$inv\alpha_{t,a}$ Involute of Transverse Pressure Angle at Tooth Tip

$j$ Counter for Looping Through Stock Thickness Passes

$k$ Counter for Looping Through Tooth Number

$L$ Vectorial position of Center of B Rotation Relative to Gear Coordinate System

$R_a$ Radius at Tooth Tip

$R_b$ The radius of the base cylinder

$R_i$ Radius at Some Arbitrary Point $i$ Along Involute

$R_p$ Radius of Pitch Circle

$R_{tip}$ Radius of Tip Circle

$r_{tool}$ Tool Radius

$Rot_B$ B Axis Rotation Angle

$Rot_C$ C Axis Rotation Angle

$s$ Arc Length of the Involute

$s_i$ Arc Length from the Base Circle to some Point $i$ Along the Involute

$t_i$ Time to Go From the Base Circle to some Point $i$ Along the Involute

$z$ Number of teeth

$z_b$ Axial Position

$z_{b,step}$ Axial Step Size

X X Axis Label

Y Y Axis Label

# CHAPTER 1: INTRODUCTION AND LITERATURE REVIEW

## 1.1 Introduction

In the past, gears have largely been manufactured by means of complex tooling and actuation systems [6]. The complexity, cost, and narrow applicability of the necessary systems, such as hobbing machines and tools, gear shapers, etc., has been inevitable due to limited computation power. With the rise of Computer Numerically Controlled (CNC) technology over the last few decades, new methods of gear manufacturing have been made possible, however these have been largely neglected.

One limitation of CNC machines is the rudimentary coding language that controls the axes motions. To profile cut a contour, the software usually must segment the curve into a finite number of linear or circular segments. This introduces form errors that are prohibitive for the flanks of gears. Most gear flanks follow an involute profile [2, 3], which has certain properties suggesting that it is possible to use the generation principle to mill high quality gears without the need for linear approximation. Simultaneous rotary and linear motions are used to achieve low form deviations. In previous research, external gears have been manufactured using a slot cutter on a five axis CNC machine [7], although this is not the preferred method in industry.

Internal gears have been widely used since planetary gear stages became standard in modern car transmissions, as well as other large components, such as gear boxes and pitch and yaw control mechanisms in wind energy systems [4]. But internal gears present geometrical limitations that inhibit certain production processes that are usual with other types of gears, such as hobbing for external gears. With the aid of a standard five axis machining center and standard cylindrical milling tools, high quality internal involute gears can be machined by means of the generation principle, without having to rely on the sophisticated machinery and tooling that was previously necessary.

This thesis investigates the use of a Mori Seiki NMV5000DCG five axis milling center and standard cylindrical end mills in the production of internal and external involute gears. Algorithms have been developed in MATLAB to generate the G-code leading the machine to follow generation principle motion. Inverse time feed rates, as opposed to standard millimeters per minute feed rates, have been utilized to achieve simultaneous rotary and linear motion, complying with the generation principle. Spur gears have been manufactured, and subsequently measured by means of a coordinate measuring device (Leitz PMMF302016) and the Quindos7 software. Gear flanks with profile deviations of less than 5 $\mu$m have been produced. Future work will include improvements in profile deviations and surface finish, as well as implementing flank modifications such as slope and crowning.

A Mori Seiki (now owned by DMG Mori) NMV5000DCG 5-axis milling machine with a Fanuc MSX-711III control was used for the experiments presented herein. The G code syntax is thus suitable for this Fanuc control, and may not be available on other controllers.

## 1.2 Literature Review

Regarding parametric representations of gear flanks, Hedlund et al. provided a parametric representation of gear flanks in [13] by modeling the reference rack, and approximating the resulting flank. Profile deviations from a pure involute of a few nanometers were found, so the result was good, but it was still an approximation, based on the reference rack profile. The benefit of this approach is that flank modifications can be defined parametrically in terms of modifications to the rack profile, and indirectly transferred to the flank profile. This makes it fairly simple to define flank modifications, but still does not directly give a true mathematical representation of the flank itself, in terms of the base gear parameters. Antoniadis et al. uses a similar approach in [19].

Kawalec in [20] gives a list, with mathematical definitions and descriptions, of mathematical functions that can be useful representations of gear flanks. All methods are numerical however, and are simply approximations or estimations of the surface, not a true analytical representation of the flank.

Flank modifications is an area which is not investigated in this thesis, but will be an important next step. In [9], flank modifications are defined as the areal change of flank geometry, described by local distances from the pure involute in the transverse plane. In his Ph.D. Dissertation, Ni represented deviations in the flanks of gears in terms of Chebychev polynomials superimposed onto a pure involute flank, which was mapped to a planar surface [12]. The primary focus was the metrology of gears, and describing either intended or unintended form deviations. The same approach could possibly be used to define intended form deviations for manufacturing, but more work is required in this area.

Regarding manufacturing of gears, hobbing is by far the most common method for green cutting of non-hardened external cylindrical gears [1, 3, 26]. Hobbing utilizes a screw like tool whose cross section, if cut through a plane intersecting the center axis of the tool, emulates a reference rack. A hob is shown next to a hobbed gear in Figure 1.1.

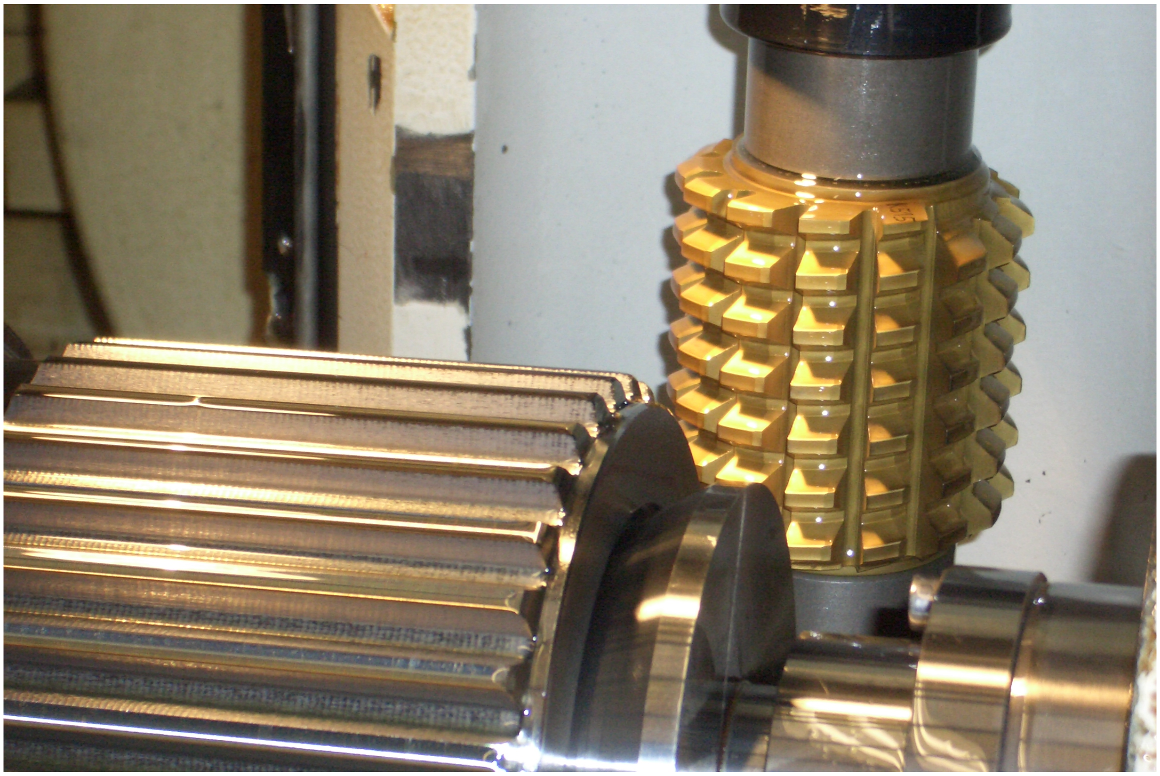

Figure 1.1: Gear Hobbing [30]

Another method is referred to as gear profile milling [1], and uses a specific tool shaped to match the shape of the tooth space. This method can only be used on gears with a single set of parameters such as module, pressure angle, pitch diameter, etc. This is shown in Figure 1.2.

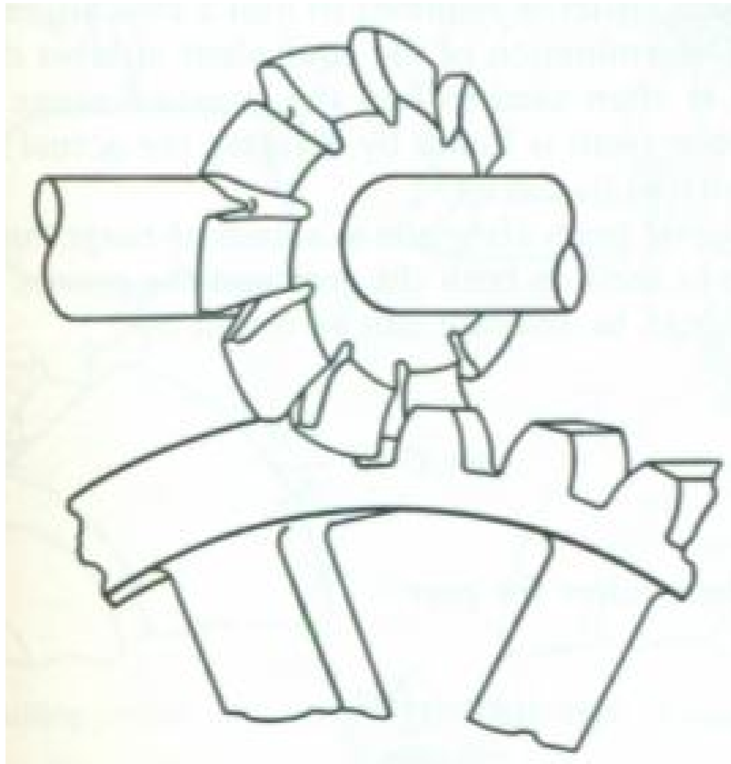

Figure 1.2: Gear Profile Milling [1]

Gear shaping [1, 21, 25] is another method of gear manufacture, which utilizes a tool shaped much like a gear, but with a relief from the bottom face of the gear, forming a cutting edge. This gear like tool is reciprocated in the axial direction, beginning with very light cuts to the gear to be manufactured, gradually moving inward with each pass, while rotating in generation motion.

Under certain conditions, gear shaping can be used for internal gears as well as external gears. An example is shown in Figure 1.3.

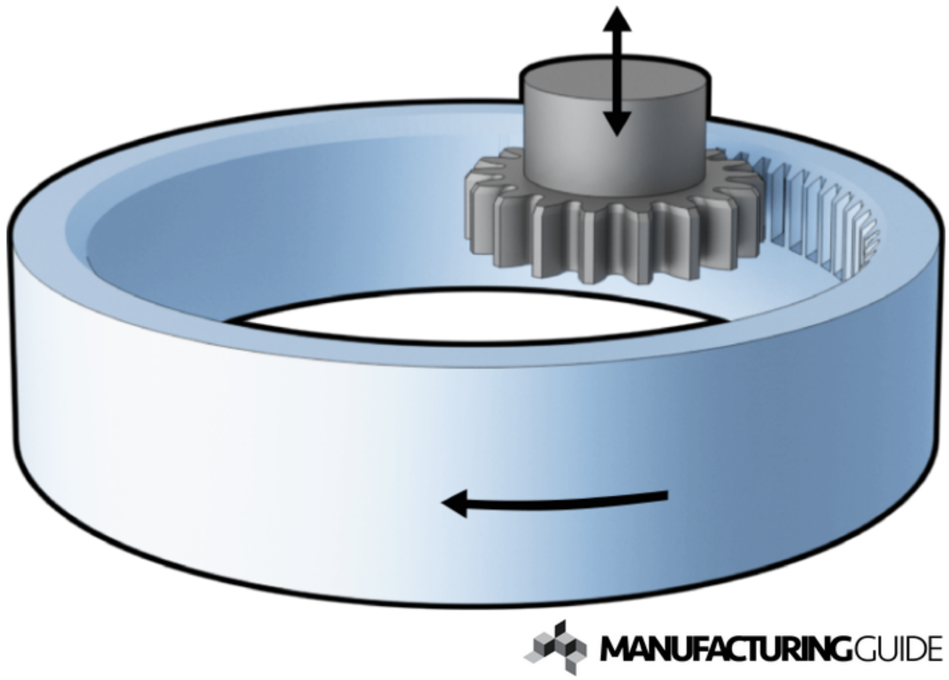


Figure 1.3: Gear Shaping of an Internal Gear [31]

One method specifically used for internal gears is broaching [1]. Broaching uses a long tool with successive cutting edges arranged along the axis of the tool, which is pulled through a pre-bored hole in the workpiece. As the successive cutting edges pull through the material, the final shape is cut. Again, a very specialized tool and machine must be used, which is expensive and inflexible. A cut away view of an internal gear being broached is shown in Figure 1.4.

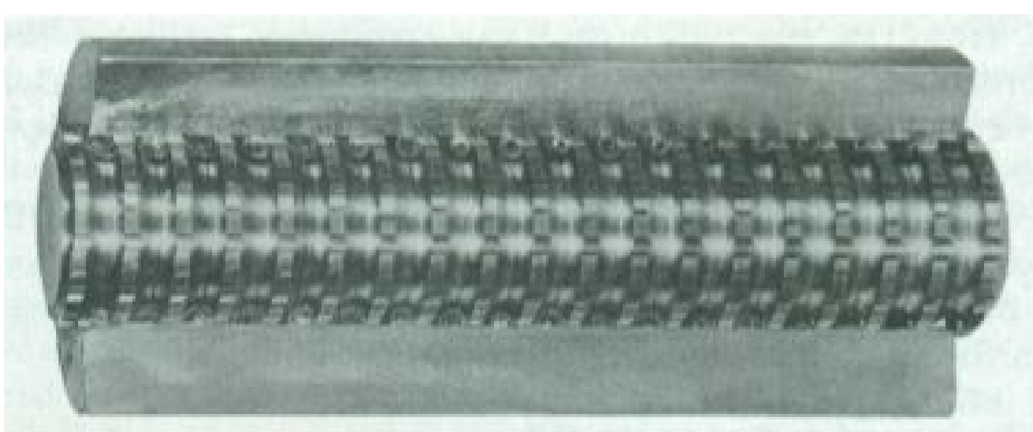

Figure 1.4: Gear Broaching, Cut Away View [1]

Other types of gears, such as bevel gears, spiral bevel gears, hypoid gears, etc. [23, 24], have their own methods of manufacture such as skiving [19] and face hobbing [22], for example.

A common theme in all of these manufacturing methods is complex machines and dedicated gear cutting tools. Both the machines and tools are expensive to manufacture and maintain. A comprehensive method of accurately manufacturing internal and external spur and helical cylindrical involute gears using conventional 5-axis milling machines common in manufacturing, and common, simple, inexpensive tools has not yet been adopted. This thesis attempts to offer a method to do just that.

# CHAPTER 2: THE INVOLUTE

## 2.1 Background

The use of gears in machinery of every sort has been ubiquitous for many years. Many different types of gears and tooth forms have been experimented with, including pin type gears, cycloidal gears, involute gears, etc. The involute has certain advantages that no other tooth forms possess, rendering it by far the most common tooth form in use today, for good reason. Namely, the involute tooth form is the only tooth form that guarantees constant velocity and torque ratios between interchangeable pairs of mating gears. This property leads to more constant loading conditions, lower fatigue stress, and thus longer lifetimes of the gears themselves, and the machinery in which they are used.

The involute can be visualized by imagining a cylinder of radius $R_b$ around which is wrapped a cord. If the cord is unwrapped from the cylinder, the end of the cord traces an involute relative to the stationary reference frame attached to the cylinder, as shown in Figure 2.1. Another way to represent this is with the same cylinder and cord, but instead of holding the cylinder stationary, it is rotated an angle $\xi$, while the end of the cord is pulled off along a straight line. The same involute is generated, relative to the now rotating reference frame attached to the cylinder. An angle $\Lambda$ denotes starting angular position of the cylinder.

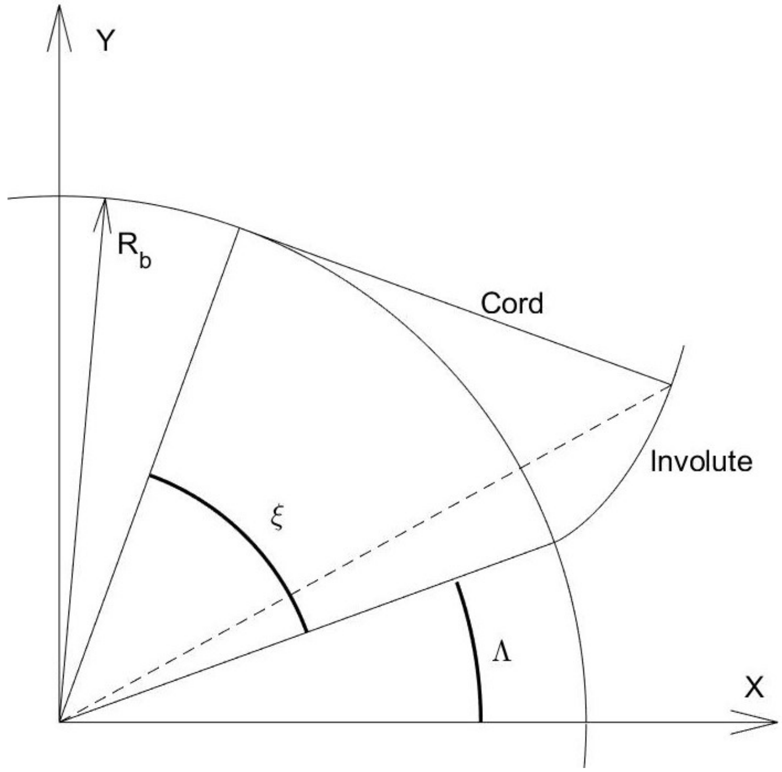


Figure 2.1: Basic Involute

This is called the "generation principle", and is the approach used in this thesis for the generation milling of involute gears. The gear, or workpiece, corresponds to the cylinder, which is rotated about its center, and the tool corresponds to the point at the end of the cord. The result is an involute curve generated truly by means of the generation principle. There are several main challenges to overcome, which will be discussed in more detail in the following chapters.

It can be seen that the involute curve is always normal to the cord, which is always tangent to a transverse cut through the cylinder, called the “base circle”. The base circle has radius $R_b$, as shown in Figure 2.1. If the curve defines a surface on which there is some contact force, neglecting friction, the force vector must always be normal to the surface. Therefore, the force vector must always be tangent to the base circle. As a result, the effective moment arm is a constant length.

If this gear is paired with a second gear also possessing an involute tooth form, then the force vector (again, neglecting friction), lies tangent to the base circles of both gears, along the red line shown in Figure 2.2. If a line is drawn connecting the center axes of the two base circles (Line A-A), then the point at which the force vector line (red line tangent to both base circles) crosses this center-to-center line is called the "pitch point", and the angle between the two lines is the complement of what is called the "transverse pressure angle", $\alpha_t$. This is shown in Figure 2.2.

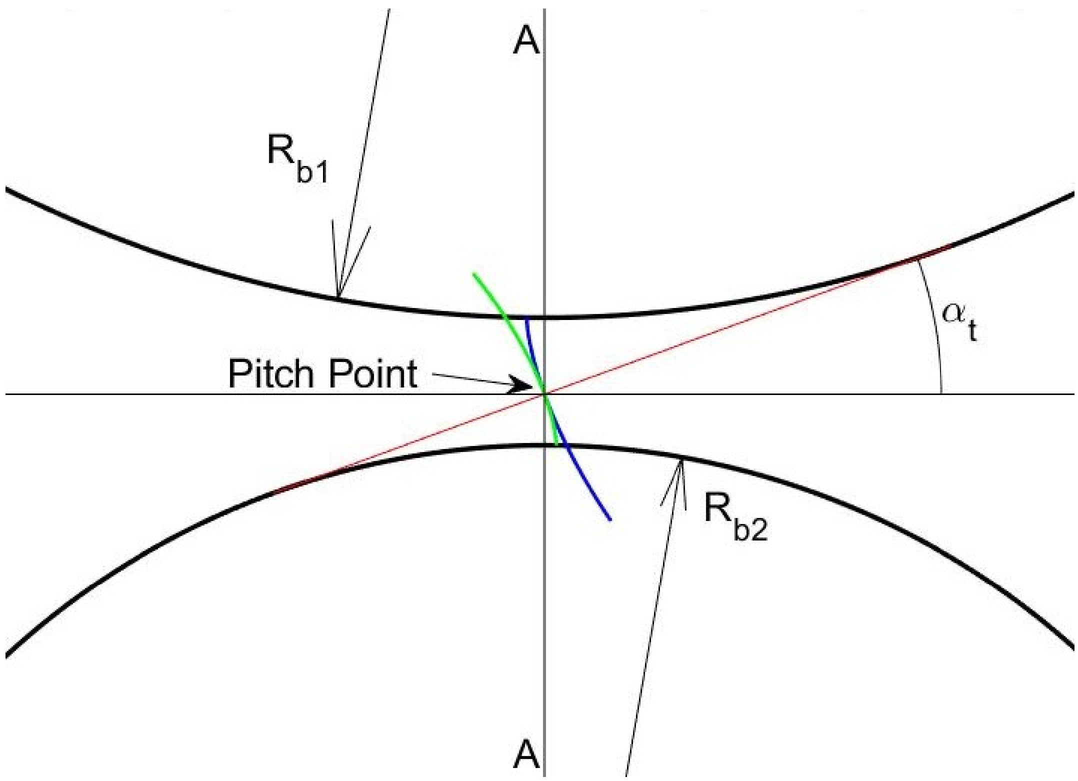


Figure 2.2: Two Mating Involutes

## 2.2 Mathematical Representations

The base equations for an involute are as follows [12, 3]

$$x = R_b(\cos(\xi + \Lambda) + \xi \sin(\xi + \Lambda))$$
$$y = R_b(\sin(\xi + \Lambda) - \xi \cos(\xi + \Lambda)) \tag{2.1}$$
$$z = z_b$$

where $R_b$ is the radius of the base cylinder, $\xi$ is the roll angle, or rotation of the base cylinder, and $\Lambda_0$ is the initial offset angle, or the angular start position of the base cylinder. The difference between the roll angle $\xi$ and the transverse pressure angle $\alpha_t$ is called the "involute of $\alpha_t$", denoted by $inv\alpha_t$.

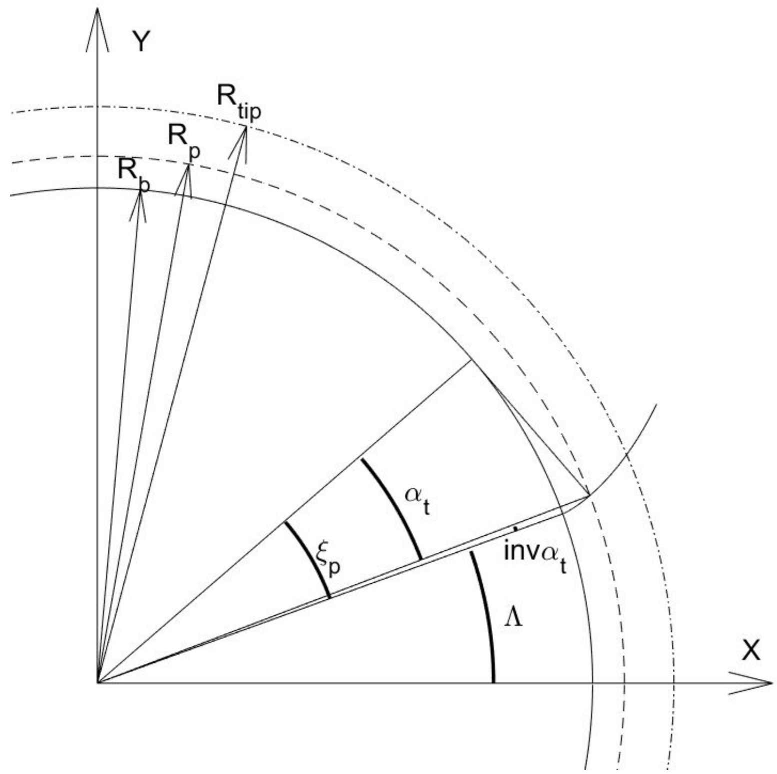


Figure 2.3: Basic Involute with Base, Reference, and Tip Radii

This can be represented in terms of the generation principle by subtracting the roll angle $\xi$ within the sine and cosine terms, and applied to the rotation position of the gear $C$, as follows:

$$\begin{aligned} x_{gen} &= R_b(\cos(\Lambda) + \xi \sin(\Lambda)) \\ y_{gen} &= R_b(\sin(\Lambda) - \xi \cos(\Lambda)) \\ z &= z_b \\ C &= \xi \end{aligned} \tag{2.2}$$

For the purposes of breaking the cut into steps for feedrate calculations (addressed further in Chapter 4), it can be useful to express the profile in terms of arc length $s$, instead of roll angle $\xi$. From [27], the arc length $s$ of a profile defined by parametric equations $x = f(t)$ and $y = g(t)$ is,

$$s(t) = \int_{t_0}^{t_f} \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} dt \tag{2.3}$$

In the case of an involute profile, the following equations are used.

$$\begin{aligned} x &= R_b cos(\xi) + R_b \xi sin(\xi) \\ y &= R_b sin(\xi) - R_b \xi cos(\xi) \end{aligned} \tag{2.4}$$

$$\begin{aligned} \frac{dx}{d\xi} &= -R_b sin(\xi) + R_b(\xi cos(\xi) + sin(\xi)) = R_b \xi cos(\xi) \\ \frac{dy}{d\xi} &= R_b cos(\xi) - R_b(-\xi sin(\xi) + cos(\xi)) = R_b \xi sin(\xi) \end{aligned} \tag{2.5}$$

Substituting Equation 2.5 into Equation 2.3 yields:

$$s(\xi) = \int_{\xi_0}^{\xi_f} \sqrt{(R_b \xi cos(\xi))^2 + (R_b \xi sin(\xi))^2} d\xi \tag{2.6}$$

$$s(\xi) = \int_{\xi_0}^{\xi_f} R_b \xi \sqrt{cos^2(\xi) + sin^2(\xi)} d\xi \tag{2.7}$$

$$s(\xi) = \int_{\xi_0}^{\xi_f} R_b \xi d\xi \tag{2.8}$$

$$s(\xi) = R_b \int_{\xi_0}^{\xi_f} \xi d\xi \tag{2.9}$$

$$s(\xi) = R_b \frac{1}{2} \xi^2 \Big|_{\xi_0}^{\xi_f} \tag{2.10}$$

If $\xi_0 = 0$ (assume starting at the base circle), then the equation reduces to

$$s(\xi) = \frac{1}{2} R_b \xi^2, \tag{2.11}$$

which, when solved for $\xi$ becomes

$$\xi(s) = \sqrt{\frac{2s}{R_b}}. \tag{2.12}$$

Substitution into Equations 2.1 and 2.2 gives

$$x = R_b \left( \cos\left( \Lambda + \sqrt{\frac{2s}{R_b}} \right) + \sqrt{\frac{2s}{R_b}} \sin\left( \Lambda + \sqrt{\frac{2s}{R_b}} \right) \right)$$
$$y = R_b \left( \sin\left( \Lambda + \sqrt{\frac{2s}{R_b}} \right) - \sqrt{\frac{2s}{R_b}} \cos\left( \Lambda + \sqrt{\frac{2s}{R_b}} \right) \right) \tag{2.13}$$
$$z = z_b$$
$$C = 0$$

$$x_{gen} = R_b \left( \cos(\Lambda_0) + \sqrt{\frac{2s}{R_b}} \sin(\Lambda_0) \right)$$
$$y_{gen} = R_b \left( \sin(\Lambda_0) - \sqrt{\frac{2s}{R_b}} \cos(\Lambda_0) \right) \tag{2.14}$$
$$z = z_b$$
$$C = \sqrt{\frac{2s}{R_b}}$$

Here, a coefficient $f$ is introduced, which is $+1$ for an involute curving in a counterclockwise direction from the base circle, or -1 for a clockwise curving involute, as shown in Figure 2.4. It is simply applied to the roll angle $\xi$ as follows:

$$x = R_b( \cos( f\xi + \Lambda) + f\xi \sin( f\xi + \Lambda) )$$
$$y = R_b( \sin( f\xi + \Lambda) - f\xi \cos( f\xi + \Lambda) ) \tag{2.15}$$
$$z = z_b$$
$$C = 0$$

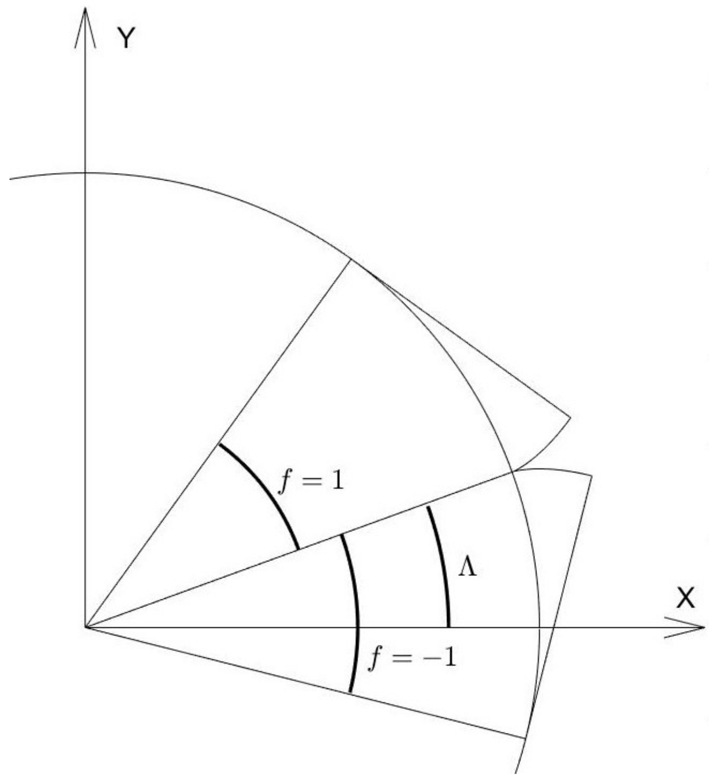


Figure 2.4: Affect of Involute with $f$ Included

If the gear is helical, then there is an offset that is a function of $z_b$ (axial distance from reference face) that must be applied.

$$\Lambda(z_b) = \Lambda_0 + \frac{z_b}{R_b} tan(\beta_b) \tag{2.16}$$

In Equation 2.16, $\Lambda_0$ is the constant offset value determined by tooth spacing, and $\beta_b$ is the helix angle at the base circle. This can be represented visually by unwrapping the cylinder (in this case the base cylinder) onto a plane. This is shown in Figure 2.5. If a helix is contained in the cylinder, then the cylinder is unwrapped, that helix becomes a straight line, rotated an angle $\beta_b$ from a vertical line, which is a projection of the center axis of the original cylinder. If the vertical line and the unwrapped helix meet at a point at the top, then the horizontal distance between the two lines is equal to $z_b tan(\beta_b)$. Now think about it before the cylinder was unwrapped. That same distance is the arc length on the cylinder between the two lines at that $z_b$ position, equal to $\Lambda_{step} R_b$. $\Lambda_{step}$ can then be solved for, and input into Equation 2.16. All of this is shown in Figure 2.5.

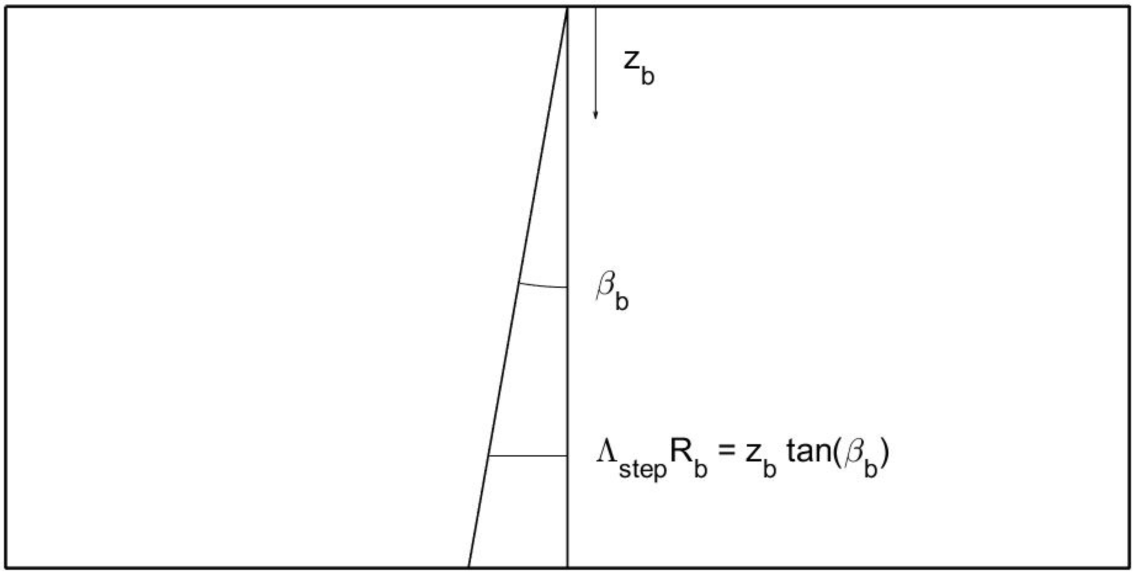


Figure 2.5: Base Cylinder Unwrapped Onto Tangent Plane

When a gear is defined according to various parameters detailed in [9], the helix angle at the pitch, or reference, radius $\beta$ is defined. The helix angle at the base circle is related to the helix angle at the reference radius by:

$$tan(\beta_b) = tan(\beta)cos(\alpha_t) \tag{2.17}$$

$$\beta_b = tan^{-1}(tan(\beta)cos(\alpha_t)) \tag{2.18}$$

## 2.3 Vectorial Representation

It is useful to think of points on the involute, as defined by these equations, as vectors beginning at the origin of the part coordinate system ($C = 0$), which is the center of the gear, as shown in Figure 2.6. At its most basic, the vectorial representation is as follows:

$$\mathbf{P} = \begin{bmatrix} R_b(\cos(\xi + \Lambda(z_b)) + \xi\sin(\xi + \Lambda(z_b))) \\ R_b(\sin(\xi + \Lambda(z_b)) - \xi\cos(\xi + \Lambda(z_b))) \\ z_b \end{bmatrix} \tag{2.19}$$

With the $f$ term and the helix angle offset included, the equation becomes:

$$\mathbf{P} = \begin{bmatrix} R_b(\cos(f\xi + \Lambda_0 + \frac{z_b}{R_b}tan(\beta_b)) + f\xi\sin(f\xi + \Lambda_0 + \frac{z_b}{R_b}tan(\beta_b))) \\ R_b(\sin(f\xi + \Lambda_0 + \frac{z_b}{R_b}tan(\beta_b)) - f\xi\cos(f\xi + \Lambda_0 + \frac{z_b}{R_b}tan(\beta_b))) \\ z_b \end{bmatrix} \quad (2.20)$$

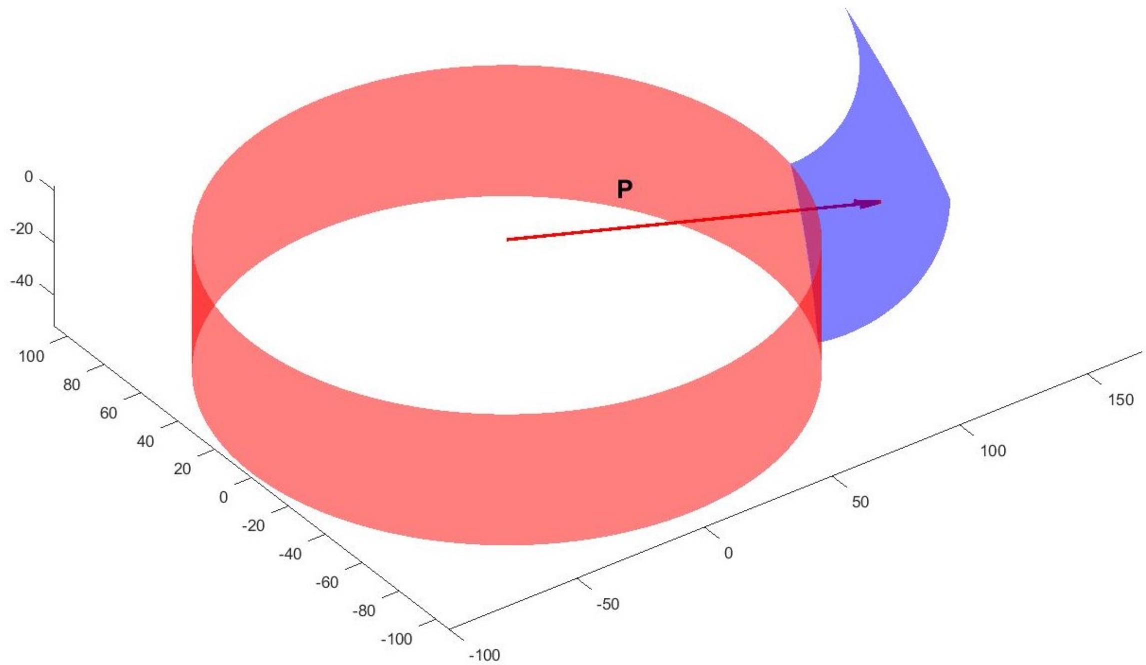


Figure 2.6: Point P on Helical Involute Flank

## 2.4 Surface Normal Vector

It is useful to know the vector normal to the involute surface. Here, note that the normal vector must lie in the plane tangent to the base circle. This knowledge gives the initial components in the XY (transverse) plane. Equation 2.21 gives the basic normal vector components of a pure involute in a transverse plane as a function of the roll angle $\xi$, and the initial start angle $\Lambda$.

$$\mathbf{n_{basic}} = \begin{bmatrix} sin(\xi + \Lambda) \\ -cos(\xi + \Lambda) \\ 0 \end{bmatrix} \quad (2.21)$$

Incorporating all the other relevant terms yields:

$$\mathbf{n_{transverse}} = \begin{bmatrix} \frac{z}{|z|} sin(f\xi + \Lambda) \\ \frac{-z}{|z|} cos(f\xi + \Lambda) \\ 0 \end{bmatrix} \tag{2.22}$$

Here it must be noted that according to [9], $z$ is the number of teeth, where an internal gear is denoted by a negative tooth number ($z$). This must be differentiated from the axial distance from the reference face, $z_b$.

Another consideration that must be made is regarding flank modifications. Flank modifications are defined as the areal change of flank geometry, described by local distances from the pure involute in the transverse plane [9]. Thus, even for helical gears, a normal vector in the transverse plane is useful, and can be determined by substituting Equation 2.16. The result is given in Equation 2.23.

$$\mathbf{n_{transverse}} = \begin{bmatrix} \frac{z}{|z|} sin(f\xi + \Lambda_0 + \frac{z_b}{R_b} tan(\beta_b)) \\ \frac{-z}{|z|} cos(f\xi + \Lambda_0 + \frac{z_b}{R_b} tan(\beta_b)) \\ 0 \end{bmatrix} \tag{2.23}$$

For a spur gear, Equation 2.22 is all that is needed to provide a true surface normal vector, because the Z component of the vector is clearly zero. For helical gears however, another term must be added. Recall Figure 2.5, and note that if a normal vector in the figure plane is applied to the inclined line, then the angle between that normal vector and horizontal is $\beta_b$. Thus, the X and Y components of the transverse normal vector as given in Equation 2.22 must be scaled by $\cos(\beta_b)$, and the Z component of the normal vector is then $\sin(\beta_b)$. Incorporating all the added terms to account for flanks with negative roll angles ($f$), helix angles ($\beta$), and internal gears ($\frac{z}{|z|} = -1$), the complete surface unit normal vector is as follows:

$$\mathbf{n} = \begin{bmatrix} \frac{z}{|z|} sin(f\xi + \Lambda_0 + \frac{z_b}{R_b} tan(\beta_b)) cos(f\beta_b) \\ \frac{-z}{|z|} cos(f\xi + \Lambda_0 + \frac{z_b}{R_b} tan(\beta_b)) cos(f\beta_b) \\ \frac{z}{|z|} sin(\beta_b) \end{bmatrix} \tag{2.24}$$

In [28], Guenther presented the following expression for the surface normal vector.

$$\mathbf{n_G}(\mathbf{P}) = \begin{bmatrix} +\sin(\xi+\Lambda)/|\mathbf{n_{Evol}}| \\ -\cos(\xi+\Lambda)/|\mathbf{n_{Evol}}| \\ R_b \cdot \tan(\beta)/\left(R_p|\mathbf{n_{Evol}}|\right) \end{bmatrix} \tag{2.25}$$

$$|\mathbf{n_G}| = \sqrt{1 + (R_b \cdot \tan(\beta)/R_p)^2} \tag{2.26}$$

In these equations, $\xi$ is the roll angle, $\Lambda$ is the initial offset angle, $\beta$ is the helix angle at the reference radius, and $R_p$ is the reference (pitch) radius.

The derivation for that expression is significantly different than what is given above. In [28], the base equations of the involute profile, $\mathbf{P}$, are differentiated with respect to roll angle $\xi$ and axial position $z_b$. The cross product of these two partial derivatives is taken, and the result is a vector mutually orthogonal to the two tangent vectors in the profile and axial direction, $\frac{\partial P_s}{\partial \xi}$, and $\frac{\partial P_s}{\partial z_b}$ respectively. The vector is then divided by its own magnitude to create a unit normal vector. The method is known to work, but it must now be shown that Equations 2.24 and 2.25 are equivalent. Specifically, it must be shown that the following relation is valid:

$$\sqrt{1 + \left(\frac{R_b}{R_p}\tan(\beta)\right)^2} = \frac{1}{\cos(\beta_b)} \tag{2.27}$$

From [9], it is known that

$$\tan(\beta_b) = \tan(\beta)\cos(\alpha_t) \tag{2.28}$$

where

$$\cos(\alpha_t) = \frac{R_b}{R_p} \tag{2.29}$$

Thus

$$\tan(\beta_b) = \tan(\beta)\frac{R_b}{R_p} \tag{2.30}$$

Equation 2.30 can be directly substituted into Equation 2.26.

$$|\mathbf{n_G}| = \sqrt{1 + (\tan(\beta_b))^2} \tag{2.31}$$

It is well known that,

$$\tan(\theta) = \frac{\sin(\theta)}{\cos(\theta)} \tag{2.32}$$

Thus

$$\begin{aligned} |\mathbf{n_G}| &= \sqrt{\frac{\cos^2(\beta_b)}{\cos^2(\beta_b)} + \frac{\sin^2(\beta_b)}{\cos^2(\beta_b)}} \\ &= \sqrt{\frac{\cos^2(\beta_b) + \sin^2(\beta_b)}{\cos^2(\beta_b)}} \\ &= \sqrt{\frac{1}{\cos^2(\beta_b)}} \\ &= \frac{1}{\cos(\beta_b)} \end{aligned} \tag{2.33}$$

The two expressions are therefore equivalent.

Note that Equations 2.21 and 2.24 are unit normal vectors with a total length of 1. For offsetting by the tool radius, stock removal, etc., the total distance from the nominal surface is calculated, and multiplied by $\mathbf{n}$.

$$\mathbf{N} = d\mathbf{n} \tag{2.34}$$

It must also be noted that this is the normal vector as relates to the pure involute only. Both the pure (Equation 2.24) and transverse (Equation 2.22) surface normal vectors are shown in Figure 2.7.

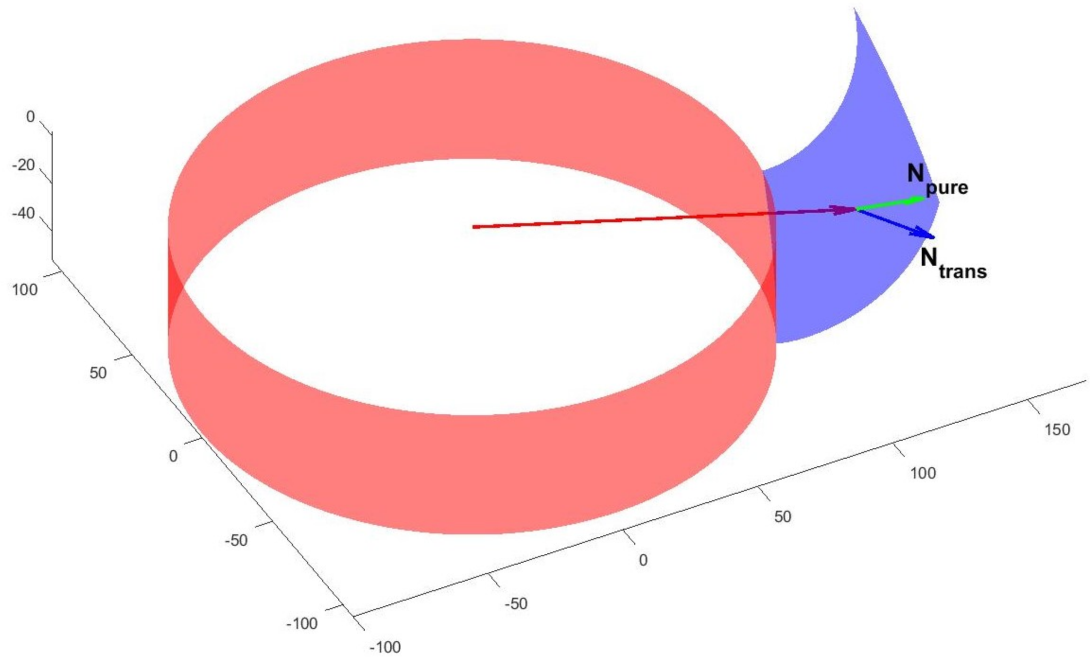


Figure 2.7: Pure and Transverse Normal Vectors on a Helical Involute Surface

## 2.5 Commanded Tool Position and Orientation

To get an actual commanded tool center point position $\mathbf{Q}$, the following equation is used to generate a point in the part coordinate system, represented vectorially by $\mathbf{Q}$:

$$\mathbf{Q} = \mathbf{P} + \mathbf{N} = \mathbf{P} + d\mathbf{n} \tag{2.35}$$

Vector $\mathbf{Q}$ is shown below in Figure 2.8.

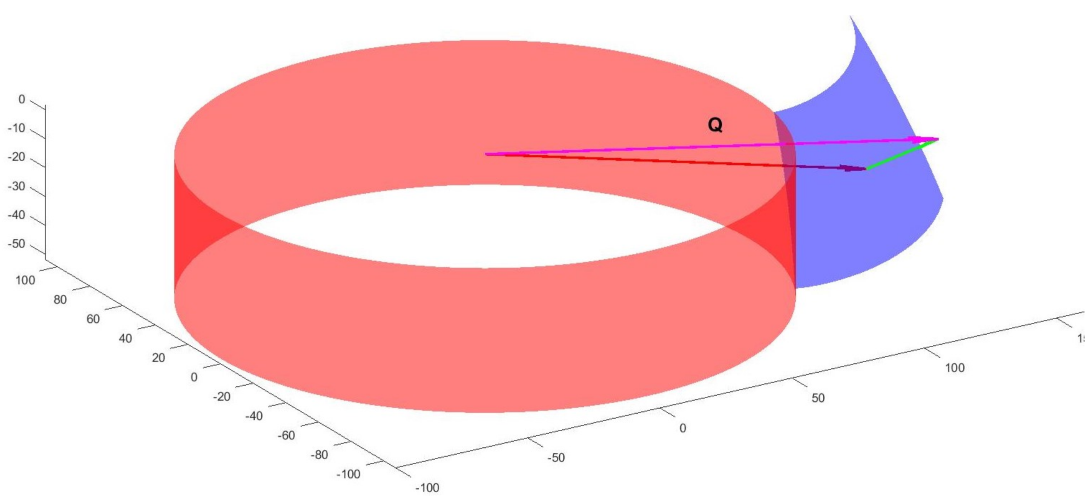


Figure 2.8: Tool Center Point Position on a Helical Involute Surface

## 2.6 The Tool Orientation Vector

For five axis motions, a tool centerpoint coordinate is not enough. A tool orientation in the part coordinate system must also be defined.

Here another property of the involute profile can be used. Note that if two involutes on the same base circle are separated by some offset angle $\Lambda$, then the normal difference between them at every point is $R_b\Lambda$, as shown in Figure 2.9.

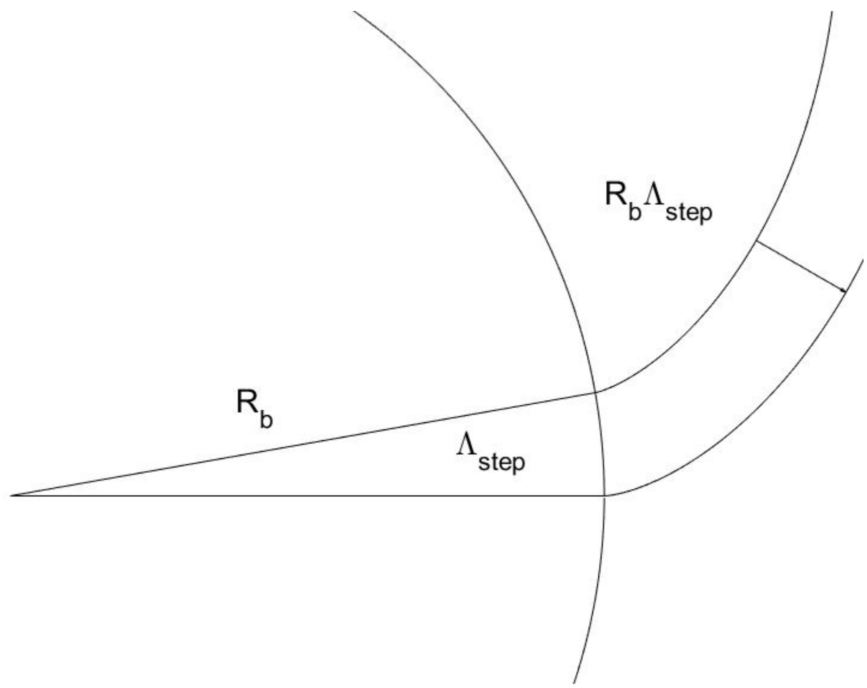


Figure 2.9: Two Parallel Involutes, Separated by Offset Angle $\Lambda_{step}$

Recall Equation 2.16. The equation is clearly linear with $z_b$, the coefficient being $\frac{tan(\beta_b)}{R_b}$. Thus, at every $z_b$ position, there is a corresponding offset angle $\Lambda_{z_b}$, giving a constant offset distance of $R_b\Lambda_{z_b}$, which is linear with $z_b$. Thus, the intersection between a helical involute flank and a plane tangent to the base circle will be a straight line, rotated an angle $\beta_b$ from a projection of the center axis of the gear onto the plane. This is shown in Figure 2.10.

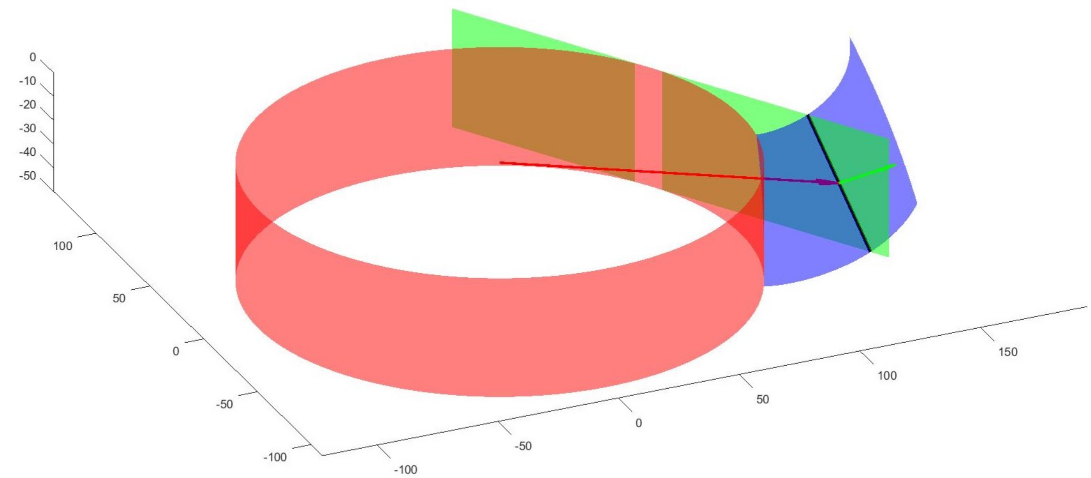

Figure 2.10: Tangent Plane Intersection with Helical Involute Surface

It is this line with which the tool, a cylindrical endmill, must be aligned. Thus, a vectorial representation of this intersection line can be used to define the tool orientation, as shown in Figure 2.11. Mathematically, this is represented by Equation 2.36.

$$\mathbf{T} = \begin{bmatrix} -sin(f\xi + \Lambda_0 + \frac{z_b}{R_b}tan(\beta_b))sin(\beta_b) \\ cos(f\xi + \Lambda_0 + \frac{z_b}{R_b}tan(\beta_b))sin(\beta_b) \\ cos(\beta_b) \end{bmatrix} \tag{2.36}$$

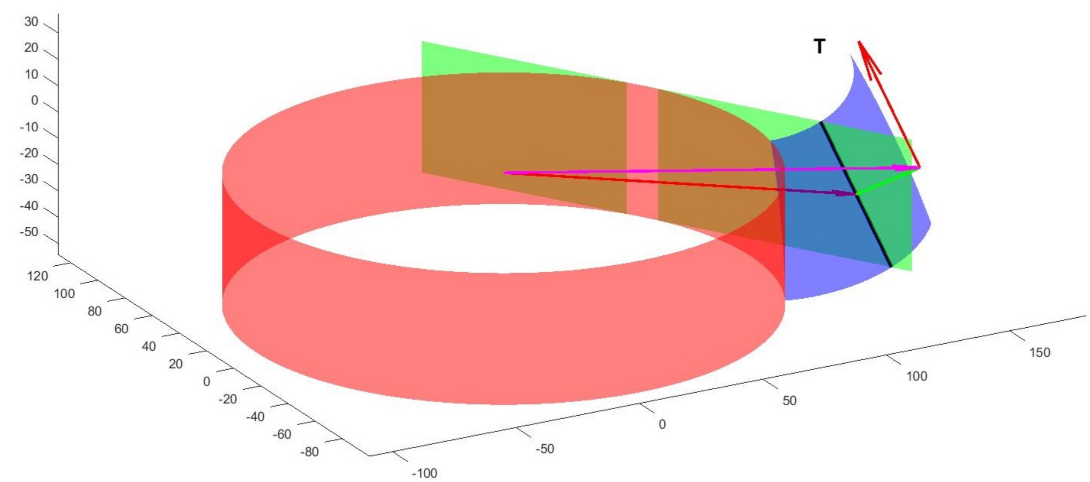


Figure 2.11: Tool Orientation Vector on Helical Involute Surface

For spur gears, the tool is to be oriented vertically, as shown in Figure 2.12, and indeed, if $\beta_b$ is taken to be zero, Equation 2.36 reduces:

$$\mathbf{T_{spur}} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.37}$$

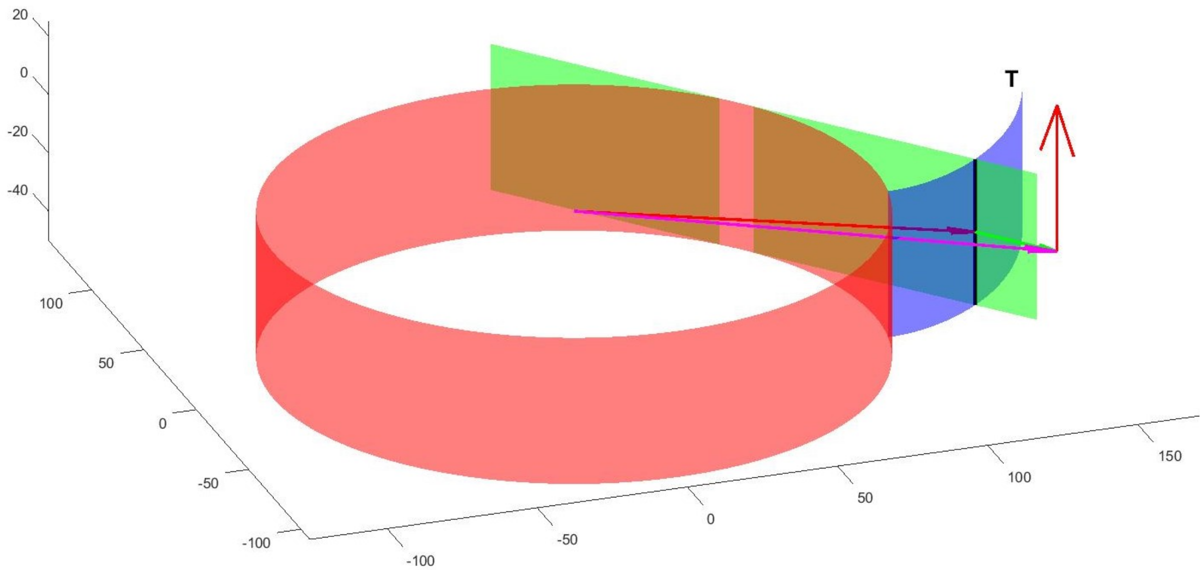


Figure 2.12: Tool Orientation Vector on Spur Involute Surface

## 2.7 Pure Involute Tool Travel Direction Vector

One more vector that is interesting but not directly used is the direction vector of the tool through the material, $\mathbf{V}$. In part coordinates, the travel direction takes the form:

$$\mathbf{V} = \begin{bmatrix} dir * cos(f\xi + \Lambda_0 + \frac{z_b}{R_b} tan(\beta_b)) \\ dir * sin(f\xi + \Lambda_0 + \frac{z_b}{R_b} tan(\beta_b)) \\ 0 \end{bmatrix} \tag{2.38}$$

where $dir$ is 1 if the tool is traveling from the base circle outward, and -1 if the tool is traveling toward the base circle.

$\mathbf{V}$ is not used directly, but it is interesting to note that for a pure involute, $\mathbf{n}$, $\mathbf{T}$, and **V** are mutually orthogonal, forming a local coordinate system at each point.

# CHAPTER 3: IMPLEMENTATION

There are several considerations to be made when actually using the information in the previous chapter to plan the toolpaths for cutting a gear.

First, there are three different "dimensions" that must be considered for a particular gear. These are distance from nominal surface (call it stock thickness, for use in roughing passes and finish pass), axial position $z_b$ (cannot realistically cut full depth all at once), and tooth number. For speed, these have been prioritized as follows:

a) Axial Depth

b) Stock Thickness

c) Tooth Number

For example, starting at tooth (or rather, space) number 1, at the farthest distance from the nominal surface, it is desired to cut at successive $z_b$ positions all the way down to the full face width. When this is complete, it steps in to the next closest distance from the nominal surface, in steps of $RI \times d_{tool}$ (where $RI$ is the tool radial immersion, and $d_{tool}$ is the tool diameter), and repeats the previous axial steps down to full face width. When it has completely cut down to the final geometry for that space, it moves to space number 2, and repeats the previous process.

## 3.1 Point Generation

The basic gear parameters to be input are number of teeth $z$ (negative if internal), transverse module $m_t$, transverse pressure angle $\alpha_t$, helix angle $\beta$, and facewidth $b$. All pertinent parameters are then calculated according to [9]. A simple diagram of the cross section of a gear is shown in Figure 3.1.

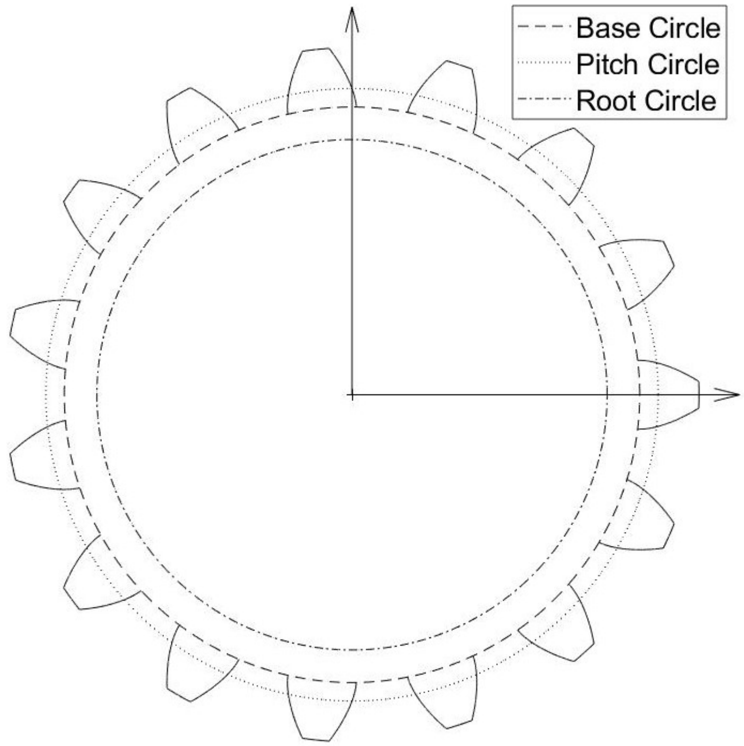


Figure 3.1: Basic Gear Geometry

Two of the most important parameters generated by these calculations are the nominal starting (small radius) and ending (large radius) roll angles, $\xi$.

The problem can be approached by looking at the three priorities given at the beginning of this chapter, and working from the bottom up.

### 3.1.1 Ordering

For a conventional down milling cut in an external gear, the tool needs to be traveling from the tip to the root of a right handed flank, and from the root to the tip on a left handed flank, as shown in Figure 3.2.

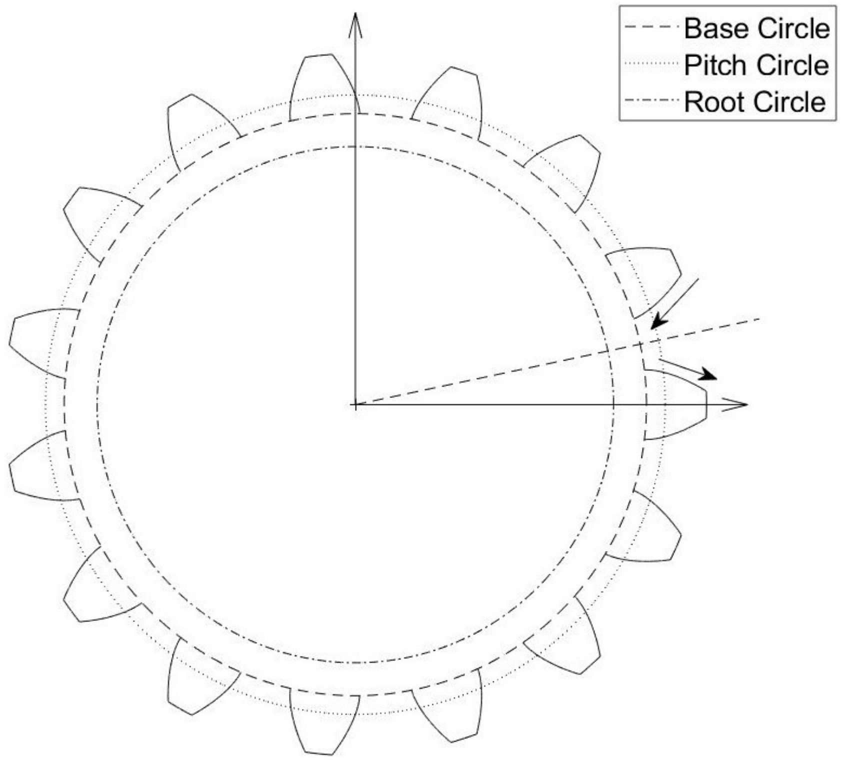


Figure 3.2: Cutting Direction in an External Gear

For an internal gear, also downmilling, technically the same terms apply according to the terminology defined in [9], but the tool needs to be on the opposite side of the involute, traveling in the opposite direction, as shown in Figure 3.3.

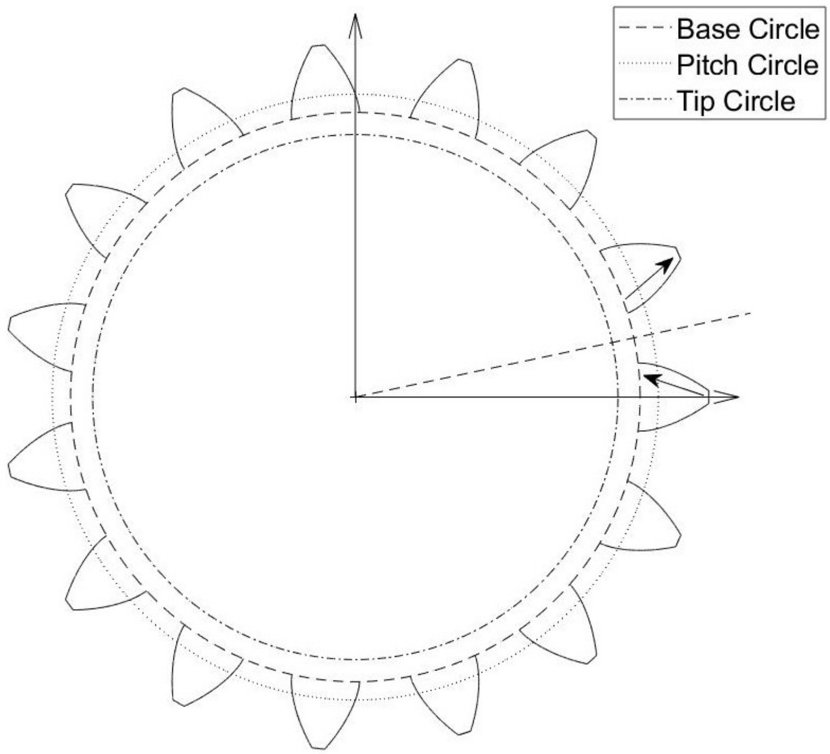


Figure 3.3: Cutting Direction in an Internal Gear

In order to realize this, a supplemental term $f_{cut}$ is defined. $f_{cut}$ is used to determine which direction to add certain offsets (see Section 3.1.2). See Section 2.2 for an explanation of $f$.

$$f_{cut} = \frac{z}{|z|} f \tag{3.1}$$

### 3.1.2 Offsets

The assumption is made that a tooth (for external gears) or a space (for internal gears) is always centered on the positive X axis, and a starting offset angle $\Lambda_N$ is defined as being centered on the tooth space (external) or tooth (internal) in the positive angular direction, which is $\frac{1}{2}$ the angular tooth pitch offset from the tooth centered on the X axis. The counter $k$ corresponds to priority $c$), tooth number. This is shown in Figure 3.4 for the first offset angle $\Lambda_{N,1}$.

$$\Lambda_{N,k} = (k-1)\frac{2\pi}{|z|} + \frac{\pi}{|z|} \tag{3.2}$$

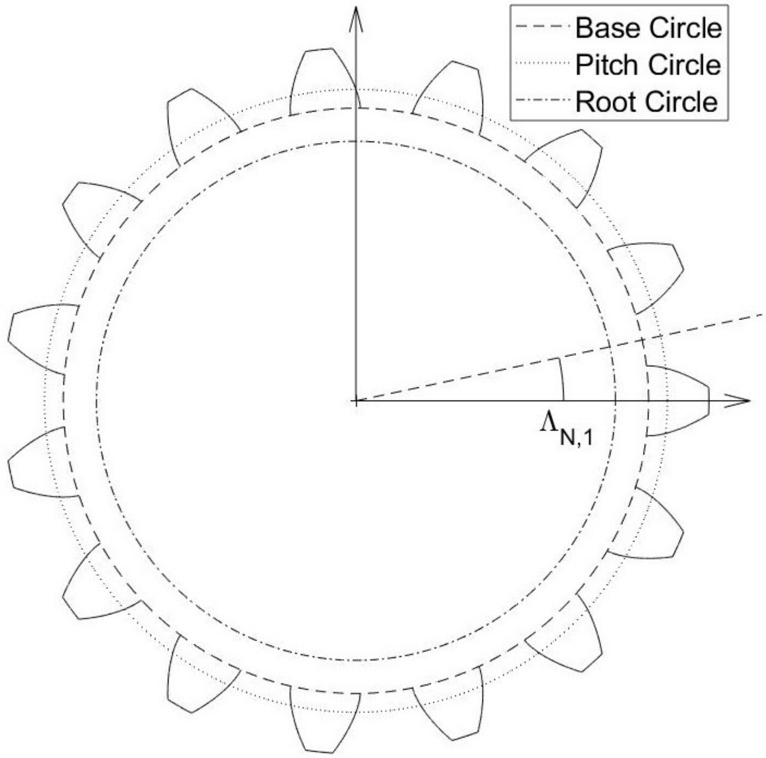


Figure 3.4: Basic Gear Geometry with Offset $\Lambda_N$

Another offset must be added, to get the actual start position of the involute flank. This offset, denoted by $\Lambda_S$, is different for internal and external gears.

$$\Lambda_S = \begin{cases} f_{cut}\eta_b, External Gears \\ f_{cut}\psi_b, Internal Gears \end{cases} \tag{3.3}$$

where $\eta_b$ and $\psi_b$ are the tooth space half width angle, and tooth thickness half width angle respectively, according to [9]. $f_{cut}$ is as defined in Section 3.1.1. This offset is shown in Figure 3.5.

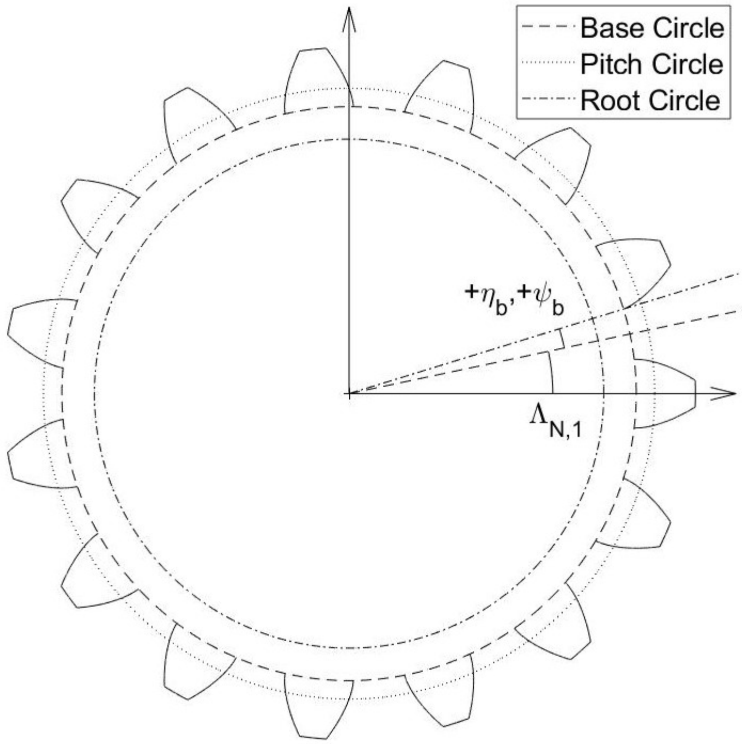


Figure 3.5: Basic Gear Geometry with Offset $\eta_b$, or $\psi_b$

A third offset angle is added, which is simply the offset angle due to any helix angle present, and is simply given according to Equation 2.16.

$$\Lambda_\beta = \frac{z_{b,i}}{R_b} tan(\beta_b) \tag{3.4}$$

Note that this is a function of Z height $z_b$, and thus must be calculated at every Z position, and then added to the final offset which determines the start point of the involute. The counter $i$ corresponds to priority $a$), axial depth.

Finally, these offsets are summed to determine the appropriate offset angle for a particular involute flank.

$$\Lambda_{Total} = \Lambda_N + \Lambda_S + \Lambda_\beta \tag{3.5}$$

For external gears, the resulting equation is:

$$\Lambda_{Total} = (k-1)\frac{2\pi}{|z|} + \frac{\pi}{|z|} + f_{cut}\eta_b + \frac{z_{b,i}}{R_b} tan(\beta_b) \tag{3.6}$$

For internal gears, the resulting equation is:

$$\Lambda_{Total} = (k-1)\frac{2\pi}{|z|} + \frac{\pi}{|z|} + f_{cut}\psi_b + \frac{z_{b,i}}{R_b}tan(\beta_b) \tag{3.7}$$

### 3.1.3 Vector Generation

Once the offsets described in Section 3.1.2 are known parametrically, they can be used to define involutes at specific locations. Recall the beginning of this chapter, where three priorities are given, namely axial depth, stock thickness, and tooth number. If a list of possible values for each is given, in a particular order, then those parameters can be used to define all the vectors described in Chapter 2.

Axial depth must go from 0 (the top face), to $-b$ (the face width in the downward direction, corresponding to -Z), in steps defined by the user, say $z_{b,step}$.

$$z_b = [0 : z_{b,step} : -b] \tag{3.8}$$

The stock thickness $d$ must go from some maximum value $d_{MaxStock}$ (described and defined in Section 3.2), to a user defined minimum value (final stock clearance for the finishing operation), $d_{finalstock}$ in steps of $-RI \times d_{tool}$, where $RI$ is the desired radial immersion of the tool, and $d_{tool}$ is the tool diameter.

$$d = [d_{MaxStock} : -RI \times d_{tool} : d_{finalstock}]; \tag{3.9}$$

The tooth number must obviously go from 1 to $|z|$. At each of these points, the roll angle must go from some minimum $\xi_{inner}$ to some maximum $\xi_{maximum}$. The starting and ending roll angles are calculated at the beginning, but these are subsequently modified according to the following limits, as described in Section 3.3. The roll angle can be iterated by even increments of arc length $s$, or roll angle $\xi$. Note that the involute equations in Chapter 2 are given in terms of roll angle $\xi$. Thus, if it is desired to increment by arc length, the beginning and ending roll angles are converted to arc lengths, the list of values is generated with even increments, and then all values are converted back to their respective roll angles. Those conversions are performed by means of Equations 2.11 and 2.12.

Thus far, all involutes have been considered going from the base circle outward. For those involutes desired to be traveling inward, the resulting list of roll angle points along that involute is reversed, so that the first value is at the outermost radius, and the final value is close to the base circle.

The pseudo code looks something like the following:

```
for Loop through tooth number (k)
        for Loop through Stock Thickness (j)
                for Loop through axial depth (i)
                        Approach
                        Involute 1
                                Starting Point to Ending Point
                        Transition
                        Involute 2
                                Starting Point to Ending Point
                        Retract
                end Axial Depth
        end Stock Thickness
end Tooth Number
```

## 3.2 Maximum Stock Thickness

For both external and internal gears, it is desired to determine a maximum stock thickness that can be cut. For external gears, a line is defined extending radially from the center of the gear, offset from the base point ($\xi = 0$) in the direction of the tooth space (away from the flank) by an angle $\gamma$ (dashed line in Figure 3.6). The maximum stock thickness is the distance from the point on that radial line at the tip radius, $R_a$, to the nominal flank in the direction normal to the flank. Starting with the knowledge that the offset is in the normal direction to the involute flank, it can be said that the maximum offset distance is going to be some function of a theoretical offset angle $\Lambda_{MaxStock}$. Note that the subscript $a$ denotes those quantities relating to the tooth tip.

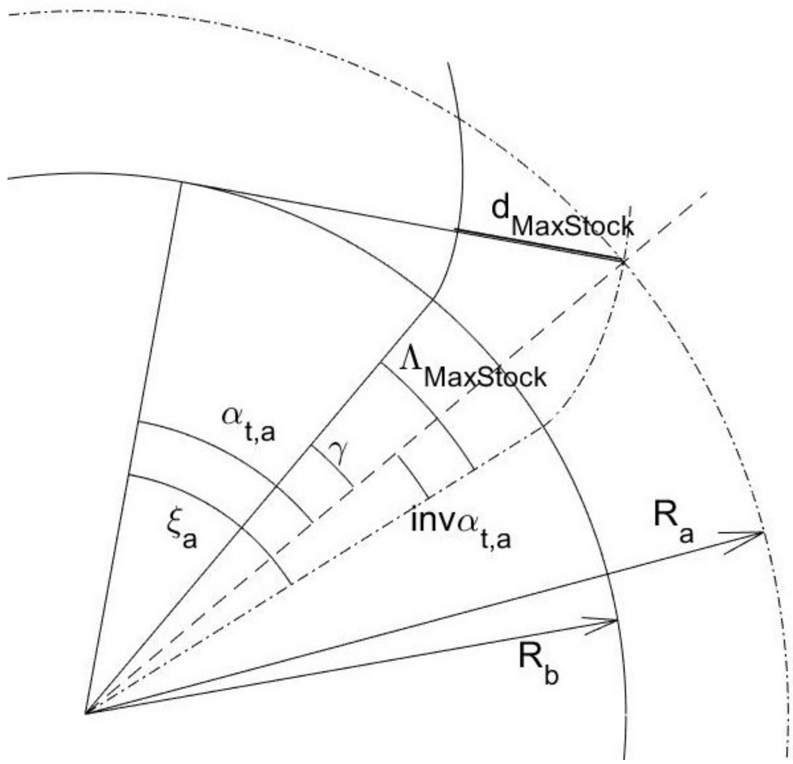


Figure 3.6: Maximum Stock Thickness

$$d_{MaxStock} = R_b \Lambda_{MaxStock} \tag{3.10}$$

It can be seen in Figure 3.6 that the following equations are valid.

$$\Lambda_{MaxStock} = inv\alpha_{t,a} + \gamma \tag{3.11}$$

$$\alpha_{t,a} = cos^{-1}\left(\frac{R_b}{R_a}\right) \tag{3.12}$$

Equation 18 from [9] gives the following:

$$inv\alpha_t = \xi - \alpha_t = tan(\alpha_t) - \alpha_t \tag{3.13}$$

Substitution yields:

$$\Lambda_{MaxStock} = tan(\alpha_{t,a}) - \alpha_{t,a} + \gamma \tag{3.14}$$

$$\Lambda_{MaxStock} = tan\left(cos^{-1}\left(\frac{R_b}{R_a}\right)\right) - cos^{-1}\left(\frac{R_b}{R_a}\right) + \gamma \tag{3.15}$$

$$d_{MaxStock,Basic} = R_b\left(tan\left(cos^{-1}\left(\frac{R_b}{R_a}\right)\right) - cos^{-1}\left(\frac{R_b}{R_a}\right) + \gamma\right) \tag{3.16}$$

For actual toolpath generation, the tool radius must be added to this, as follows:

$$d_{MaxStock,WithTool} = R_b\left(tan\left(cos^{-1}\left(\frac{R_b}{R_a}\right)\right) - cos^{-1}\left(\frac{R_b}{R_a}\right) + \gamma\right) + r_{tool} \tag{3.17}$$

For internal gears, recall that the tool center point must not cross the tooth space center line, for safety reasons (avoiding cutting into the adjacent flank). Thus the maximum stock thickness is simply:

$$d_{maxstock} = R_b\eta_b - r_{tool} \tag{3.18}$$

From [9], $\eta_b$ is defined as the tooth space half width angle.

## 3.3 Roll Angle Limits

There are certain limits which must be applied to the roll angle. These are given in the sections below.

### 3.3.1 Tooth Gap Center Line

In both external and internal gears, the involute is to be cut using successive passes. The passes must be offset from the nominal geometry by the thickness of stock to be left after the pass, as well as the tool radius, since the commanded coordinates are the tool centerpoint coordinates. As a result, the roll angle must be limited in both cases such that in the preliminary cutting passes, the tool does not cut into the adjacent flank. Sections 3.3.1.1 and 3.3.1.2 provide the derivations for limiting the roll angle to the tooth space centerline for both external and internal gears.

An approach similar to that presented in Section 3.2 may be used.

#### 3.3.1.1 External Gears

In external gears, the normal vector and the tooth space centerline are both on the convex side of the involute curve, as shown in Figure 3.7. Thus, any limiting roll angle will be on the end closer to the base circle.

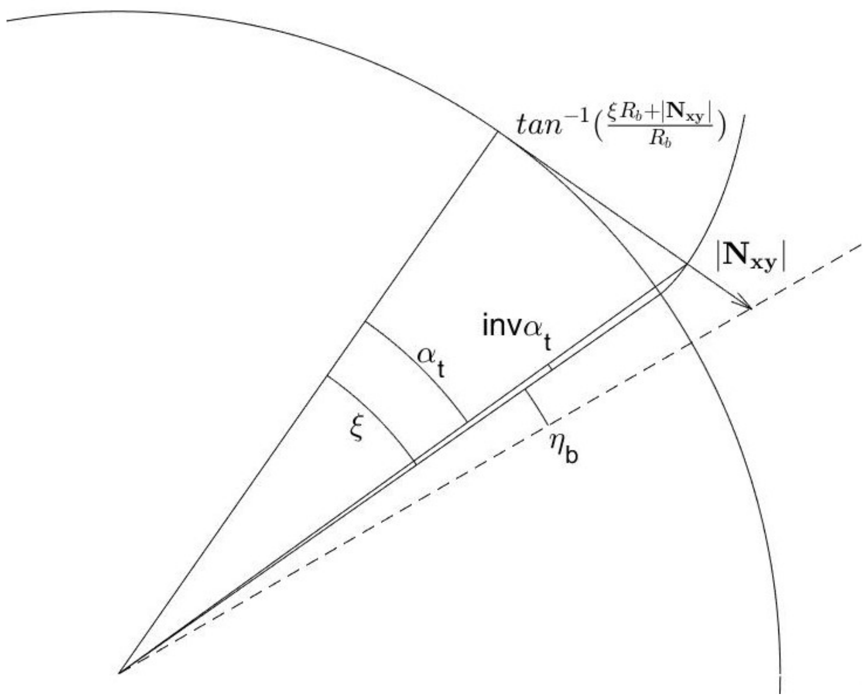


Figure 3.7: Roll Angle Limit For External Gear

$$\xi + \eta_b = tan^{-1}\left(\frac{R_b\xi + |\mathbf{N_{xy}}|}{R_b}\right) \tag{3.19}$$

$$\xi + \eta_b = tan^{-1}\left(\xi + \frac{|\mathbf{N_{xy}}|}{R_b}\right) \tag{3.20}$$

$$tan(\xi + \eta_b) = \xi + \frac{|\mathbf{N_{xy}}|}{R_b} \tag{3.21}$$

#### 3.3.1.2 Internal Gears

For internal gears, the normal vector and the tooth space centerline are both on the concave side of the involute curve, shown in Figure 3.8.

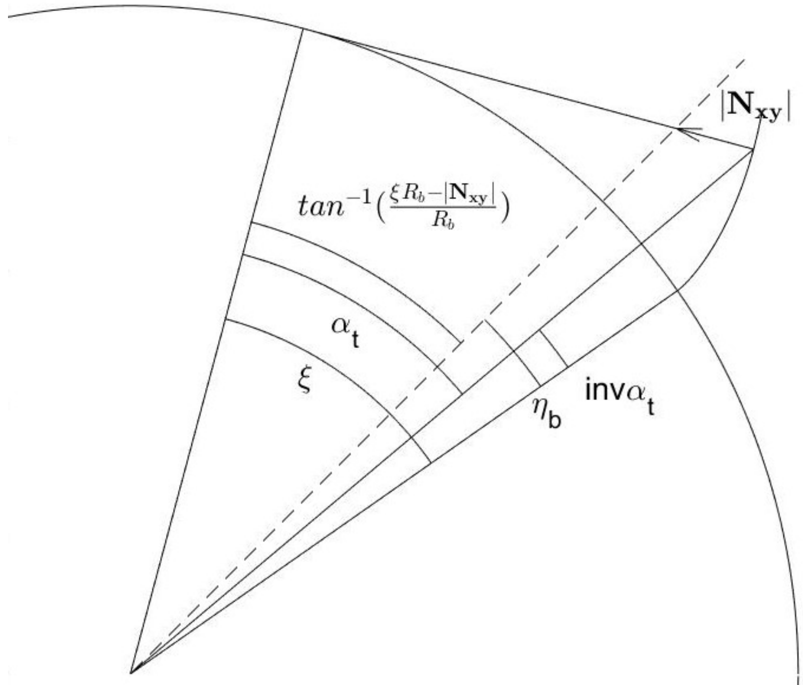


Figure 3.8: Roll Angle Limit For Internal Gear

$$\xi = \eta_b + tan^{-1}\left(\frac{R_b\xi - |\mathbf{N_{xy}}|}{R_b}\right) \tag{3.22}$$

$$\xi - \eta_b = tan^{-1}\left(\frac{R_b\xi - |\mathbf{N_{xy}}|}{R_b}\right) \tag{3.23}$$

$$\xi - \eta_b = tan^{-1}\left(\xi - \frac{|\mathbf{N_{xy}}|}{R_b}\right) \tag{3.24}$$

$$tan(\xi - \eta_b) = \xi - \frac{|\mathbf{N_{xy}}|}{R_b} \tag{3.25}$$

#### 3.3.1.3 A Comprehensive Solution

From Sections 3.3.1.1 and 3.3.1.2, it can be seen that if the two equations are compared, then the following comprehensive solution is clear:

$$tan(\xi + \frac{z}{|z|}\eta_b) = \xi + \frac{z}{|z|}\frac{|\mathbf{N_{xy}}|}{R_b} \tag{3.26}$$

where $z$ is the number of teeth, $\xi$ is the roll angle in radians, $\eta_b$ is the tooth space half width angle at the base circle, $|\mathbf{N_{xy}}|$ is the normal distance from the nominal involute in the XY (transverse) plane, and $R_b$ is the base radius. $|\mathbf{N_{xy}}|$ is given by the following equation, where $th$ is the total stock thickness to be left after the pass, $d_{tool}$ is the tool diameter, and $\beta_b$ is the helix angle at the base circle.

$$|\mathbf{N_{xy}}| = th + \frac{d_{tool}}{2cos(\beta_b)} \tag{3.27}$$

Equation 3.26 must be numerically solved for $\xi$ in order to obtain the actual limiting value. For external gears, this is a starting roll angle (the point closest to the base circle). For an internal gear, this is an ending roll angle (the point farthest from the base circle).

In both cases, a situation can arise where no limit is found. This can be represented graphically. If the left side of Equation 3.26 is denoted by $A$, and the right side denoted by $B$,

$$A = tan(\xi + \frac{z}{|z|}\eta_b) \tag{3.28a}$$

$$B = \xi + \frac{z}{|z|}\frac{N_{xy}}{R_b} \tag{3.28b}$$

then the two can be plotted on the same log-log plot, both versus roll angle $\xi$. The roll angle limit is the point at which the two curves intersect, shown in the top panel of Figure 3.9. It can be easier to visualize if the inverse tangent of both sides is taken, and the equations plotted again on linear axes.

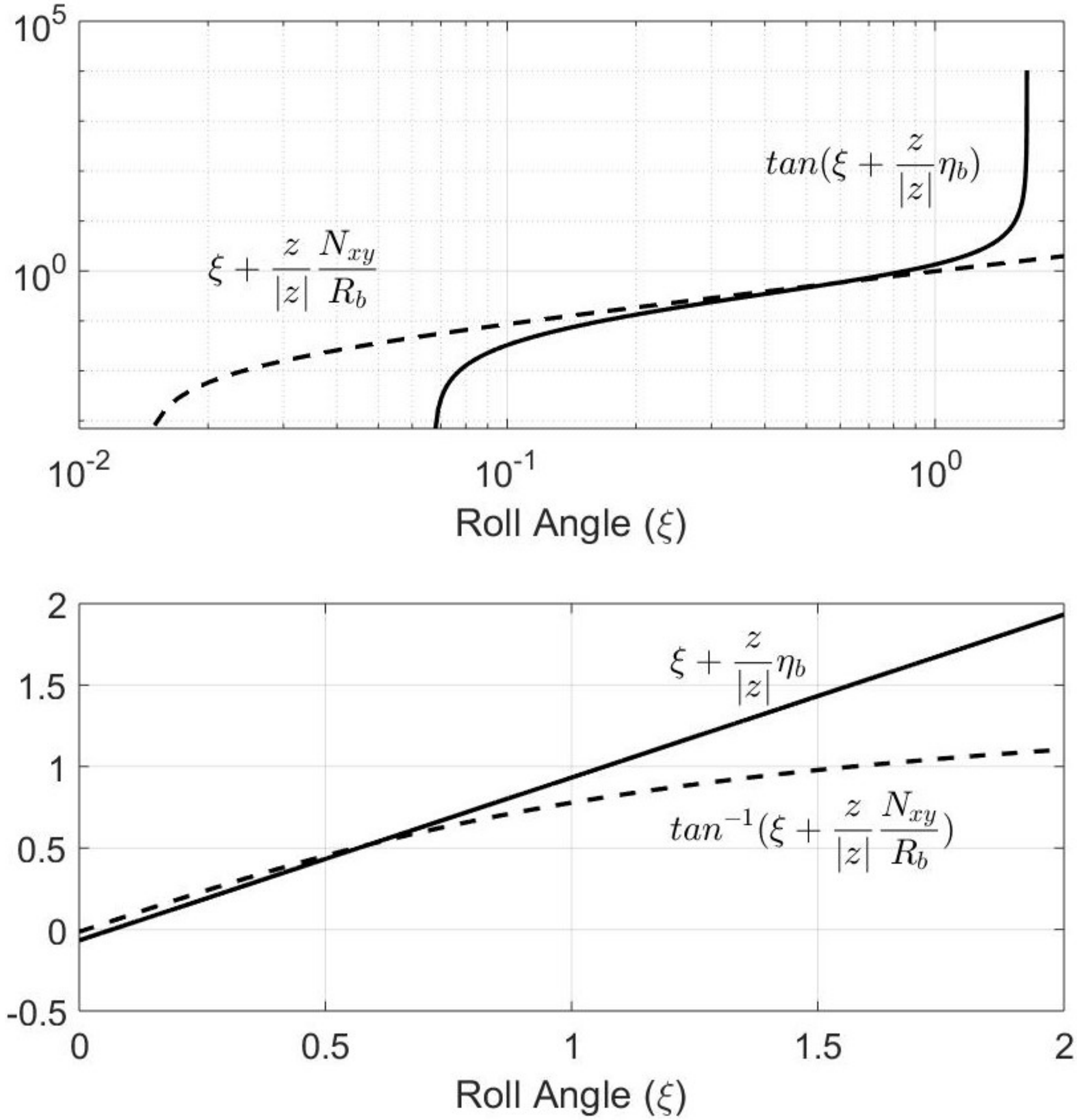


Figure 3.9: Roll Angle Limit with Intersecting Lines

If there are no cases where the limit can be reached, then the two lines will not intersect, as shown in Figure 3.10.

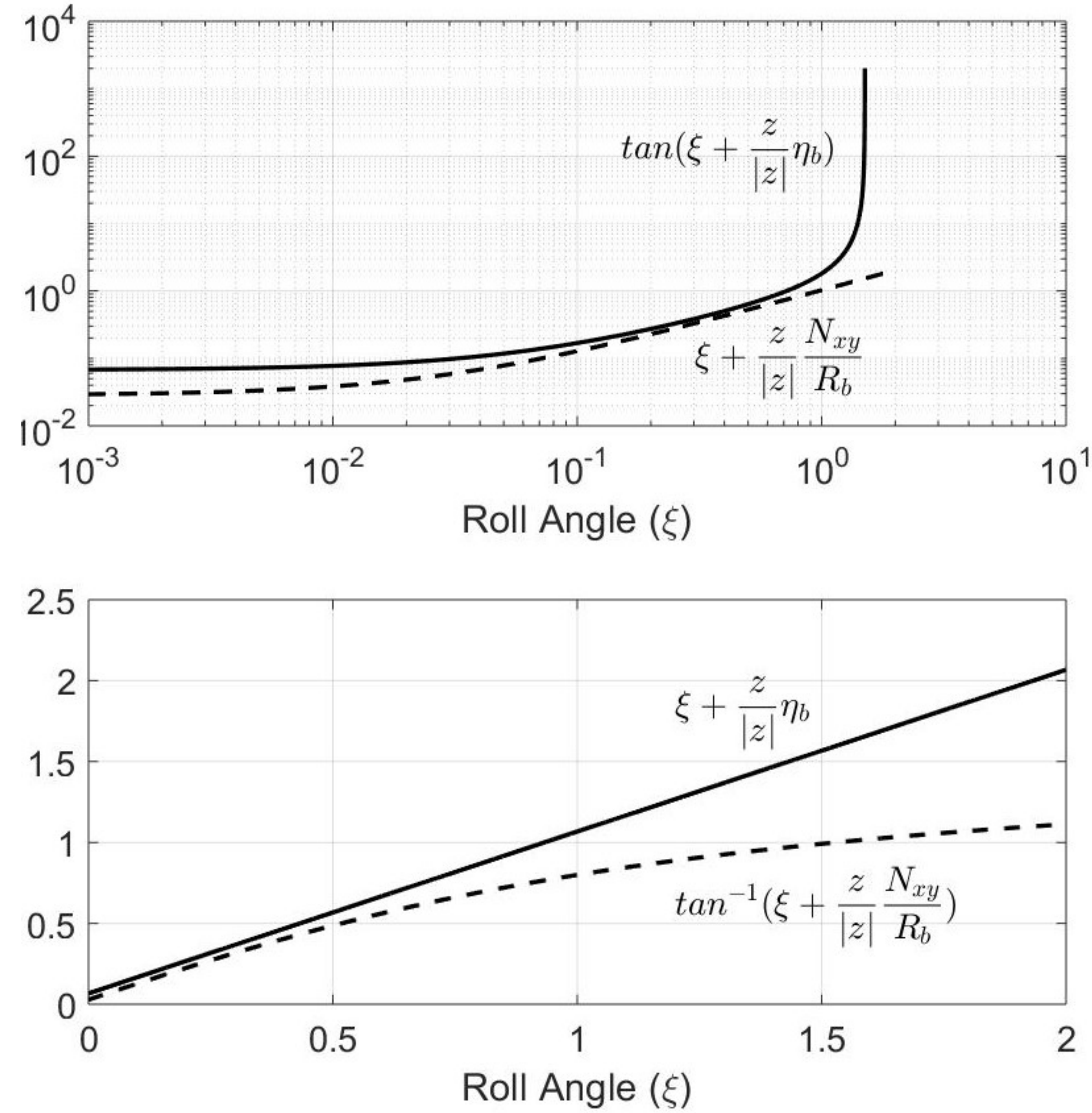


Figure 3.10: Roll Angle Limit with Non-Intersecting Lines

For external gears, a valid limiting roll angle (an intersection point of the two lines in Figure 3.9) can be found only if the normal distance $\mathbf{N_{xy}}$ satisfies the following condition:

For external gears, the limiting roll angle will be the minimum roll angle, e.g. closer to the base circle. Thus, if a limit cannot be found directly at the base circle, where $\xi = 0$, then there is no limit. The cut can begin right at the base circle. To determine if this is the case, simply substitue 0 into Equation 3.26, as shown below. Note that this only applies to external gears, so $\frac{z}{|z|} = 1$, and drops out.

$$tan(0 + \eta_b) = 0 + \frac{|\mathbf{N_{xy}}|}{R_b} \tag{3.29}$$

$$tan(\eta_b) = \frac{|\mathbf{N_{xy}}|}{R_b} \tag{3.30}$$

$$|\mathbf{N_{xy}}| \geq R_b tan(\eta_b) \tag{3.31}$$

In the case of an internal gear however, the limiting roll angle is the ending roll angle, or the farthest point from the base circle. In this case, if Equation 3.32 is not met, then a valid limiting roll angle cannot be found. This means that the cut cannot be completed at all, as even a roll angle of zero is beyond the limit. $\mathbf{N_{xy}}$ must be made smaller, or $\eta_b$ made larger.

$$|\mathbf{N_{xy}}| \leq R_b tan(\eta_b) \tag{3.32}$$

### 3.3.2 Internal Gear, Helix Limit on Inaccessible Flank

In internal helical gears, it is impossible to entirely cut one flank on each tooth, due to geometrical constraints. The intersection line between the tangent plane and the helical involute surface will reach either the top or the bottom of the flank first. Recall that this line is the line with which the tool is aligned. If this line (essentially the tool) reaches the bottom first, there is no problem. But if it reaches the top first, in an internal gear, if it were to keep going, the tool would cut into the adjacent flank. This line is shown in Figure 3.11, below.

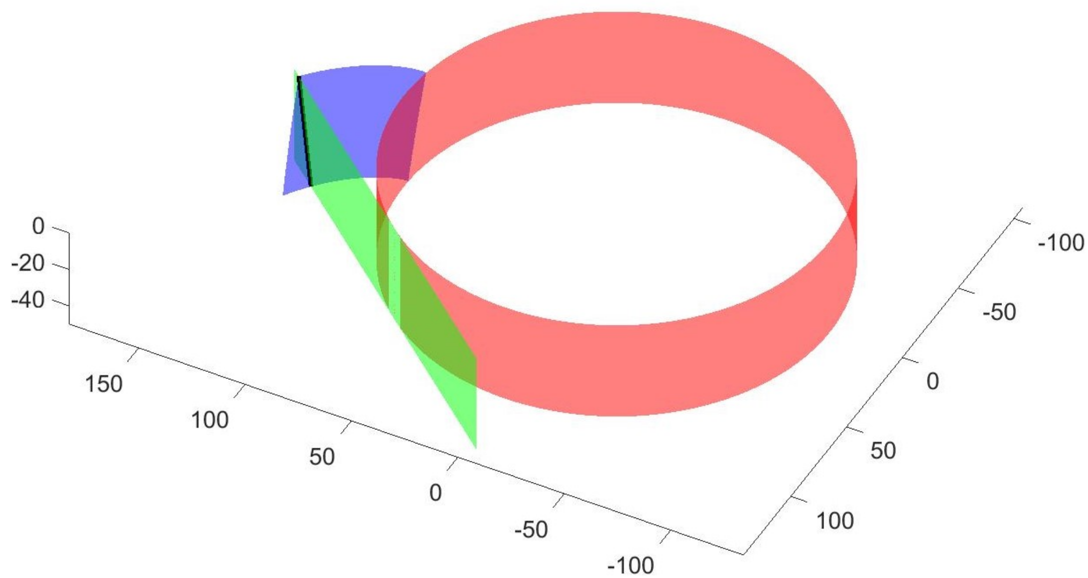


Figure 3.11: Roll Angle Limit of an Internal Helical Gear

If $\xi_a$ is the nominal final roll angle (farthest from the base circle), then the limiting roll angle is given by:

$$\xi_{f,lim} = \xi_f + \frac{z_b}{R_b} tan(\beta_b) \tag{3.33}$$

where $z_b$ is the axial distance from the zero point (the top face of the gear), and $\beta_b$ and $R_b$ are the helix angle at the base circle and base radius, as always.

It should be noted that $\xi_a$ is the roll angle **after** being limited by Equation 3.26. Also note that this is only to be used for the ending roll angle (outermost point) in the inaccessible flank for internal gears.

The inaccessible flank is the flank "facing upward". This can be represented mathematically by

$$\beta > 0 \ \& \ f = -1 \tag{3.34a}$$

$$\beta < 0 \ \& \ f = 1 \tag{3.34b}$$

Note here that a positive helix angle, $\beta > 0$, corresponds to a right handed helix (like a conventional screw), and a negative helix angle, $\beta < 0$, corresponds to a left handed helix. This holds true for both internal and external gears.

### 3.3.3 Tip Roll Angle for External Gears

For external gears only, the roll angle at the tip must be extended by a small amount. The tool will come off the tip of the tooth into free space with no danger of interference, and insures that the tooth is cut completely. For spur gears, this is not critical if the tip radius (as previously cut) is accurate. For helical gears however, it is critical, particularly for the flanks for which one of the following is true:

$$\beta > 0 \ \& \ f = 1 \tag{3.35a}$$

$$\beta < 0 \ \& \ f = -1 \tag{3.35b}$$

In these cases, the minimum roll angle that must be added is as follows, where $z_{step}$ is the axial step size. Not that this is a minimum value, and a greater roll angle addition is acceptable.

$$\xi_{add} = \frac{z_{step}}{R_b} tan(\beta_b) \tag{3.36}$$

## 3.4 Coordinate Transformations

The vectors above are represented in part coordinates. In other words, they all assume that the rotary axes, C and B, are at the zero position. This can be represented by a matrix **R**, which includes the C and B positions.

$$\mathbf{R} = \begin{bmatrix} C \\ B \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.37}$$

In Figures 3.12 and 3.13 are shown original un-transformed tool path data for the finishing pass of an external helical gear with a 10 degree helix angle. The red vectors are the tool orientation vectors **T** detailed in Section 2.6, and the green vectors are the surface normal vectors **N** detailed in Section 2.4.

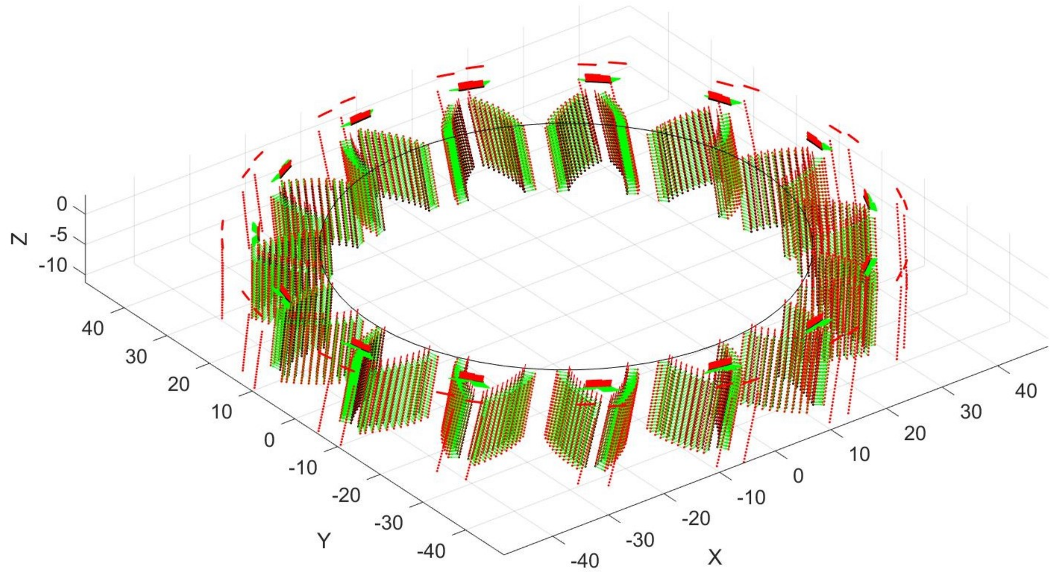


Figure 3.12: Original Tool Paths in Part Coordinates

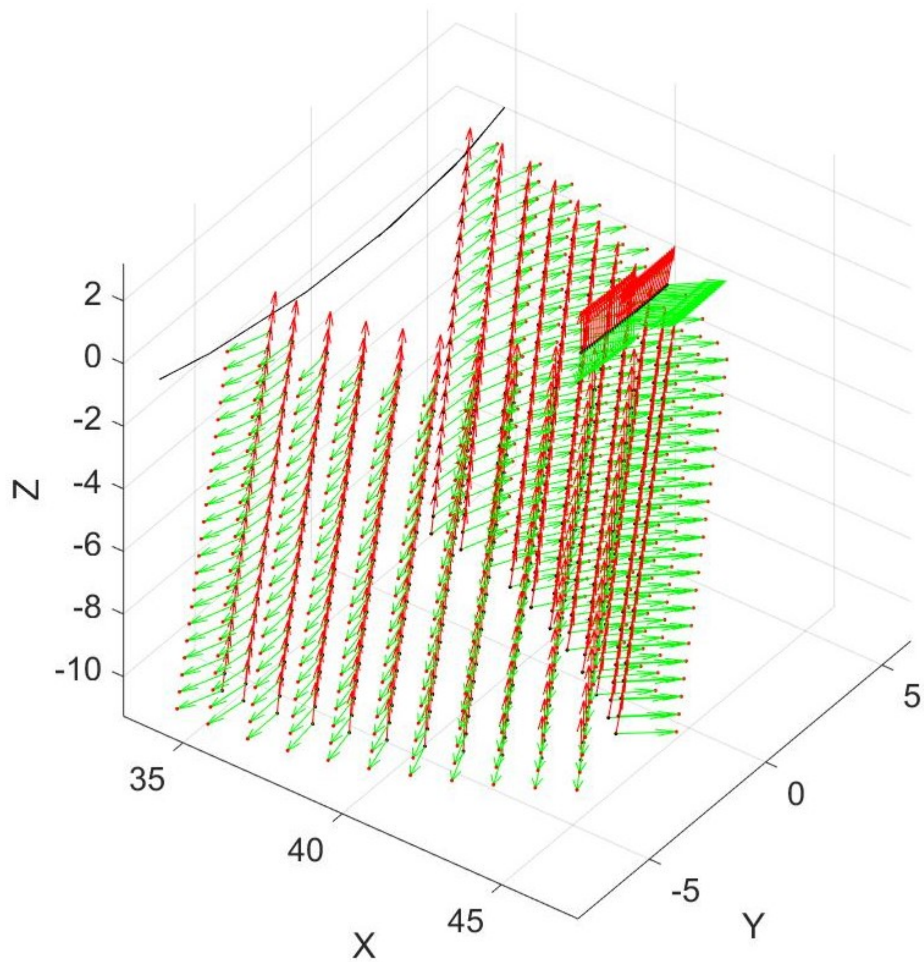


Figure 3.13: Original Tool Paths in Part Coordinates, Zoomed in on One Tooth

Coordinate transformations must be performed such that the toolpaths satisfy two main requirements: First, the tool must be oriented vertically in the machine space. This is a general five-axis constraint that must be satisfied for any part to be made in this particular machine. Second, the tool centerpoint must lie in a plane tangent to the base circle. This is a constraint unique to involute gears, and is not required for other parts to be made.

In the case of helical gears, due to the nature of the involute profile, if the first requirement above is satisfied, then the second requirement is automatically satisfied as well. On spur gears however, the tool vector is already oriented vertically in part coordinates, and more information is needed.

### 3.4.1 Orienting the C Axis

The first step is to rotate the points such that **T** lies in the XZ plane. To do this, the C rotation angle is be calculated by:

$$Rot_C = atan\left(\frac{\mathbf{T_y}}{\mathbf{T_x}}\right) \tag{3.38}$$

For spur gears however, **T** is already in the XZ plane. In this case, **n** (or **N**) is used.

$$Rot_C = atan\left(\frac{\mathbf{n_y}}{\mathbf{n_x}}\right) \tag{3.39}$$

For left hand flanks ($f = -1$), $\pi$ is added to $Rot_C$. This puts two opposing flanks toolpaths in line with each other when converted to generation motion.

If $\beta_b < 0$ (a left hand helix), then $\pi$ can be added to $Rot_C$, in order to keep the tool on the front side of the machine for visibility. This is not critical, but may be useful for observation purposes.

$Rot_C$ is actually the C coordinate as commanded in degrees for every point. A positive rotation of the machine C axis is clockwise. So if the C axis is commanded to go to $Rot_C$ in the positive direction (clockwise), then that point and all associated vectors in part coordinates must be rotated clockwise about the centerpoint of the table, which is a negative rotation, considered numerically.

To perform this coordinate rotation, the appropriate HTM can be generated using a standard rotation matrix inserted into a 4x4 identity matrix.

$$\mathbf{H_1} = \begin{bmatrix} cos(-Rot_C) & -sin(-Rot_C) & 0 & 0 \\ sin(-Rot_C) & cos(-Rot_C) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cos(Rot_C) & sin(Rot_C) & 0 & 0 \\ -sin(Rot_C) & cos(Rot_C) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.40}$$

This HTM is then applied to the **P**, **Q**, **T**, and **n** vectors. This aligns all the toolpaths in a plane tangent to the base circle.

$$\mathbf{P_{n1}} = \mathbf{H_1} * \mathbf{P} \tag{3.41a}$$

$$\mathbf{Q_{n1}} = \mathbf{H_1} * \mathbf{Q} \tag{3.41b}$$

$$\mathbf{T_{n1}} = \mathbf{H_1} * \mathbf{T} \tag{3.41c}$$

$$\mathbf{n_{n1}} = \mathbf{H_1} * \mathbf{n} \tag{3.41d}$$

Figures 3.14 and 3.15 show the same tool paths from Figures 3.12 and 3.13, rotated about the C axis for generation motion. Notice that the red **T** vectors are not vertical (aligned with the Z axis), nor are the green **N** vectors horizontal. The next step is to rotate the B axis to orient the **T** vectors.

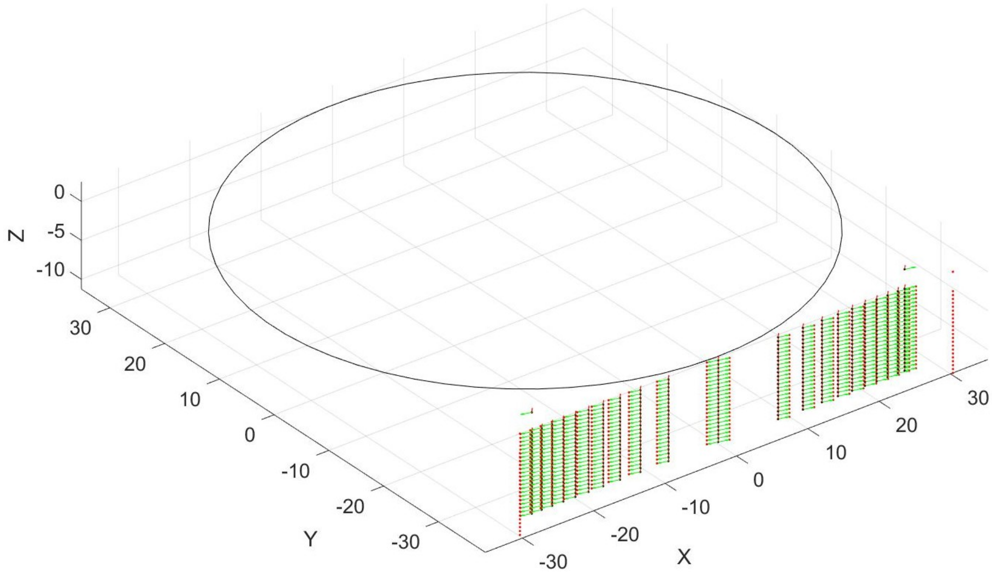


Figure 3.14: Tool Paths Rotated About C Axis

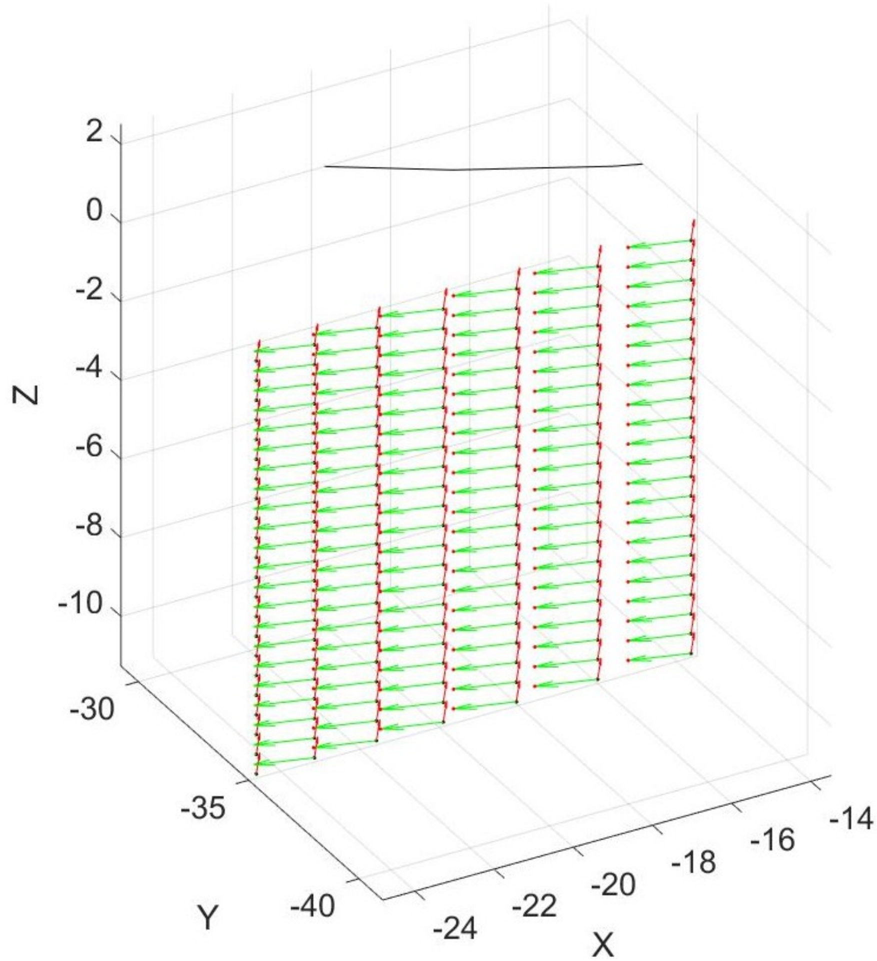


Figure 3.15: Tool Paths Rotated About C Axis, Zoomed In on a Small Section

### 3.4.2 Orienting the B Axis

At this point, the toolpaths lie in a plane tangent to the base circle, satisfying the generation motion criteria. The points are still, however, not rotated about Y (a B rotation) to account for any helix angle. In other words, the tool orientation vector is not yet vertical in the machine coordinate system. Unlike the C rotation, the rotation zero point is no longer at the part coordinate system zero point. To account for this, a vector $L$ is generated, which is the position of the part coordinate system in relation to the center of rotation of the B axis.

$$\mathbf{L} = \begin{bmatrix} BcentX - WCS_x \\ CcentY - WCS_y \\ BcentZ - WCS_z \end{bmatrix} = \begin{bmatrix} L_x \\ 0 \\ L_z \end{bmatrix} \tag{3.42}$$

First, the part coordinate system is translated by $-\mathbf{L}$ to the center of rotation of the B axis.

$$\mathbf{H_2} = \begin{bmatrix} 1 & 0 & 0 & -L_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.43}$$

Then the rotation about B is applied, using $Rot_B$, calculated similarly to the C rotation above.

$$Rot_B = atan\left(\frac{\mathbf{T_x}}{\mathbf{T_z}}\right) \tag{3.44}$$

$$\mathbf{H_3} = \begin{bmatrix} cos(-Rot_B) & 0 & sin(-Rot_B) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(-Rot_B) & 0 & cos(-Rot_B) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cos(Rot_B) & 0 & -sin(Rot_B) & 0 \\ 0 & 1 & 0 & 0 \\ sin(Rot_B) & 0 & cos(Rot_B) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.45}$$

In this transformation, **L** is also rotated. After this, the part coordinate system is again translated back to its original, although translated and rotated, position.

$$\mathbf{H_4} = \begin{bmatrix} 1 & 0 & 0 & L_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.46}$$

The final result is as follows:

$$\mathbf{P_{n2}} = \mathbf{H_4} * \mathbf{H_3} * \mathbf{H_2} * \mathbf{H_1} * \mathbf{P} \tag{3.47a}$$

$$\mathbf{Q_{n2}} = \mathbf{H_4} * \mathbf{H_3} * \mathbf{H_2} * \mathbf{H_1} * \mathbf{Q} \tag{3.47b}$$

$$\mathbf{T_{n2}} = \mathbf{H_4} * \mathbf{H_3} * \mathbf{H_2} * \mathbf{H_1} * \mathbf{T} \tag{3.47c}$$

$$\mathbf{n_{n2}} = \mathbf{H_4} * \mathbf{H_3} * \mathbf{H_2} * \mathbf{H_1} * \mathbf{n} \tag{3.47d}$$

Figures 3.16 and 3.17 show the same tool paths as before rotated about the B axis to account for the helix angle. Note the red **T** vectors are now aligned with the Z axis.

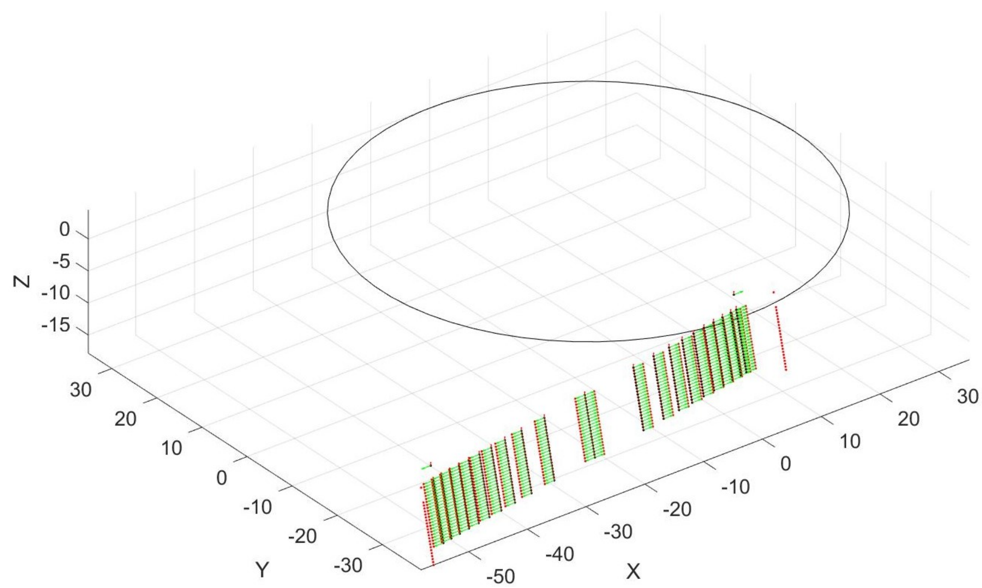


Figure 3.16: Tool Paths Rotated About C and B Axes

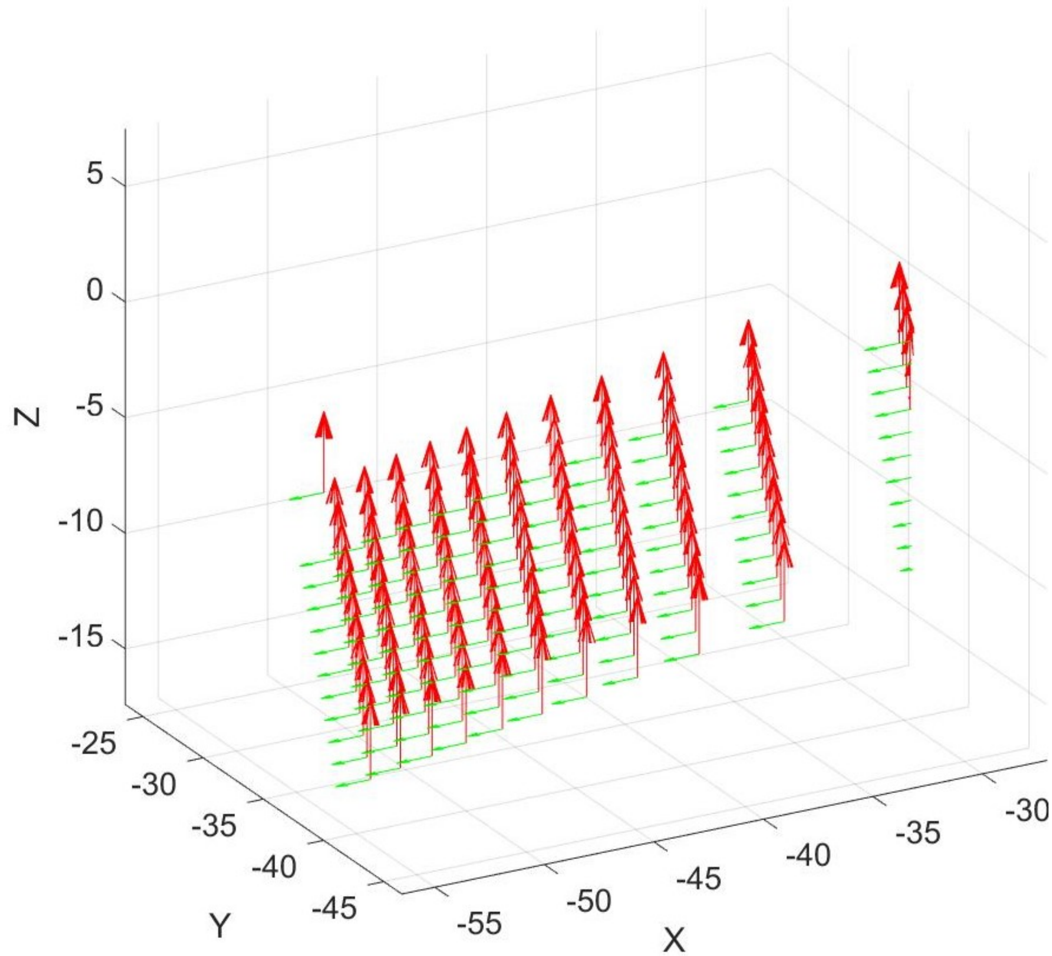


Figure 3.17: Tool Paths Rotated About C and B Axes, Zoomed In on a Small Section, Tool Vectors Enlarged to Show Detail

The tool point XYZ coordinates in the machine coordinate system, with work offsets applied, are stored in $\mathbf{Q_{n2}}$, and the B and C axes positions, $Rot_B$ and $Rot_C$, are stored in a 2 by i matrix $\mathbf{R}$, where i is the number of data points.

$$\mathbf{R} = \begin{bmatrix} Rot_C \\ Rot_B \end{bmatrix} \tag{3.48}$$

# CHAPTER 4: FEED VELOCITY

## 4.1 Basic Theory

In basic generation principle motion, the rotational speed of the gear and the linear speed of the tool, within the global, or machine, coordinate system, are constant. The speed of the tool through the material, however, is not constant. In fact, the feed velocity of the tool through the material, $\mathbf{V_{t,g}}$, is the vector difference of the tool velocity relative to the machine, $\mathbf{V_{t,g}}$, and gear velocity relative to the machine, $\mathbf{V_{g,m}}$, at the contact point, as shown in Equation 4.1, and Figure 4.1.

$$\mathbf{V_{t,g}} = \mathbf{V_{t,m}} - \mathbf{V_{g,m}} \tag{4.1}$$

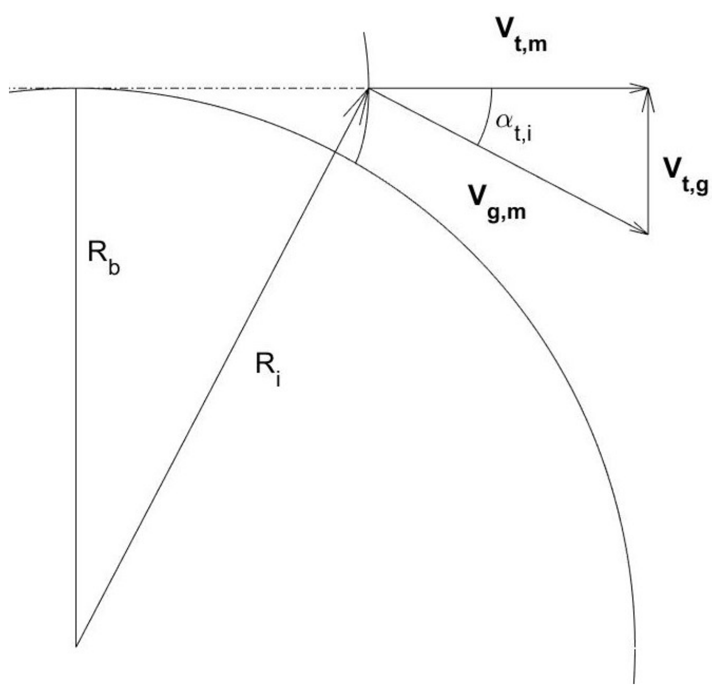


Figure 4.1: Feed Velocity Vectorial Representation.

From Equation 4.1, it is clear that

$$V_{t,g,x} = V_{t,m,x} - V_{g,m,x}$$
$$V_{t,g,y} = V_{t,m,y} - V_{g,m,y} \tag{4.2}$$

It is also clear that

$$|\mathbf{V_{g,m}}| = \omega R_i \tag{4.3}$$

where $R_i$ is the radius at the contact point, or the distance from the cutting point to the center of rotation. From Figure 4.1, it can been seen that

$$\mathbf{V_{g,m}} = \begin{bmatrix} |V_{g,m}|\cos(\,-\alpha_{t,i}) \\ |V_{g,m}|\sin(\,-\alpha_{t,i}) \end{bmatrix} = \begin{bmatrix} \omega R_i \cos(\,\alpha_{t,i}) \\ -\omega R_i \sin(\,\alpha_{t,i}) \end{bmatrix} \tag{4.4}$$

The tool is traveling directly along the machine X axis, at a constant rate of $\omega R_b$, thus

$$\mathbf{V_{t,m}} = \begin{bmatrix} \omega R_b \\ 0 \end{bmatrix} \tag{4.5}$$

Thus, substitution of Equations 4.4 and 4.5 into Equation 4.2 gives

$$\mathbf{V_{t,g}} = \begin{bmatrix} \omega R_b - \omega R_i \cos(\,\alpha_{t,i}) \\ \omega R_i \sin(\,\alpha_{t,i}) \end{bmatrix} \tag{4.6}$$

The goal is to control the actual feed velocity at a particular point $i$, i.e. the magnitude of $\mathbf{V_{t,g,i}}$, which is

$$|\mathbf{V_{t,g,i}}| = \sqrt{V_{t,g,i,x}^2 + V_{t,g,i,y}^2} \tag{4.7}$$

$$|\mathbf{V_{t,g,i}}| = \sqrt{(\,\omega R_b - \omega R_i \cos(\,\alpha_{t,i})\,)^{\,2} + (\,\omega R_i \sin(\,\alpha_{t,i})\,)^{\,2}} \tag{4.8}$$

The angle $\alpha_{t,i}$ is

$$\alpha_{t,i} = \tan^{-1}\left(\frac{R_b \xi_i}{R_b}\right) \tag{4.9}$$

$$\alpha_{t,i} = \tan^{-1}(\,\xi_i)\,. \tag{4.10}$$

Thus, substituting into Equation 4.8,

$$|\mathbf{V_{t,g,i}}| = \sqrt{(\,\omega R_b - \omega R_i \cos(\,\tan^{-1}(\,\xi_i)\,)\,)^{\,2} + (\,\omega R_i \sin(\,\tan^{-1}(\,\xi_i)\,)\,)^{\,2}}. \tag{4.11}$$

The radius from the center point of rotation to the point of contact $R_i$ can be determined by recognizing that a right triangle, opposite the involute, is generated, the two sides of which are $R_b$ and $\xi_i R_b$ respectively, and the hypotenuse is $R_i$.

$$R_i = \sqrt{(\,R_b)^{\,2} + (\,\xi_i R_b)^{\,2}} \tag{4.12}$$

$$R_i = \sqrt{R_b^2(\,\xi_i^2 + 1)} \tag{4.13}$$

$$R_i = R_b\sqrt{\xi_i^2 + 1} \tag{4.14}$$

As an aside, if Equation 4.14 is solved for $\xi_i$, the following is obtained:

$$\xi_i = \sqrt{\left(\frac{R_i}{R_b}\right)^2 - 1} \tag{4.15}$$

Substituting Equation 4.14 into Equation 4.11, the result is

$$|\mathbf{V_{t,g,i}}| = \sqrt{\left(\omega R_b - \omega R_b\sqrt{\xi_i^2+1}\cos(\tan^{-1}(\xi_i))\right)^2 + \left(\omega R_b\sqrt{\xi_i^2+1}\sin(\tan^{-1}(\xi_i))\right)^2}. \tag{4.16}$$

$$|\mathbf{V_{t,g,i}}| = \sqrt{\omega^2 R_b^2\left(\left(1 - \sqrt{\xi_i^2+1}\cos(\tan^{-1}(\xi_i))\right)^2 + \left(\sqrt{\xi_i^2+1}\sin(\tan^{-1}(\xi_i))\right)^2\right)}. \tag{4.17}$$

$$|\mathbf{V_{t,g,i}}| = \omega R_b \sqrt{\begin{array}{l}\left(1 - \sqrt{\xi_i^2+1}\cos(\tan^{-1}(\xi_i))\right)^2 \\ \quad + \left(\sqrt{\xi_i^2+1}\sin(\tan^{-1}(\xi_i))\right)^2\end{array}} \tag{4.18}$$

$$|\mathbf{V_{t,g,i}}| = \omega R_b \sqrt{\begin{array}{r} 1 \\ -2\sqrt{\xi_i^2+1}\cos(\tan^{-1}(\xi_i)) \\ +(\xi_i^2+1)\cos^2(\tan^{-1}(\xi_i)) \\ +(\xi_i^2+1)\sin^2(\tan^{-1}(\xi_i)) \end{array}} \tag{4.19}$$

$$|\mathbf{V_{t,g,i}}| = \omega R_b \sqrt{2 - 2\sqrt{\xi_i^2+1}\cos(\tan^{-1}(\xi_i)) + \xi_i^2}. \tag{4.20}$$

Equation 4.20 is an expression relating the feed velocity of the tool through the material to the rotary speed of the workpiece in the machine coordinate system and previously decided gear parameters. $|\mathbf{V_{t,g,i}}|$ is what must be chosen as the desired linear feedrate, $F_{IPM}$. Thus, Equation 4.20 becomes:

$$F_{IPM} = \omega R_b \sqrt{2 - 2\sqrt{\xi_i^2+1}\cos(\tan^{-1}(\xi_i)) + \xi_i^2}. \tag{4.21}$$

## 4.2 Inverse Time Feedrates

It now becomes necessary to explain the use of inverse time feedrates in the application of milling gears by the generation principle.

### 4.2.1 Motivation

There are three ways of specifying a feedrate for a CNC machine tool, specified by G93, G94, and G95 modal commands.

- G93 - Inverse time feedrate, in units of 1/min
- G94 - Conventional mm/min or in/min feedrate
- G95 - Feed per spindle revolution, such as mm/rev or in/rev

The first, G93 or Inverse time feedrate, is rarely used in practice, and only when generated by a suitable CAM software. The value to be input is dependent on the length of the particular motion, thus a feedrate value is required in every single block while the machine is in G93 mode.

The second, commanded by G94, is the most commonly used type of feedrate. It specifies the motion in terms of the actual velocity at which the tool should travel in whatever coordinate system is specified. The units used are either inches per minute or millimeters per minute. It is independent of the type of motion (linear or circular) and the length of the motion.

The last, feed per spindle revolution, is common for turning, and for milling operations such as tapping a threaded hole, where the Z traversal of the spindle must be directly correlated to the rotational position of the spindle.

The question at hand is which is the best feedrate mode to use to cut an involute using the generation principle. It should be noted that G93 allows all axes to move independently of each other. Each axis will move at a constant velocity, calculated such that they all begin and end at the same time, and take the time indicated by the feedrate value. This is important for the generation principle.

Another important consideration is the coordinate system to be used. A coordinate system such as G54 will be used, with "work offsets" that are set in the machine XYZ coordinate system. But if the part is moving within this coordinate system, such as in the present case where the part sits on a rotational C axis, which is in turn stacked on a B axis, then it is not so simple. Does the coordinate system move with the part, or not? Normally, it would not, which means that any XYZ coordinates sent to the tool may not be at the intended place on the part. There is another mode, however, sometimes called "five axis mode," called by G43.4. Technically it is a type of tool length offset, to be chosen as opposed to the usual G43 tool length offset. The only difference is that where G43 allows the coordinate system to remain stationary within the machine, G43.4 causes the coordinate system to translate and rotate with the part, wherever it may be in the machine workspace.

When using G43.4, if a G01 command is specified, the machine adjusts all axes to move such that the tool actually travels in a straight line within the part coordinate system. In the case at hand, a G01 is used to cause straight line motion of the tool, but an involute is desired in the part, which is clearly not a straight line (or a circular arc, for that matter). Thus, G43.4 does not accomplish the desired task, and G43 must be used.

Inverse time feedrate (G93) in conjunction with G43 is used to “trick” the machine into moving in accordance with the generation principle.

### 4.2.2 Specifics

The machine needs some feedrate value in the G code. This value can take one of three forms, described in the previous section. In this case, inverse time feed rates (G93) are used for those portions of the cutting process directly generating the involute surface of the gear teeth. Inverse time feedrate is simply the reciprocal of the desired time duration of the cut. In order to get a value for this feedrate corresponding to a particular motion, the desired feedrate in inches per minute (or mm per minute) $F_{IPM}$ must be known, as well as the total traversal distance for the motion. The relationship between $F_{IPM}$ and the inverse time feedrate $F_{INV}$ is shown in Equation 4.22, where $D$ is the total traversal distance of the motion. This is a representation involving linear terms.

$$F_{INV} = \frac{1}{t} = \frac{F_{IPM}}{D}, \tag{4.22}$$

The following relations are obvious, if $\omega$ is assumed to be constant.

$$\xi_i = \omega_i t_i \tag{4.23}$$

$$t_i = \frac{\xi_i}{\omega_i} \tag{4.24}$$

Thus, a rotational equivalent to Equation 4.22 is

$$F_{INV} = \frac{1}{t_i} = \frac{\omega}{\xi_i} \tag{4.25}$$

Equation 4.21 can be solved for $\omega$ as

$$\omega = \frac{F_{IPM}}{R_b\sqrt{2 - 2\sqrt{\xi_i^2 + 1}\cos(\tan^{-1}(\xi_i)) + \xi_i^2}}. \tag{4.26}$$

Substituting Equation 4.26 into Equation 4.25, an expression for the inverse time feedrate for any involute between the base circle ($\xi = 0$) and some roll angle $\xi_i$, traveling either direction, is obtained as the following:

$$F_{INV} = \frac{F_{IPM}}{\xi_i R_b\sqrt{2 - 2\sqrt{\xi_i^2 + 1}\cos(\tan^{-1}(\xi_i)) + \xi_i^2}}. \tag{4.27}$$

At this inverse time feedrate, the desired linear feedrate $F_{IPM}$ will be reached at the outermost point on the involute.

If it is desired to have this represented in terms of arc $s$ length instead of roll angle $\xi$, the resulting relation is

$$F_{INV} = \frac{F_{IPM}}{\sqrt{\frac{2s}{R_b}} R_b\sqrt{2 - 2\sqrt{\frac{2s}{R_b} + 1}\cos\left(\tan^{-1}\left(\sqrt{\frac{2s}{R_b}}\right)\right) + \frac{2s}{R_b}}}. \tag{4.28}$$

## 4.3 Incremented Motion

The above derivation applies to a motion beginning on the base circle, and traveling outward to some arbitrary point along the involute, or vice versa. What would be better, however, would be to get the inverse time feedrate needed to go from one arbitrary point along the involute to another. The following developments of the expressions above accomplish this.

Consider a point $P_i$ on the involute. Now consider a second point $P_{i+1}$ along the same involute, further out, beyond $P_i$, each with a known $\xi$. The amount of time to traverse from $P_i$ to $P_{i+1}$, assuming a constant rotational velocity $\omega$, where $t_i$ and $t_{i+1}$ are the traversal times to go from $\xi = 0$ to $\xi_i$ or $\xi_{i+1}$, respectively, at rotational speed $\omega$, is

$$t_{i,i+1} = t_{i+1} - t_i = \frac{\xi_{i+1}}{\omega} - \frac{\xi_i}{\omega} = \frac{\xi_{i+1} - \xi_i}{\omega} \tag{4.29}$$

$$\omega = \frac{\xi_i}{t_i} = \frac{\xi_{i+1}}{t_{i+1}} \tag{4.30}$$

$$t_{i,i+1} = \frac{\xi_{i+1} - \xi_i}{\xi_{i+1}} t_{i+1}. \tag{4.31}$$

Here, it is worth noting that the time to go from point $i$ to $i+1$ cannot be negative, but point $i+1$ may be closer to the base radius than point $i$. Thus, a more accurate representation is:

$$t_{i,i+1} = \frac{|\xi_{i+1} - \xi_i|}{\xi_{i+1}} t_{i+1}. \tag{4.32}$$

The $F_{INV,i+1}$ corresponding to this incremental motion would be

$$F_{INV,i+1} = \frac{1}{t_{i,i+1}} = \frac{\xi_{i+1}}{t_{i+1}|\xi_{i+1} - \xi_i|}, \tag{4.33}$$

From Equations 4.24 and 4.26,

$$t_{i+1} = \frac{\xi_{i+1} R_b \sqrt{2 - 2\sqrt{\xi_{i+1}^2 + 1}\cos(\tan^{-1}(\xi_{i+1})) + \xi_{i+1}^2}}{F_{IPM}}. \tag{4.34}$$

Substitution into Equation 4.33 yields

$$F_{INV,i+1} = \frac{1}{t_{i,i+1}} = \frac{F_{IPM}}{R_b|\xi_{i+1} - \xi_i| \sqrt{\begin{array}{l} 2 + \xi_{i+1}^2 \\ - 2\sqrt{\xi_{i+1}^2 + 1}\cos(\tan^{-1}(\xi_{i+1})) \end{array}}}. \tag{4.35}$$

In terms of arc length, according to Equation 2.12, Equation 4.35 becomes

$$F_{INV,i+1} = \frac{F_{IPM}}{2\sqrt{R_b}|\sqrt{s_{i+1}} - \sqrt{s_i}| \sqrt{\begin{array}{l} 1 + \frac{s_{i+1}}{R_b} \\ - \sqrt{\frac{2s_{i+1}}{R_b} + 1}\cos\left(\tan^{-1}\left(\sqrt{\frac{2s_{i+1}}{R_b}}\right)\right) \end{array}}}. \tag{4.36}$$

Equations 4.35 and 4.36 should work traveling inward or outward along the involute, as long as the $i+1$ term is the larger roll angle (or arc length).

In the application at hand, the commanded C axis position is the roll angle, so even though the coordinates may be generated in terms of arc length, the feedrates will always be determined by roll angle.

## 4.4 Considering a Normal Offset

The analysis above is considering cutting along the nominal pure involute, essentially a finish pass. For the initial roughing process however, the tool will be offset a constant normal amount from the pure involute flank, and the feed velocity will again be different. This is simpler than it may seem at first glance, however.

If two adjacent involutes on the same base circle are separated by an offset angle $\Lambda_d$ (in radians), then the normal distance between them at every point is

$$d = R_b \Lambda_d \tag{4.37}$$

If it is desired to offset from a certain involute by an amount $d$, then the resulting curve is an involute with an additional offset angle of

$$\Lambda_d = \frac{d}{R_b} \tag{4.38}$$

If this offset angle is simply added to the roll angle at every point in Equation 4.35, an equation is obtained which takes into account a normal offset distance from the pure involute.

$$F_{INV,i+1} = \frac{F_{IPM}}{R_b|\xi_{i+1} - \xi_i| \sqrt{\begin{array}{l} 2 + \left(\xi_{i+1} + \frac{z}{|z|}\frac{d}{R_b}\right)^2 \\ -2\sqrt{\left(\xi_{i+1} + \frac{z}{|z|}\frac{d}{R_b}\right)^2 + 1}\cos\left(\tan^{-1}\left(\xi_{i+1} + \frac{z}{|z|}\frac{d}{R_b}\right)\right) \end{array}}}. \tag{4.39}$$

Note that the $\frac{z}{|z|}$ term was added to take into account if the gear is internal or external. For an external gear, the offset will be outward, on the convex side of the involute, which corresponds to an increase in the roll angle. For an internal gear, the offset will be going in the opposite direction, corresponding to a smaller roll angle. This is shown in Figure 4.2.

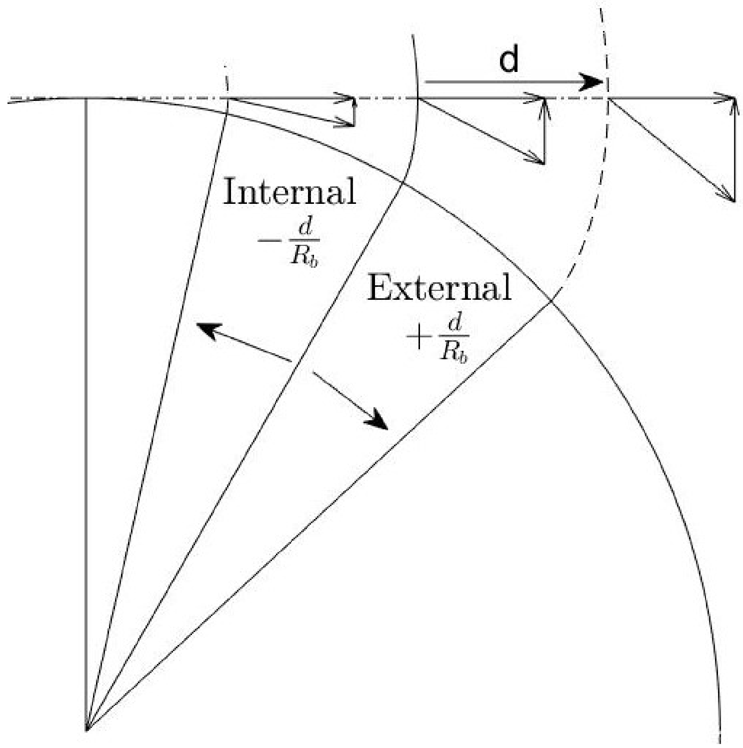


Figure 4.2: Cutting Speed Vectorial Representation, Modified for External and Internal Gears

# CHAPTER 5: RESULTS AND CONCLUSIONS

Both spur and helical, internal and external gears were machined. All measurements were performed on a Leitz PMMF302016 coordinate measuring machine (CMM) by Dr. Kang Ni and Yue Peng. The roughing and finishing machining parameters for the following gears are as listed in Tables 5.1 and 5.2, respectively.

Table 5.1: Gear Machining Roughing Parameters

| | |
|---|---|
| Tool Diameter | 1/8 in (3.175 mm) |
| Number of Flutes | 3 |
| Tool Helix Angle | 60° |
| Spindle Speed | 12000 $min^{-1}$ |
| Feed Per Tooth | 0.0508 mm (0.002 in) |
| Radial Immersion | 25% |
| Axial Step Size | 1 mm |
| Involute Step Size | 1 mm |

Table 5.2: Gear Machining Finishing Parameters

| | |
|---|---|
| Tool Diameter | 1/8 in (3.175 mm) |
| Number of Flutes | 3 |
| Tool Helix Angle | 60° |
| Spindle Speed | 12000 $min^{-1}$ |
| Feed Per Tooth | 0.0254 mm (0.001 in) |
| Finish Pass Stock Thickness | 0.1 mm |
| Axial Step Size | 0.5 mm |
| Involute Step Size | 1 mm |

## 5.1 External Spur Gear

External spur gears with the following parameters were machined and measured. One such gear is shown in Figure 5.1.

Table 5.3: External Spur Gear Parameters

| | |
|---|---|
| Transverse Module | 5 mm |
| Transverse Pressure Angle | 20° |
| Number of Teeth | 15 |
| Helix Angle | 0° |
| Face Width | 10 mm |

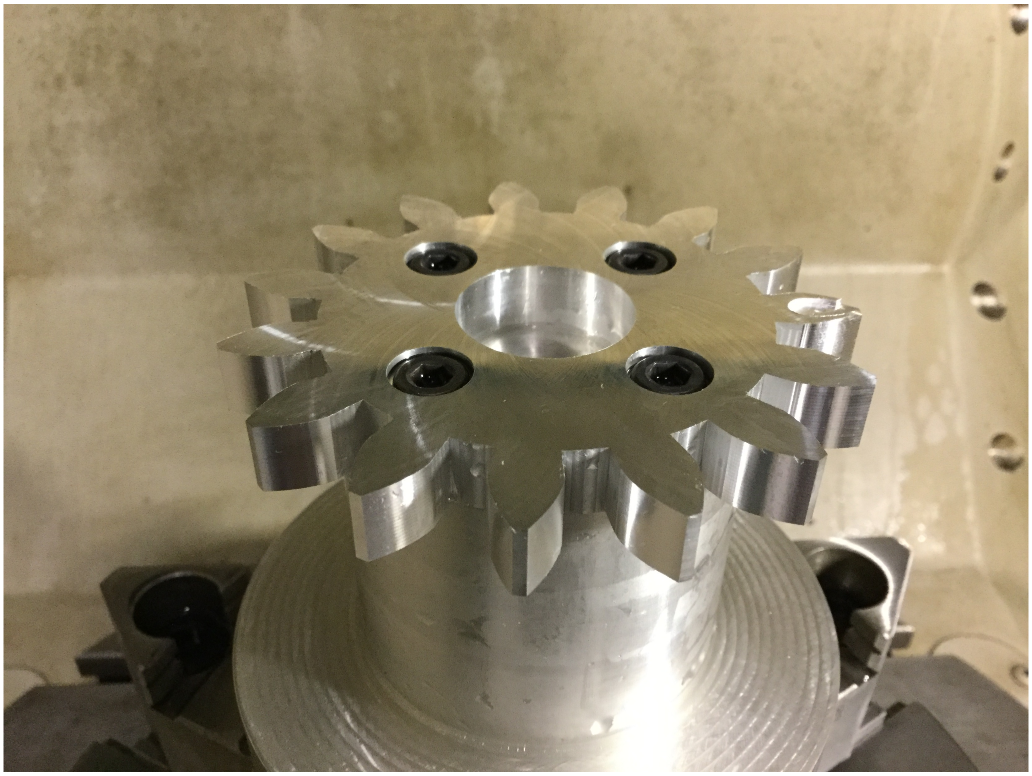

Figure 5.1: Machined External Spur Gear

## 5.2 External Helical Gear

A few external helical gears were machined, with the parameters as given in Table 5.4. A figure of one such gear is shown in Figure 5.2.

Table 5.4: External Helical Gear Parameters

| | |
|---|---|
| Transverse Module | 5 mm |
| Transverse Pressure Angle | 20° |
| Number of Teeth | 15 |
| Helix Angle | 10° |
| Face Width | 10 mm |

Figure 5.2: Machined External Helical Gear

## 5.3 Internal Spur Gear

Internal spur gears were machined, with the parameters as given in Table 5.5, and a gear is shown in Figure 5.3.

Table 5.5: Internal Spur Gear Parameters

| | |
|---|---|
| Transverse Module | 5 mm |
| Transverse Pressure Angle | 20° |
| Number of Teeth | -30 |
| Helix Angle | 0° |
| Face Width | 10 mm |

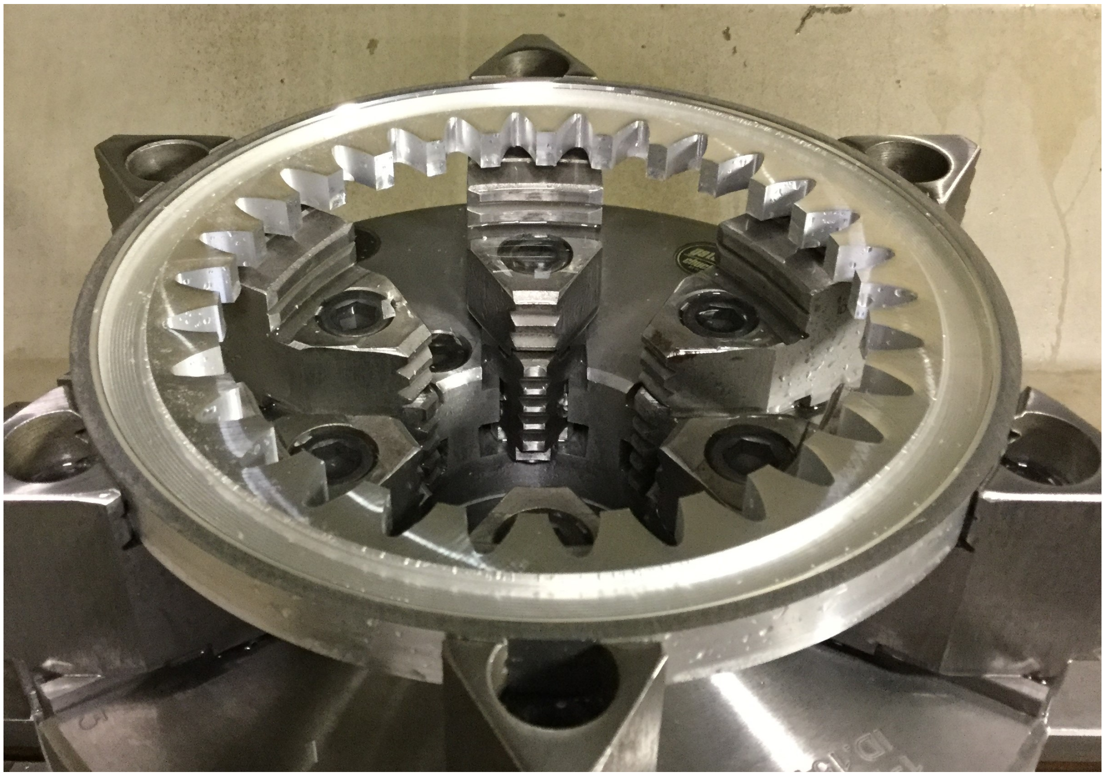

Figure 5.3: Machined Internal Spur Gear

## 5.4 Internal Helical Gear

Internal helical gears were machined, with the parameters as given in Table 5.6. One such gear is shown in Figure 5.4, and a close up view of some teeth is given in Figure 5.5.

Table 5.6: Internal Helical Gear Parameters

| | |
|---|---|
| Transverse Module | 5 mm |
| Transverse Pressure Angle | 20° |
| Number of Teeth | -30 |
| Helix Angle | 10° |
| Face Width | 10 mm |

Figure 5.4: Machined Internal Helical Gear with 10° Helix Angle

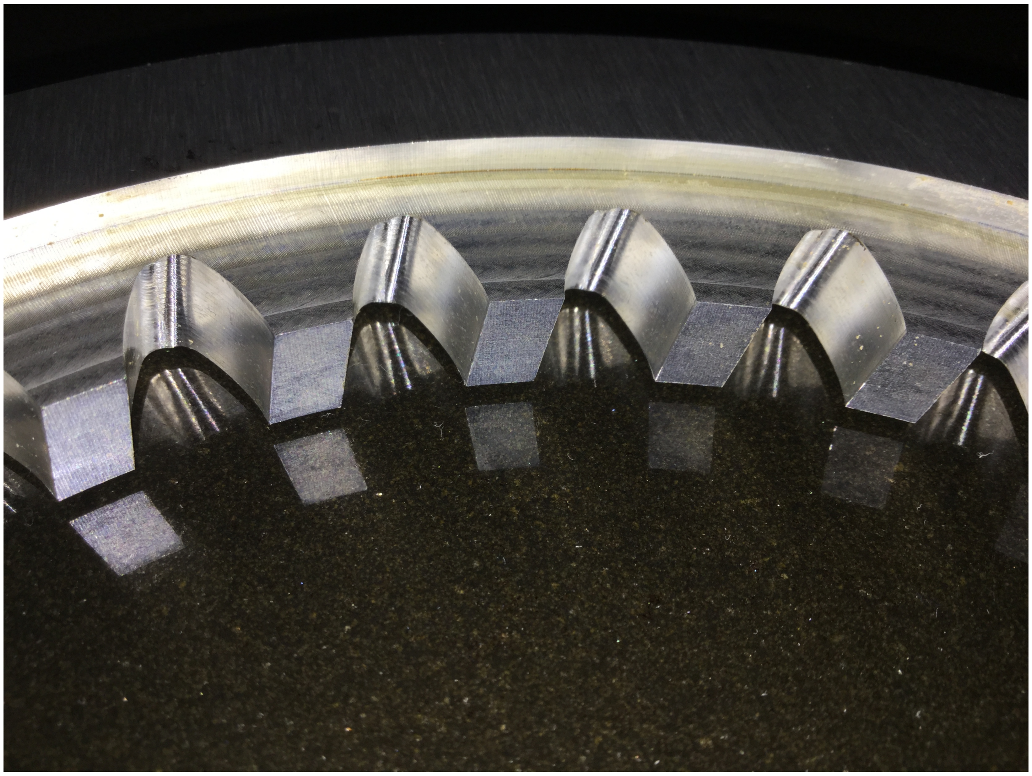

Figure 5.5: Close Up View of Teeth of a Machined Internal Helical Gear with -10° Helix Angle

## 5.5 Discussion

### 5.5.1 Preliminary Explanations

To begin the discussion of the measurement results, the measurement procedure must first be explained. On each gear, at least one tooth, and at most four teeth were measured on a Leitz PMMF302016 coordinate measuring machine (CMM). Both flanks were measured on each tooth that was measured. On each of these flanks, three lines were measured in both the profile (radial) and lead (axial) directions, as shown in Figures 5.6 and 5.7, respectively.

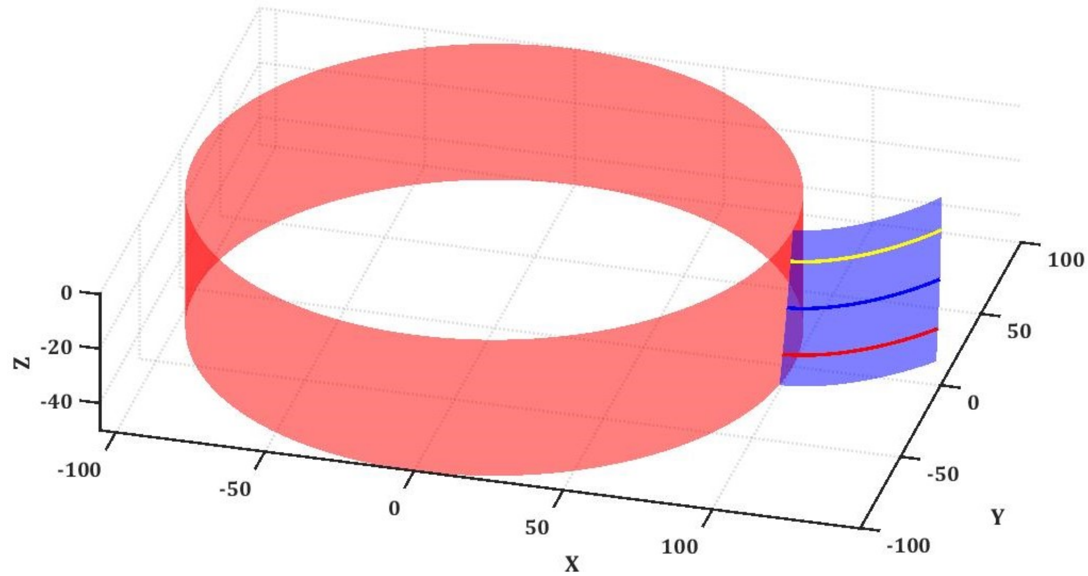


Figure 5.6: Measurement Lines in the Profile Direction

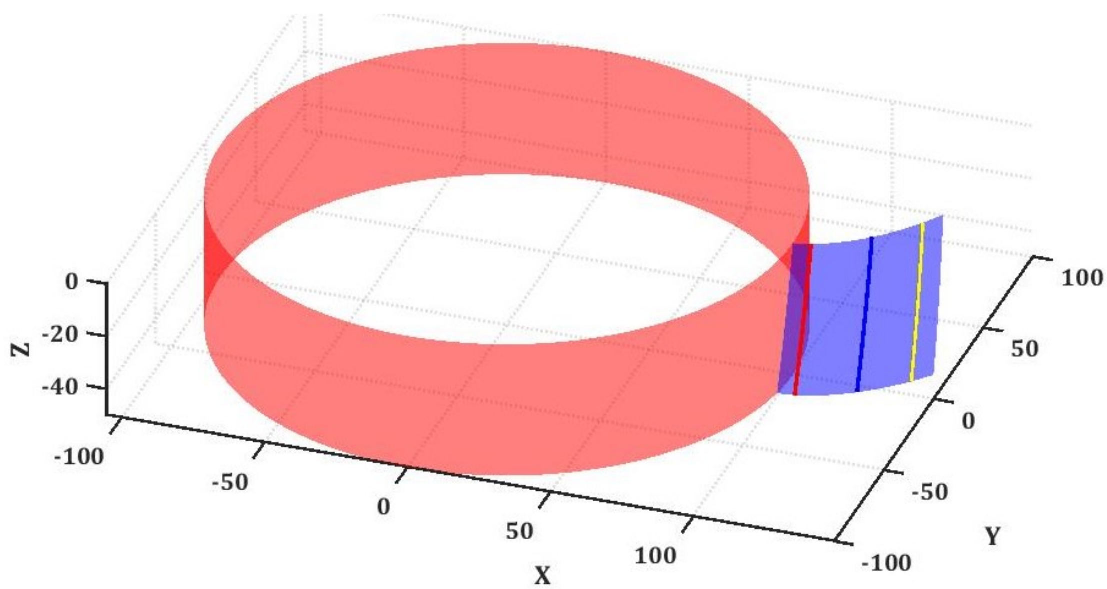


Figure 5.7: Measurement Lines in the Lead Direction

The colored lines in Figures 5.6 and 5.7 correspond to the colors shown in the sample plots in Figures 5.8 and 5.9, respectively. In both the profile and lead directions, the ideal geometry is mapped to a straight line oriented along an X axis, and deviations from this straight line are plotted in the Y direction. Positive deviations indicate extra material. This holds true for both external and internal gears. See Figures 5.8

and 5.9 for some sample measurement data in both directions.

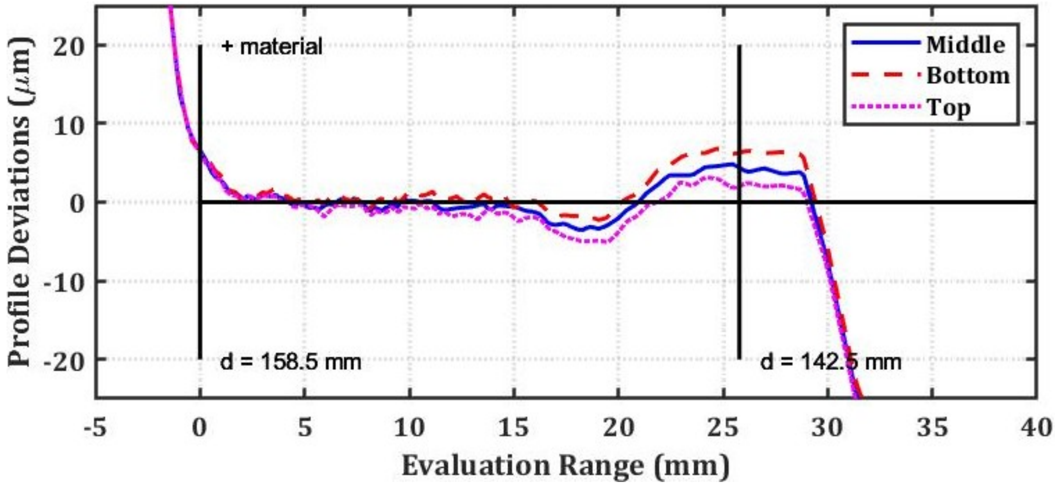


Figure 5.8: Sample Measurement Data in Profile Direction

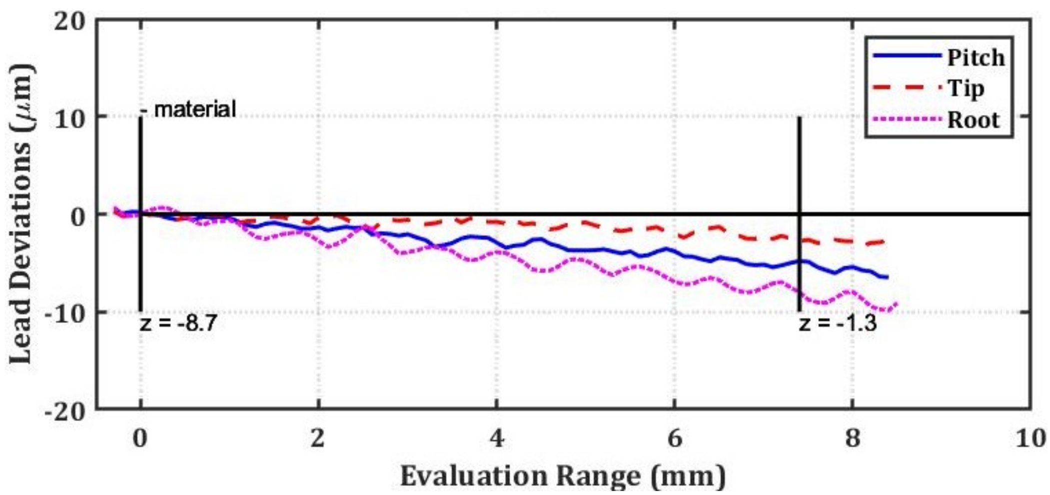


Figure 5.9: Sample Measurement Data in Lead Direction

The six measurement parameters that are primarily looked at are shown in Table 5.7.

Table 5.7: Measurement Deviation Parameters

| Parameter | Symbol |
|---|---|
| Profile Slope Deviation | $f_{H\alpha}$ |
| Profile Form Deviation | $f_{f\alpha}$ |
| Total Profile Deviation | $F_{\alpha}$ |
| Helix (lead) Slope Deviation | $f_{H\beta}$ |
| Helix (lead) Form Deviation | $f_{f\beta}$ |
| Total Helix (lead) Deviation | $F_{\beta}$ |

## 5.6 A Synthesis of Profile Measurements

This section points out the best and worst measurements in the profile direction, and discusses some factors that influenced the data.

At the outset, some bounding information is given in Table 5.8. The largest and smallest deviations for each parameter are given, as well as the largest and smallest absolute values of deviations. The mean values and standard deviations for each are also given.

Table 5.8: A Comprehensive Synthesis of Profile Deviation Measurement Data

| Value | $f_{H\alpha}$ ($\mu m$) | $F_{\alpha}$ ($\mu m$) | $f_{f\alpha}$ ($\mu m$) |
|---|---|---|---|
| Highest | 5.946 | 32.211 | 26.496 |
| Lowest | -11.136 | 2.978 | 2.467 |
| Mean | -2.312 | 8.765 | 7.511 |
| Standard Deviation | 3.285 | 6.268 | 5.706 |
| Worst (Largest Absolute Deviations) | 11.136 | 32.211 | 26.496 |
| Best (Smallest Absolute Deviations) | 0.148 | 2.978 | 2.467 |
| Mean Absolute Deviation | 3.329 | 8.765 | 7.511 |
| Standard Deviation of Absolute Values | 2.248 | 6.268 | 5.706 |

### 5.6.1 Worst Case Profile Slope, Form, and Total Deviations

The worst case profile form deviation $f_{f\alpha}$, and total profile deviation $F_{\alpha}$, were found on an internal helical gear, on a left flank, right in the middle of the facewidth (about 5 mm down from the top reference surface). The profile measurement data for this flank is shown in Figure 5.10, and full analysis results for this flank are shown in Table 5.9.

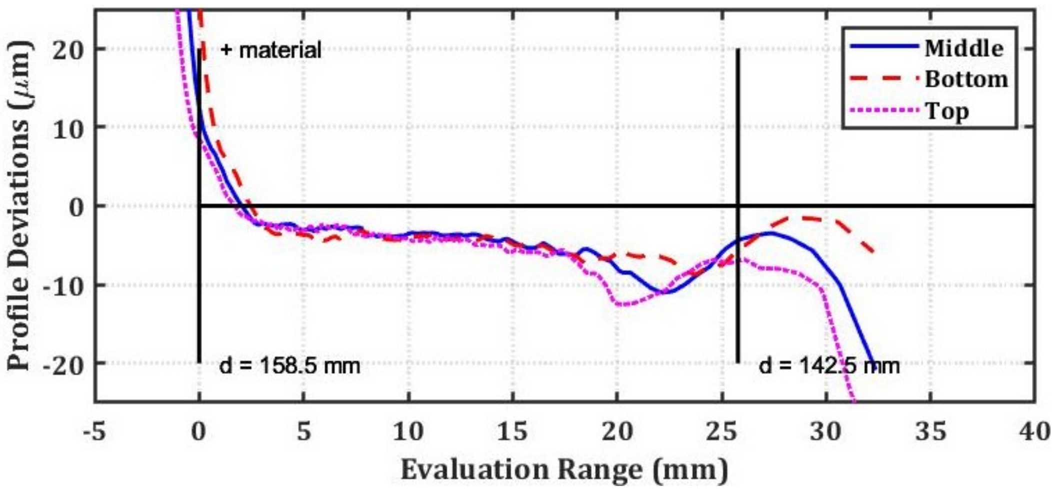


Figure 5.10: Worst Profile Data - Internal Helical Gear, Tooth 1, Left Flank

Table 5.9: Internal Helical Gear, Tooth 1, Left Flank, Profile

| Parameter | Top | Middle | Bottom |
|---|---|---|---|
| $f_{H\alpha}$ | -8.591 | -7.266 | -11.136 |
| $f_{f\alpha}$ | 17.168 | 26.496 | 13.532 |
| $F_{\alpha}$ | 23.503 | 32.211 | 21.012 |

Note the waviness exhibited in Figure 5.10, which contributed to the form deviation. This is most likely due to the segmentation by arc length of the involute during generation milling. At each segmentation point, the feed velocity of each axis, C and X, changes. If the machine accelerations of each axis are not correlated well, form deviations will result. This could be minimized by using smaller segments, so that the period over which the acceleration occurs is smaller. The problem with this is that the more points you add, the larger the file becomes. If the distance between points decreases too much, the machine control could have trouble keeping up as well, which can cause the machine to go slower than desired, and not perform the intended accelerations as programmed.

### 5.6.2 Best Case Profile Slope Deviation

The best case profile slope deviation $f_{H\alpha}$ was also found on the external helical gear, actually on the same tooth as before, but on the right flank, near the top (reference) face. The full data for this flank is shown in Figure 5.11 and Table 5.10.

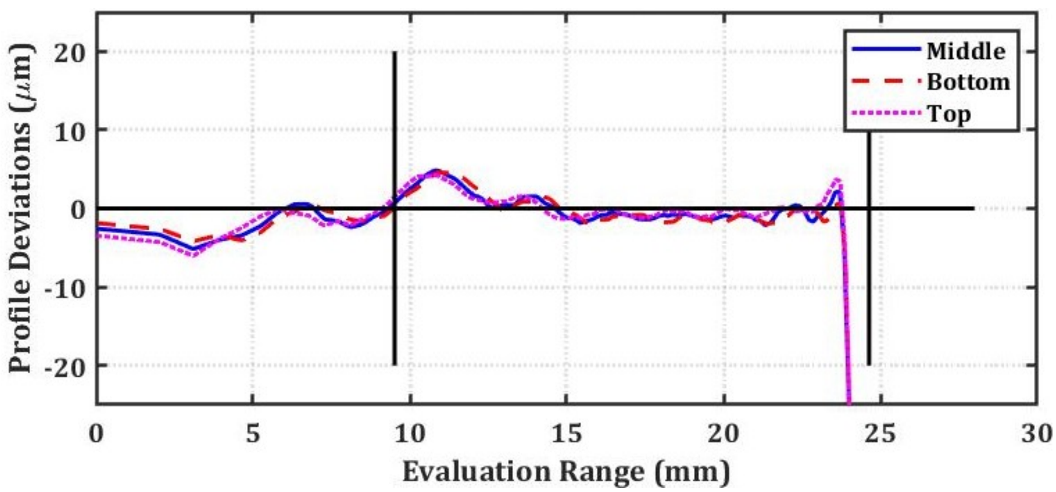


Figure 5.11: Best Profile Slope Data - External Helical Gear, Tooth 1, Right Flank

Table 5.10: External Helical Gear, Tooth 1, Right Flank, Profile

| Parameter | Top | Middle | Bottom |
|---|---|---|---|
| $f_{H\alpha}$ | -0.148 | -0.482 | -0.603 |
| $f_{f\alpha}$ | 7.320 | 8.681 | 8.788 |
| $F_{\alpha}$ | 7.268 | 8.513 | 8.583 |

Note that the profile form deviations are above 7 $\mu m$ for the whole flank, but also note that the evaluation range in Figure 5.11 includes where the probe falls off the tip of the tooth, so the slope is significantly lower than what is shown in the figure and table.

### 5.6.3 Best Case Profile Form Deviation

The best case profile form deviation was found on the external spur gear, tooth 9, on the left flank near the top (reference) face. The full measurement data for this flank is shown in Figure 5.12 and Table 5.11.

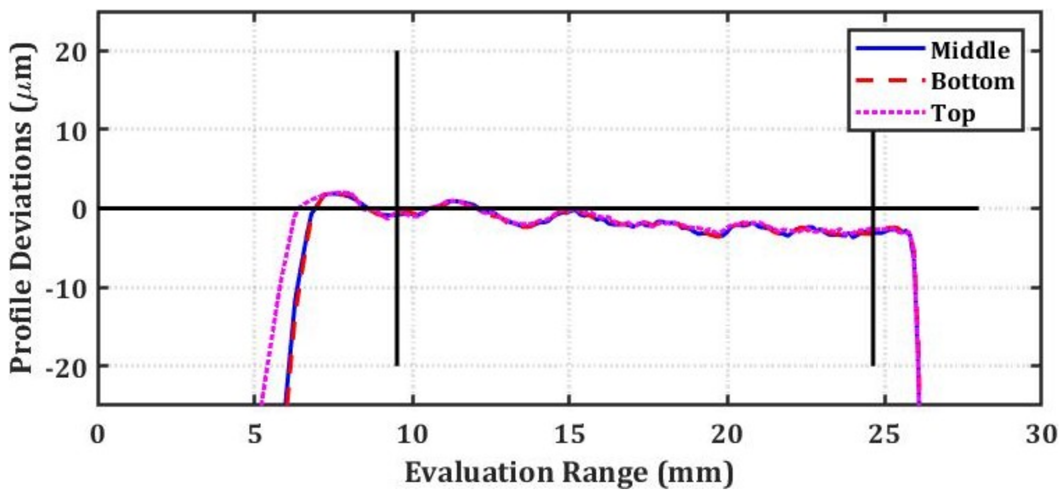


Figure 5.12: Best Profile Form Data - External Spur Gear, Tooth 9, Left Flank

Table 5.11: External Spur Gear, Tooth 9, Left Flank, Profile

| Parameter | Top | Middle | Bottom |
|---|---|---|---|
| $f_{H\alpha}$ | -3.225 | -3.556 | -3.497 |
| $f_{f\alpha}$ | 2.467 | 2.725 | 3.162 |
| $F_{\alpha}$ | 4.236 | 4.597 | 4.872 |

Note that all deviations for this flank are under 5 $\mu m$.

### 5.6.4 Best Case Total Profile Deviation

The best case total profile deviation $F_{\alpha}$ was found on tooth five of the external spur gear, on the left flank, near the top face of the gear. The full profile measurement results for this flank are given in Figure 5.13 and Table 5.12.

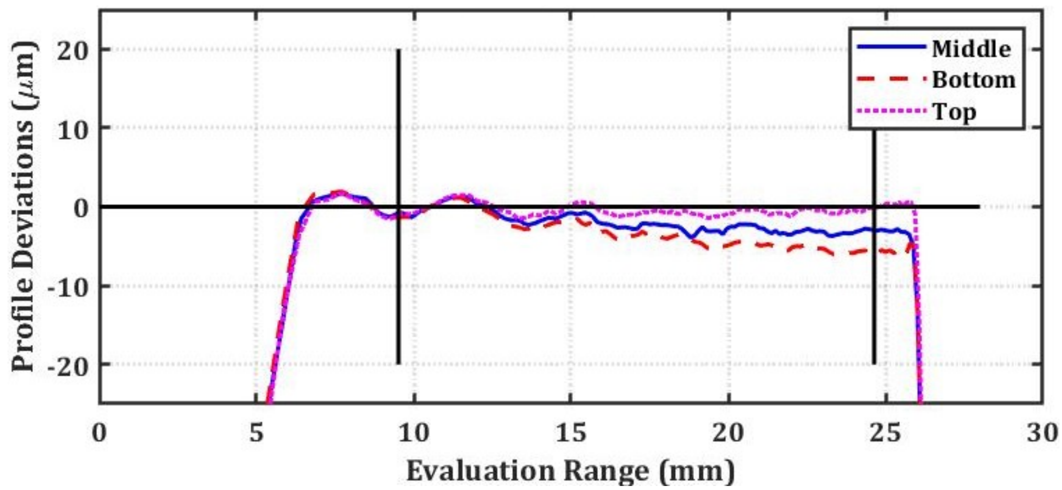


Figure 5.13: Best Total Profile Deviation Data - External Spur Gear, Tooth 5, Left Flank

Table 5.12: External Spur Gear, Tooth 5, Left Flank, Profile

| Parameter | Top | Middle | Bottom |
|---|---|---|---|
| $f_{H\alpha}$ | -0.896 | -3.872 | -6.418 |
| $f_{f\alpha}$ | 2.878 | 3.267 | 3.564 |
| $F_{\alpha}$ | 2.978 | 5.082 | 7.311 |

## 5.7 A Synthesis of Lead Measurements

The following subsections provide information regarding the best and worst measurement data in the lead direction.

### 5.7.1 Some Outliers

At the outset, some bounding information is given in Table 5.13. The largest and smallest deviations for each parameter are given, as well as the largest and smallest absolute values of deviations. The mean values and standard deviations for each are also given. Table 5.13 includes all measurement data collected.

Table 5.13: A Comprehensive Synthesis of Lead Deviation Measurement Data

| Value | $f_{H\beta}$ ($\mu m$) | $F_\beta$ ($\mu m$) | $f_{f\beta}$ ($\mu m$) |
|---|---|---|---|
| Highest | 349.118 | 225.273 | 81.457 |
| Lowest | -10.849 | 1.326 | 0.967 |
| Mean | 7.693 | 10.344 | 3.902 |
| Standard Deviation | 50.211 | 31.659 | 11.695 |
| Worst (Largest Absolute Deviations) | 349.118 | 225.273 | 81.457 |
| Best (Smallest Absolute Deviations) | 0.035 | 1.326 | 0.967 |
| Mean Absolute Deviation | 12.654 | 10.344 | 3.902 |
| Standard Deviation of Absolute Deviations | 49.195 | 31.659 | 11.695 |

There were two notable outliers, both near the roots of flanks on internal gears. The full measurement data for the two teeth are shown in Figures 5.14 and 5.15, and Tables Table 5.14 and Table 5.15.

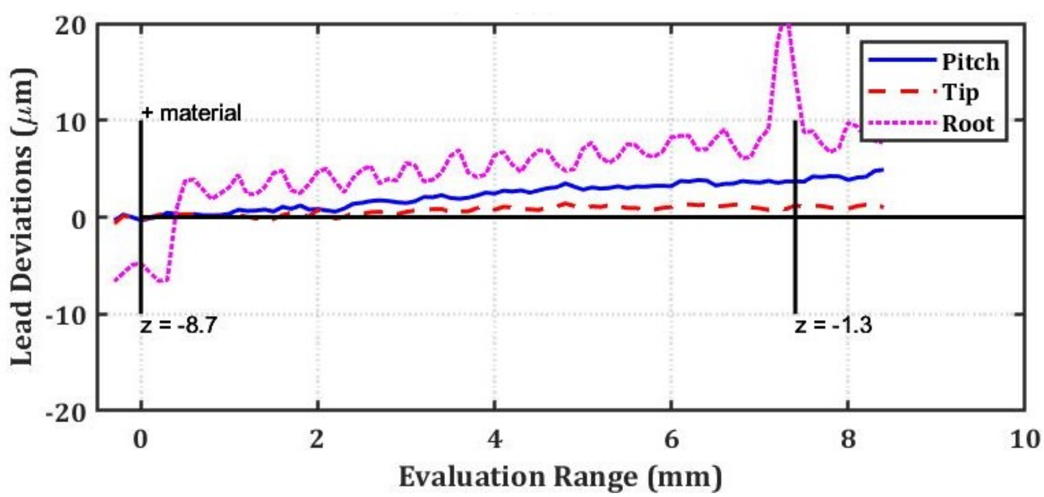


Figure 5.14: Outlier Data - Internal Spur Gear, Tooth 1, Right Flank

Table 5.14: Internal Spur Gear, Tooth 1, Right Flank, Lead

| Parameter | Root | Pitch | Tip |
|---|---|---|---|
| $f_{H\beta}$ | 1.782 | 5.603 | 14.196 |
| $f_{f\beta}$ | 1.086 | 1.459 | 18.260 |
| $F_{\beta}$ | 1.610 | 5.013 | 28.306 |

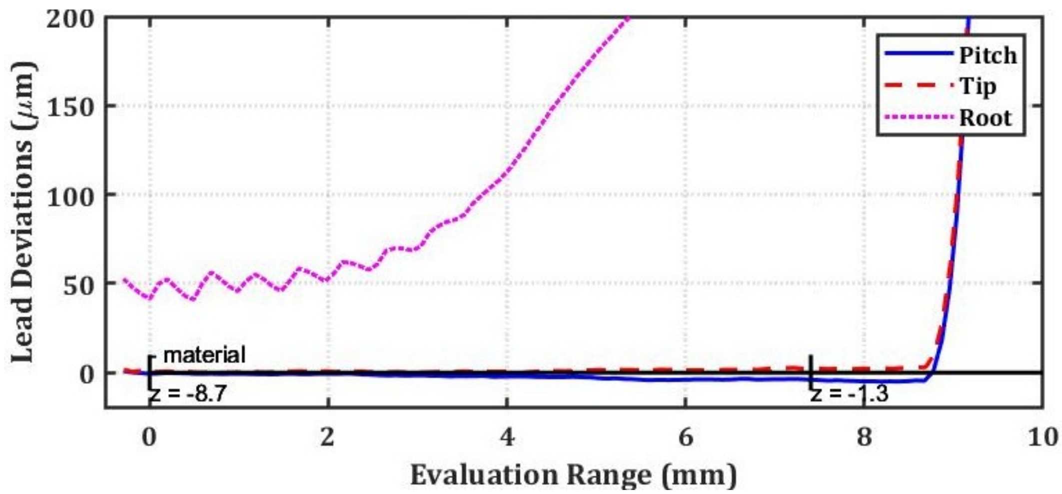


Figure 5.15: Outlier Lead Data - Internal Helical Gear, Tooth 1, Left Flank

Table 5.15: Internal Helical Gear, Tooth 1, Left Flank, Lead

| Parameter | Root | Pitch | Tip |
|---|---|---|---|
| $f_{H\beta}$ | -5.744 | 2.523 | 349.118 |
| $f_{f\beta}$ | 1.672 | 1.994 | 81.457 |
| $F_{\beta}$ | 4.875 | 2.765 | 225.273 |

After removing the data from those particular aberrant measurements shown above, the new summary data for the lead direction is shown below in Table 5.16.

Table 5.16: A Comprehensive Synthesis of Lead Deviation Measurement Data

| Value | $f_{H\beta}$ $(\mu m)$ | $F_\beta$ $(\mu m)$ | $f_{f\beta}$ $(\mu m)$ |
|---|---|---|---|
| Highest | 11.011 | 16.951 | 10.802 |
| Lowest | -10.849 | 1.326 | 0.967 |
| Mean | 0.129 | 5.281 | 1.904 |
| Standard Deviation | 6.070 | 2.934 | 1.829 |
| Worst (Largest Absolute Deviations) | 11.011 | 16.951 | 10.802 |
| Best (Smallest Absolute Deviations) | 0.035 | 1.326 | 0.967 |
| Mean Absolute Deviation | 5.091 | 5.281 | 1.904 |
| Standard Deviation of Absolute Deviations | 3.310 | 2.934 | 1.829 |

### 5.7.2 Worst Case Lead Form Deviation

The worst case lead form deviation $f_{f\beta}$ was found on an external helical gear, on a right flank, at the pitch diameter. The measurement data is shown in Figure 5.16 and Table 5.17.

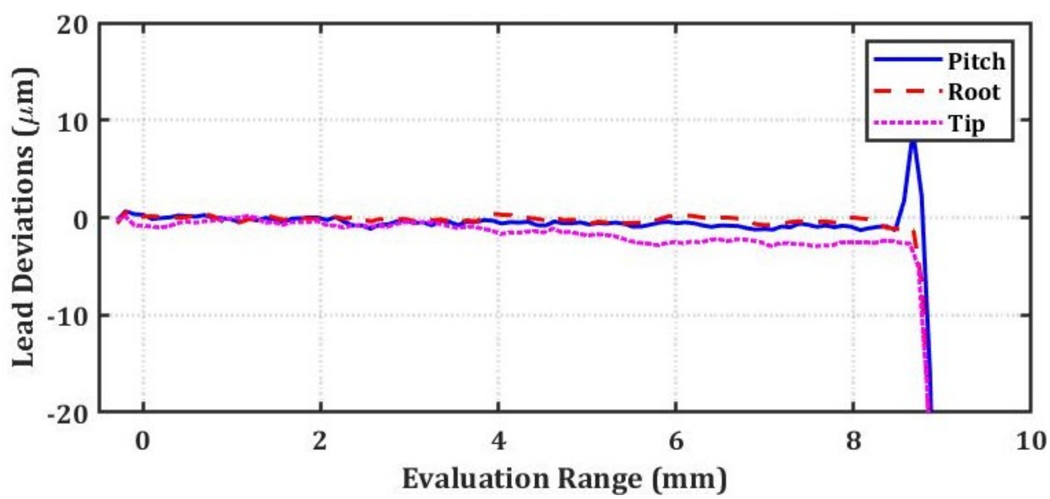


Figure 5.16: Worst Lead Form Deviation Data - External Helical Gear, Tooth 1, Right Flank

Table 5.17: External Helical Gear, Tooth 1, Right Flank, Lead

| Parameter | Root | Pitch | Tip |
|---|---|---|---|
| $f_{H\beta}$ | 0.463 | 1.320 | 3.612 |
| $f_{f\beta}$ | 1.115 | 10.802 | 1.651 |
| $F_{\beta}$ | 1.352 | 10.091 | 3.059 |

### 5.7.3 Worst Case Lead Slope and Total Lead Deviation

The worst lead slope and total lead deviations, $f_{H\beta}$ and $F_{\beta}$ respectively, were found on the internal helical gear, on the right flank. Full measurement data for this flank is shown in Figure 5.17 and Table 5.18.

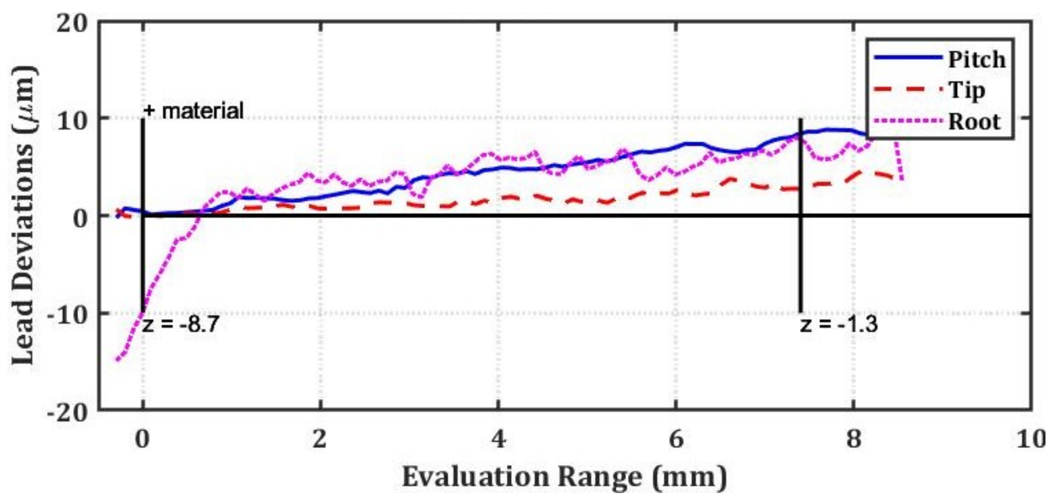


Figure 5.17: Worst Lead Slope and Total Lead Deviation Data - Internal Helical Gear, Tooth 1, Right Flank

Table 5.18: Internal Helical Gear, Tooth 1, Right Flank, Lead

| Parameter | Root | Pitch | Tip |
|---|---|---|---|
| $f_{H\beta}$ | 11.011 | 3.938 | 10.302 |
| $f_{f\beta}$ | 1.833 | 2.133 | 10.042 |
| $F_{\beta}$ | 8.874 | 4.643 | 16.951 |

### 5.7.4 Best Case Lead Form Deviation

The best case lead form deviations $f_{f\beta}$ were found on an external spur gear, on the left flank of tooth 5. Full measurement data for this flank is shown in Figure 5.18 and Table 5.19.

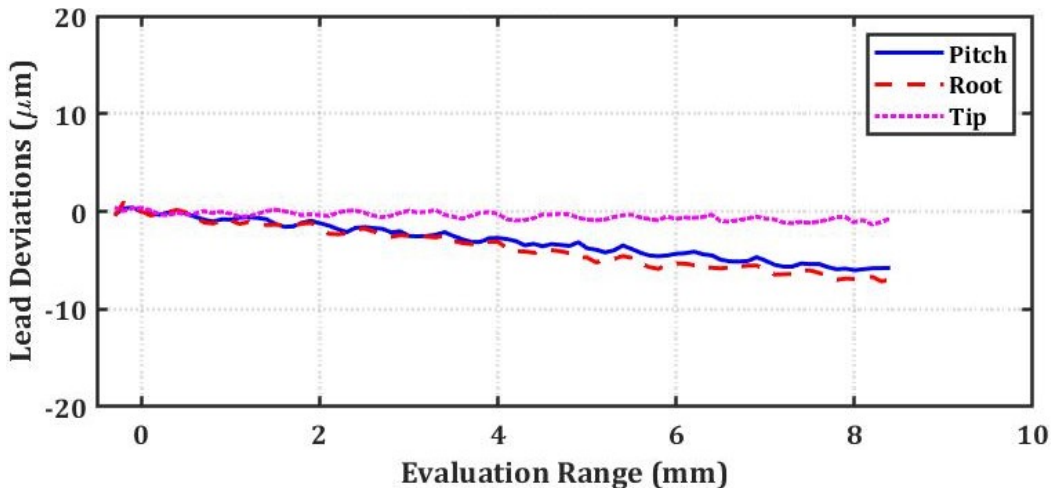


Figure 5.18: Best Lead Form Deviation Data - External Spur Gear, Tooth 5, Left Flank

Table 5.19: External Spur Gear, Tooth 5, Left Flank, Lead

| Parameter | Root | Pitch | Tip |
|---|---|---|---|
| $f_{H\beta}$ | -7.403 | -8.901 | -1.165 |
| $f_{f\beta}$ | 0.967 | 1.469 | 0.967 |
| $F_{\beta}$ | 6.043 | 7.392 | 1.619 |

Note that this same flank had the best total profile deviation, $F_{\alpha}$.

### 5.7.5 Best Case Lead Slope Deviation

The best lead slope deviation $f_{H\beta}$, was found on an external spur gear, tooth 1, on the left flank at the pitch circle. The full measurement data for this flank is shown in Figure 5.19 and Table 5.20.

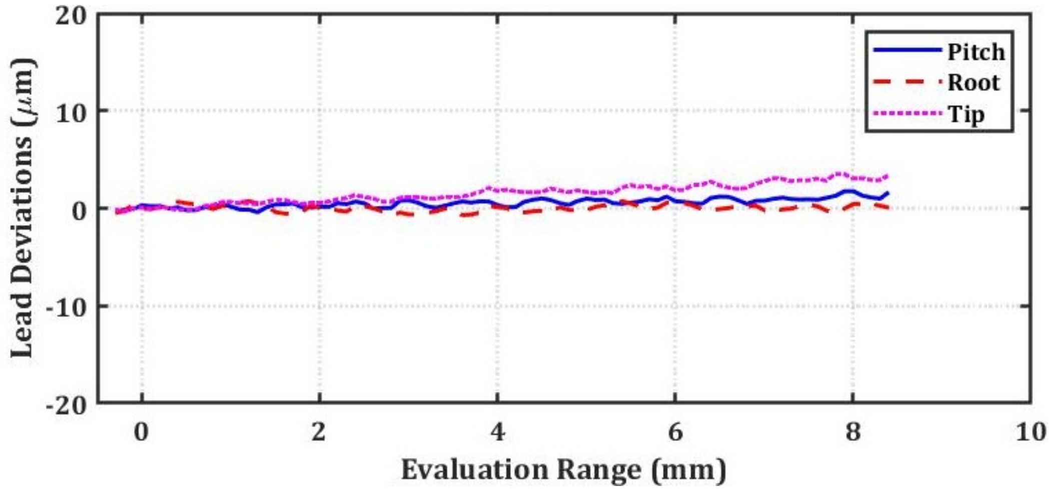


Figure 5.19: Best Lead Slope Deviation Data - External Spur Gear, Tooth 1, Left Flank

Table 5.20: External Spur Gear, Tooth 1, Left Flank, Lead

| Parameter | Root | Pitch | Tip |
|---|---|---|---|
| $f_{H\beta}$ | 1.316 | -0.035 | 3.724 |
| $f_{f\beta}$ | 1.274 | 1.434 | 1.137 |
| $F_{\beta}$ | 2.152 | 1.427 | 3.724 |

### 5.7.6 Best Case Total Lead Deviation

The best case total lead deviation was found on an external helical gear on tooth 1 on the left flank, near the root. The full measurement data for this flank is shown in Figure 5.20 and Table 5.21.

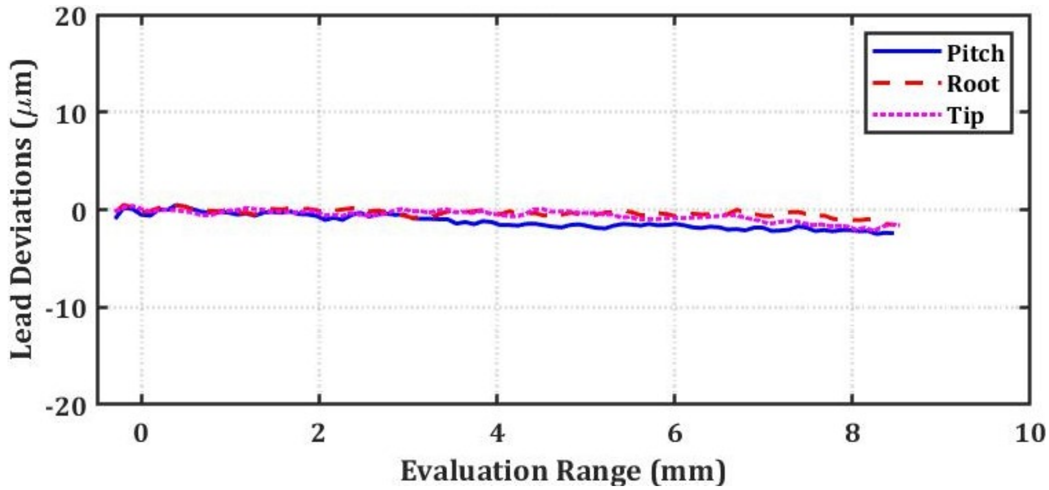


Figure 5.20: Best Total Lead Deviation Data - External Helical Gear, Tooth 1, Left Flank

Table 5.21: External Helical Gear, Tooth 1, Left Flank, Lead

| Parameter | Root | Pitch | Tip |
|---|---|---|---|
| $f_{H\beta}$ | -0.746 | -2.975 | -1.211 |
| $f_{f\beta}$ | 1.085 | 1.183 | 1.237 |
| $F_{\beta}$ | 1.326 | 2.659 | 1.867 |

## 5.8 General Observations

A limitation of the process is the necessary size of the tools. Due to the proximity of adjacent teeth, a small diameter tool is necessary, which makes it very difficult to get a stable cut using a long tool. Thus, the facewidth of the gear is limited. The smaller the teeth, the smaller the facewidth.

## 5.9 Surface Finish

Some basic surface finish measurements were performed on two flanks of an external spur gear. Mr. Greg Caskey of the UNC Charlotte Center for Precision Metrology performed these measurements using a Mahr Profilometer. The machining parameters for the finish pass were as given in Table 5.2.

Measurements were performed on one left flank and one right flank, in both the axial (helix) and radial (profile) directions. $R_a$ (total average deviations from a straight line), $R_q$ (Root Mean Square), and $R_z$ (ten point average) values are given in Equations 5.1 through 5.3, from [29].

$$R_a = \frac{1}{l_r}\int_0^{l_r} |z(x)|dx = \frac{|Z_1| + |Z_2| + ... + |Z_N|}{N} \tag{5.1}$$

$$R_q = \sqrt{\frac{1}{l_r}\int_0^{l_r} z(x)^2 dx} = \sqrt{\frac{Z_1^2 + Z_2^2 + ... + Z_N^2}{N}} \tag{5.2}$$

$$R_z = \frac{(Z_{p1} + Z_{p2} + Z_{p3} + Z_{p4} + Z_{p5}) - (Z_{v1} + Z_{v2} + Z_{v3} + Z_{v4} + Z_{v5})}{5} \tag{5.3}$$

The measurement data is shown below. Before filtering, a first order fit was removed from the helix direction measurements, and a second order fit was removed from the profile direction measurements. An evaluation length of 4 mm was used, with cutoff wavelength of 0.8 mm, according to [32].

Note a large spike in Figure 5.21 at around 3.6 mm. This is due to a scratch on the surface of the gear, and is not a product of the original machining process.

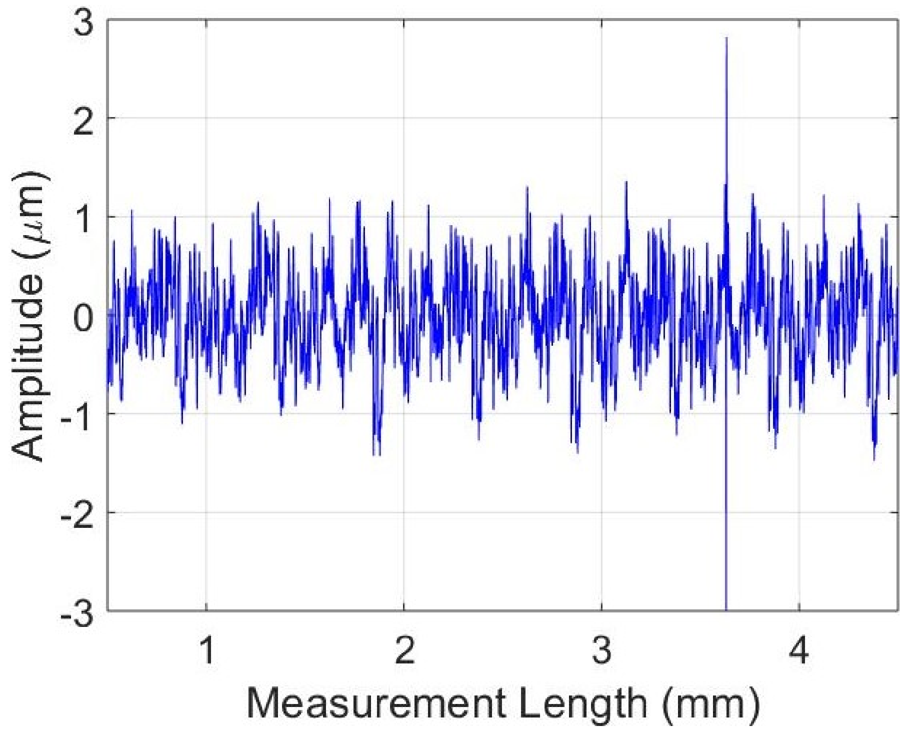


Figure 5.21: Surface Roughness of Flank 1, Helix Direction

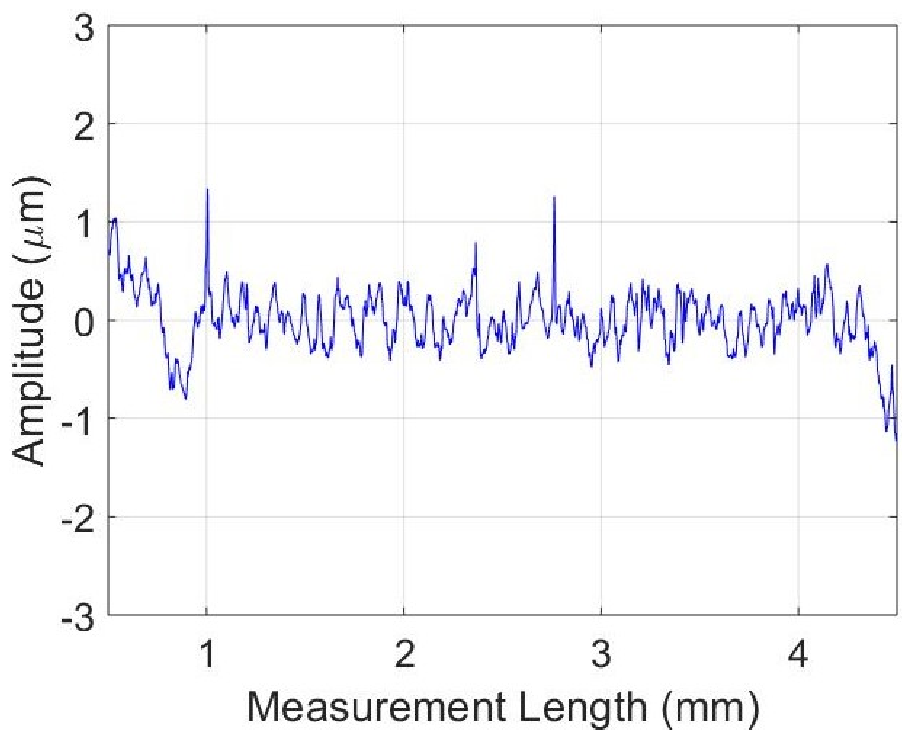


Figure 5.22: Surface Roughness of Flank 1, Profile Direction

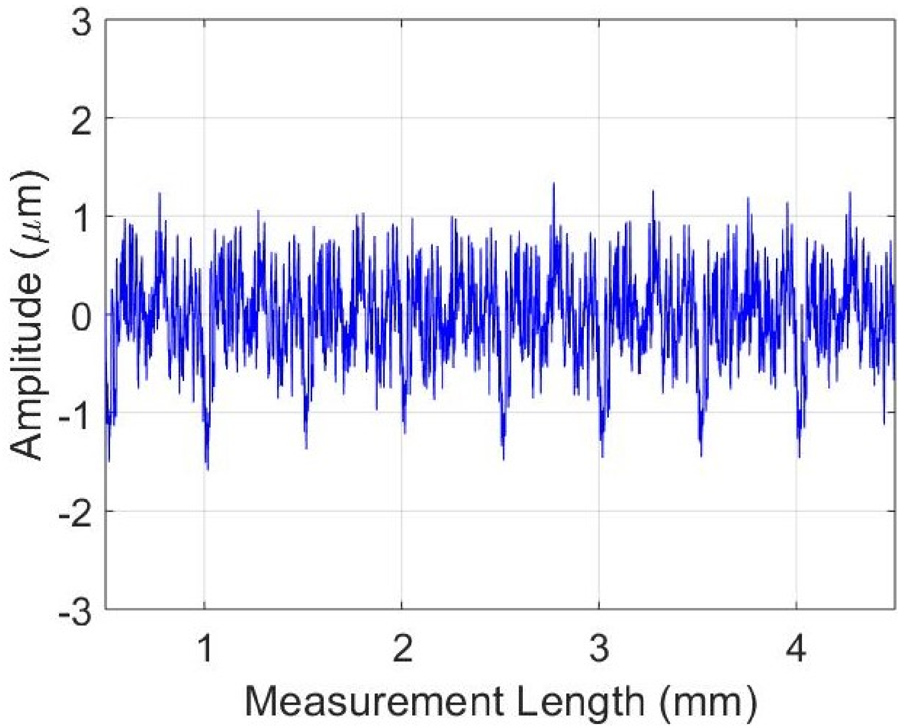


Figure 5.23: Surface Roughness of Flank 2, Helix Direction

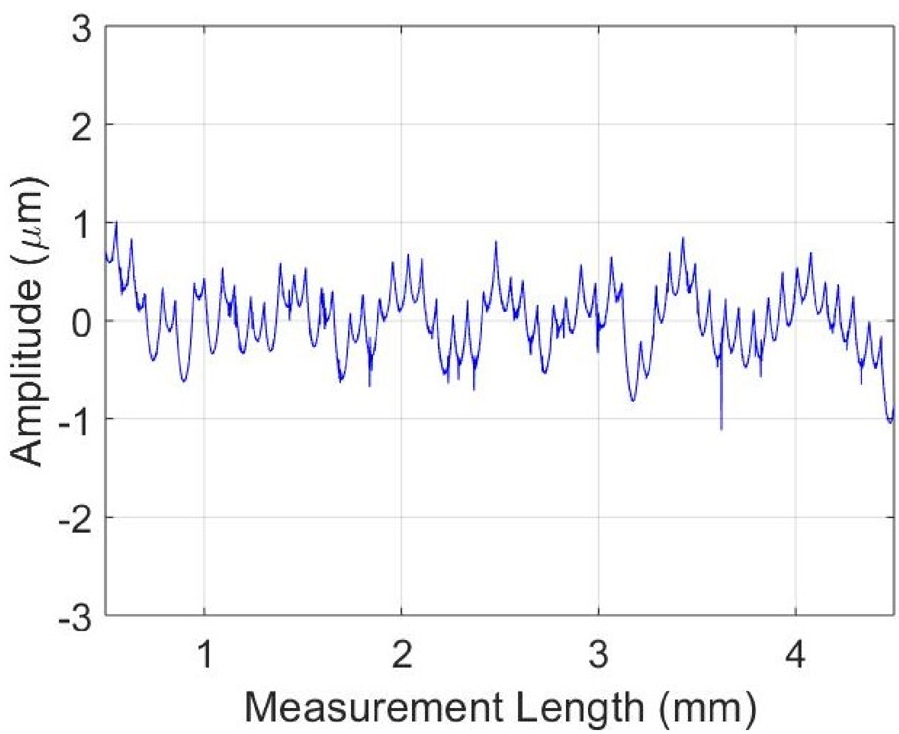


Figure 5.24: Surface Roughness of Flank 2, Profile Direction

The parameters as calculated according to Equations 5.1 through 5.3 are given in

Table 5.22, below.

Table 5.22: Surface Roughness Parameters

| Parameter | Flank 1: Helix Direction | Flank 1: Profile Direction | Flank 2: Helix Direction | Flank 2: Profile Direction |
|---|---|---|---|---|
| $R_a$ | 0.377 $\mu m$ | 0.229 $\mu m$ | 0.365 $\mu m$ | 0.268 $\mu m$ |
| $R_q$ | 0.478 $\mu m$ | 0.305 $\mu m$ | 0.460 $\mu m$ | 0.334 $\mu m$ |
| $R_z$ | 5.016 $\mu m$ | 2.532 $\mu m$ | 2.813 $\mu m$ | 2.064 $\mu m$ |

This surface finish is more than adequate for a pre-ground gear, and was standard for all machined gears. Thus, no further refinement of surface finish for the milling operation was deemed necessary.

## 5.10 Limitations

There are some limitations on the size of the gears that can be machined using the methods described in this thesis. First, the diameter of the tool must be small enough that it can fit all the way down in the root of the gear teeth. This necessitates lower feed rates, and therefore lower material removal rates. The process is therefore slow. The time to machine these gears varied from roughly 1 hour to 3 hours. If a faster, less precise method of roughing the gears was developed and generation milling only used for finishing the gears, then significant time savings could be achieved.

Second, the small diameter of the cutting tool renders it significantly difficult to achieve stable cutting conditions, and the longer the tool is, the more this problem is evident. Thus, the face width of the gears to be cut is limited to the longest length of the tool, the diameter of which is defined by the gear tooth spacing, at which stable cutting conditions can be achieved.

## 5.11 Conclusions

This thesis presents a method for machining internal and external cylindrical involute gears. The process is slow, and the gears to be machined are limited to certain size constraints. Within those constraints however, the results are good, demonstrating that high quality gears can be machined using conventional five-axis milling machines and the methods presented herein.

# REFERENCES


[1] D. P. Townsend, ed., *Dudley's Gear Handbook.* McGraw-Hill, Inc., New York, NY, 1962.

[2] G. B. Grant, *A Treatise on Gear Wheels.* Boston: Grant Gear Works, 8 ed., 1899.

[3] G. Goch, "Gear metrology," *CIRP Annals - Manufacturing Technology*, vol. 52, no. 2, pp. 659–695, 2003.

[4] G. Goch, W. Knapp, and F. Haertig, "Precision engineering for wind energy systems," *CIRP Annals - Manufacturing Technology*, vol. 61, no. 2, pp. 611–634, 2012.

[5] K.-D. Bouzakis, E. Lili, N. Michailidis, and O. Friderikos, "Manufacturing of cylindrical gears by generating cutting processes: A critical synthesis of analysis methods," *CIRP Annals - Manufacturing Technology*, vol. 57, no. 2, pp. 676–696, 2008.

[6] A. Alvarez, L. Lacalle, Lopez de, A. Olaiz, and A. Rivero, "Large spiral bevel gears on universal 5-axis milling machines: A complete process," *Procedia Engineering*, vol. 132, pp. 397–404, 2015.

[7] D. Stoebener, K. Luebke, *et al.*, "An approach for 5-axes gear production with a slot-milling cutter," in *The Proceedings of MTTRF 2012 Meeting*, pp. 115–120, 2012.

[8] I. Zeid, *CAD/CAM Theory and Practice.* McGraw-Hill, Inc., 1991.

[9] Technical Committee ISO/TC 60, Gears, Subcommittee SC 1, Nomenclature and Wormgearing, *ISO 21771 - International Standard - Gears - Cylindrical Involute Gears and Gear Pairs - Concepts and Geomoetry.* International Standards Organization, 2007.

[10] Technical Committee ISO/TC 60, Gears, *ANSI/AGMA ISO 1328-1-B14 - Cylindrical Gears - ISO System of Flank Tolerance Classification - Part 1: Definitions and Allowable Values of Deviations Relevant to Flanks of Gear Teeth.* American Gear Manufacturers Association, 2014.

[11] Q. Chen, Y. Guo, L. Zhu, and H. Zou, "Study on parameters fitting of archimedes helicoid surface," *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 7544, pp. 7544–3B, August 2010.

[12] K. Ni, *Areal Gear Metrology with Modified Flanks.* PhD thesis, University of North Carolina at Charlotte, 2017.

[13] J. Hedlund and A. Lehtovaara, "A parameterized numerical method for generating discrete helical gear tooth surface allowing non-standard geometry," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 2008.

[14] J. Jiang and Z. Fang, "Design and analysis for topologically modified helical gear finished by gear-hobbing," *Journal of Central South University (Science and Technology*, vol. 47, no. 11, pp. 3677–3684, 2016.

[15] M. Nemcek and Z. Dejl, "Geometric calculations of the chamfered tip and the protuberance undercut of a tooth profile," *Proceedings of the ASME Design Engineering Technical Conference, International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC/-CIE*, vol. 8, pp. 203–211, August 2011.

[16] H. Yong, X. Mingjun, Q. Xing, S. Pengfei, S. Jingan, and P. Fuhua, "An improved design of inserted tooth finishing hob for involute gears," *Journal of Mechanical Science and Technology*, vol. 28, pp. 3203–3208, August 2014.

[17] V.-T. Tran, R.-H. Hsu, and C.-B. Tsay, "A methodology for longitudinal tooth flank crowning of the helical gear on a cnc honing machine," *Advanced Materials Research*, vol. 1091, pp. 53–62, 2015.

[18] Y. Shen, X. Liu, D. Li, and Z. Li, "A method for grinding face gear of double crowned tooth geometry on a multi-axis cnc machine," *Mechanism and Machine Theory*, vol. 121, pp. 460–474, March 2018.

[19] A. Antoniadis, "Gear skiving cad simulation approach," *Computer Aided Design*, vol. 44, pp. 611–616, July 2012.

[20] A. Kawalec, "Numerical modeling and robust parametric estimation of surfaces useful for representation of tooth flanks," *Computer Assisted Mechanics and Engineering Sciences*, vol. 7, pp. 571–588, May 1999.

[21] Y. I. Lai-Hua, "Numerical simulation of gear shaping for planetary gear based on the masta software," *Applied Mechanics and Materials*, vol. 404, pp. 307–311, 2013.

[22] H. J. Stadtfeld, *Advanced Bevel Gear Technology for the New Millennium.* Rochester: The Gleason Works, Rochester, NY, 2000.

[23] H. J. Stadtfeld, *Handbook of Bevel and Hypoid Gears: Calculation, Manufacturing, and Optimization.* Rochester: Rochester Institute of Technology, Rochester, NY, 1993.

[24] H. J. Stadtfeld, *Bevel Gear Technology.* The Gleason Works, Rochester, NY, 1995.

[25] L. Li, L. Zhang, B. Yu, K. Wang, and F. Liu, "An efficient spur gear shaping method based on homogenizing cutting area through variational circular feed rate," *Proceedings of the Institution of Mechanical Engineers, Part B (Journal of Engineering Manufacture)*, vol. 231, pp. 1587–1598, July 2017.

[26] D. Vasilis, V. Nectarios, and A. Aristomenis, "Advanced computer aided design simulation of gear hobbing by means of three-dimensional kinematics modeling," *Journal of Manufacturing Science and Engineering*, vol. 129, pp. 911–918, October 2007.

[27] J. Stewart, *Essential Calculus, Early Transcendentals*. Brooks/Cole, Belmont, CA, 2007.

[28] A. Guenther, *Flaechenhafte Beschreibung und Ausrichtung von Zylinderraedern mit Evolventenprofil*. Diploma thesis, Universitaet Ulm, Abteilung Mess-, Regel- und Mikrotechnik, 1996.

[29] D. Whitehouse, *Surfaces and Their Measurement*. Taylor and Francis Books, Inc., New York, NY, 2002.

[30] "Gear hobbing." https://upload.wikimedia.org/wikipedia/commons/thumb/d/d2/W%C3%A4lzfr%C3%A4ser_und_gefr%C3%A4ste_Verzahnung.JPG/1280px-W%C3%A4lzfr%C3%A4ser_und_gefr%C3%A4ste_Verzahnung.JPG. Accessed: 2018-12-04.

[31] "Gear shaping." https://www.manufacturingguide.com/sites/default/files/styles/illustration/public/illustrations/gear_shaping_1163.png?itok=kBVNmgq0. Accessed: 2018-12-04.

[32] I. ISO/TC 57, Metrology and Properties of Surfaces, Subcommittee SC 1, Geometrical Parameters - Instruments and Procedures for Measurement of Surface Roughness and Waviness, ISO/TC3, Limits and Fits, *ISO 4288 - Geometrical Product Specifications (GPS) - Surface Texture: Profile Method - Rules and Procedures for the ASsessment of Surface Texture*. International Standards Organization, 1996.

# APPENDIX A: MATLAB CODE

The following sections provide the individual functions written in MATLAB used to generate the G code for gear cutting. All inputs are drawn from a Microsoft Excel spreadsheet, and the various 'inputs' functions reference individual cells in this spreadsheet.

## A.1 Main Function - MASTER.m

```
function MASTER
% Jesse Groover
% 28 November 2018
% This is the master function for generating G code for
% cutting internal and external cylindrical involute gears,
% using generation motion.
% Written as a master's project by Jesse Groover at the
% University of North Carolina at Charlotte, this program
% generates G code for a Mori Seiki NMV5000DCG five-axis
% milling machine, with a Fanuc MSX-711III control. The
% workpiece is to be centered on the rotary table (C axis),
% and a standard cylindrical endmill is used.
%
% Inputs are pulled from a Microsoft Excel Spreadsheet, the
% name of which is stored as a string in  variable "filename".

close all; clear;

% Name of Microsoft Excel Spreadsheet with all input
% information
```

```
filename = 'ParameterTable.xlsm';

% Pull in Machine Parameters
pagename_machine = 11;
[MachParams,Smax] = inputs_machine(filename,pagename_machine);

% Pull in General File Information
pagename_general = 1;
[z,txtfilename,auth,date,Onum,comment,desc,WCS,WCS_offsets]...
    = inputs_general(filename,pagename_general);

% Pull in Geometry Information
pagenum_geom = 1;
[mt,alpha_t_in,beta_in,b,x,k] = inputs_Geometry(filename,...
    pagenum_geom);

% Calculate Geometry Parameters
[Rb,Rp,rollss,ImpAngles,eta_b,psi_b,dtoolmax,Rinner,...
    Router,beta_b] = Geometry(mt,z,alpha_t_in,beta_in,x,k,1);

fprintf('Max Tool Diameter: %.3f mm.\n',dtoolmax);
fprintf('Pitch Radius: %.3f mm\n',Rp);
fprintf('Base Radius: %.3f mm\n',Rb);

% Determine operation list
if sign(z) == 1
    [~,operations,~] = xlsread(filename,pagename_general,...
```

```
        'H3:H8');
else
    [~,operations,~] = xlsread(filename,pagename_general,...
        'H11:H15');
end
% if operations == []
%     fprintf('No operations selected.\n');
% end

% Initialize NC Program File
name = fopen(string(txtfilename),'w');
header(name,auth,date,Onum,comment,desc,WCS);

% Initalize N number
Nnum = 1000;

% Logic Structure - Operations
% EXTERNAL GEARS
if sign(z) == 1
    %% Facing Operation
    if string(operations(1)) == 'Yes'
        fprintf('External Face\n');
        pagenum = 3; % Page of inputs in MS Excel spreadsheet
        [R_outer,Depth_Total,Depth_Inc,Ret,helixstep,...
            ToolName,ToolNumber,ToolDia,ToolLength,Tooln,...
            ToolRI,Toolv,Toolft,coolant]...
            = inputs_face(filename,pagenum);
```

```
    [S,Flin] = cutconditions(ToolDia,Toolv,Tooln,...
        Toolft,Smax); % Generate cutting feeds and speeds
    operation_pocket_cyl(R_outer,Depth_Total,...
        Depth_Inc,Ret,helixstep,ToolName,ToolNumber,...
        ToolDia,ToolLength,ToolRI,S,Flin,coolant,...
        Nnum,name);
    Nnum = Nnum + 1000; % Increment N number for next
        %operation
end

%% Outer Cylinder Operation
if string(operations(2)) == 'Yes'
    fprintf('External Outer Cylinder\n');
    pagenum = 5; % Page of inputs in MS Excel spreadsheet
    [R_final,R_init,Depth_Total,Depth_Inc,Ret,...
        ToolName,ToolNumber,ToolDia,ToolLength,Tooln,...
        ToolRI,Toolv,Toolft,coolant]...
        = inputs_OuterCyl(filename, pagenum);
    % Set defaults if certain parameters are left blank
    if R_final == 0
        R_final = Router;
    end

    [S,Flin] = cutconditions(ToolDia,Toolv,Tooln,...
        Toolft,Smax); % Generate cutting feeds and speeds
    operation_OuterCyl(R_final,R_init,Depth_Total,...
        Depth_Inc,Ret,ToolName,ToolNumber,ToolDia,...
```

```
        ToolLength,ToolRI,S,Flin,coolant,Nnum,name)
    Nnum = Nnum + 1000; % Increment N number for next
        % operation
end

%% Center bore Operation
if string(operations(3)) == 'Yes'
    fprintf('External␣Center␣Bore\n');
    pagenum = 4; % Page of inputs in MS Excel spreadsheet
    [R_outer,Depth_Total,Depth_Inc,Ret,helixstep,...
        ToolName,ToolNumber,ToolDia,ToolLength,Tooln,...
        ToolRI,Toolv,Toolft,coolant]...
        = inputs_bore_hog(filename,pagenum);
    [S,Flin] = cutconditions(ToolDia,Toolv,Tooln,...
        Toolft,Smax); % Generate cutting feeds and speeds
    operation_pocket_cyl(R_outer,Depth_Total,Depth_Inc,...
        Ret,helixstep,ToolName,ToolNumber,ToolDia,...
        ToolLength,ToolRI,S,Flin,coolant,Nnum,name);
    Nnum = Nnum + 1000; % Increment N number for next
        % operation
end

%% Slotting Operation
if string(operations(4)) == 'Yes'
    fprintf('External␣Slots\n');
    pagenum = 7; % Page of inputs in MS Excel spreadsheet
    [R_outer,R_inner,R_Step,ext,Ret,ToolName,...
```

```
    ToolNumber, ToolDia, ToolLength, Tooln, ToolRI, ...
    Toolv, Toolft, coolant]...
    = inputs_slots(filename,pagenum); % Slotting Inputs
[S, Flin] = cutconditions(ToolDia, Toolv, Tooln, ...
    Toolft,Smax); % Generate cutting feeds and speeds
% Set defaults
if R_inner == 0
    R_inner = Rinner;
end
if R_outer == 0
    R_outer = Router;
end

% Generate Vectors
[Pslot, Tslot, Rslot, nslot, Qslot, Wslot, Fcorrslot, ...
    fvecslot] = operation_slots_radial(z,Rb,beta_b, ...
    eta_b,b,R_outer,R_inner,R_Step,ext,ToolDia, ...
    ToolRI,Ret);
% Coordinate Transformation
[Pslotnew,Qslotnew,nslotnew,Tslotnew,Rslotnew]...
    = BCRotHTM(Pslot,Qslot,nslot,Tslot,Rslot, ...
    fvecslot,WCS_offsets,beta_b,MachParams,'Tool');
% Post
post_vectorial(Qslotnew,Rslotnew,Wslot,Fcorrslot, ...
    Nnum,name,'Precutting␣Slots',ToolName, ...
    ToolNumber,S,Flin,10000,coolant);
```

```
% TEST PLOTS
testplot = 0; % Turn on (1) or off (0)
if testplot == 1
    xline = [0 120*cos(-(pi/abs(z) - eta_b))];
    yline = [0 120*sin(-(pi/abs(z) - eta_b))];
    zline = [0 0];

    figure;
    plot3(Pslot(1,:),Pslot(2,:),Pslot(3,:),'.k');
    hold all;
    plot3(Qslot(1,:),Qslot(2,:),Qslot(3,:),'.r');
    quiver3(Pslot(1,:),Pslot(2,:),Pslot(3,:),...
        nslot(1,:),nslot(2,:),nslot(3,:),0,'g');
    quiver3(Qslot(1,:),Qslot(2,:),Qslot(3,:),...
        Tslot(1,:),Tslot(2,:),Tslot(3,:),0,'r');
    plot3(xline,yline,zline);
    xlabel('X'); ylabel('Y'); zlabel('Z');
    axis equal;
    hold off

    figure;
    plot3(Pslotnew(1,:),Pslotnew(2,:),Pslotnew(3,:),...
        '.k'); hold all;
    plot3(Qslotnew(1,:),Qslotnew(2,:),Qslotnew(3,:),...
        '.r');
    quiver3(Pslotnew(1,:),Pslotnew(2,:),...
        Pslotnew(3,:),nslotnew(1,:),nslotnew(2,:),...
```

```
            nslotnew(3,:),0,'g');
        quiver3(Qslotnew(1,:),Qslotnew(2,:),...
            Qslotnew(3,:),Tslotnew(1,:),Tslotnew(2,:),...
            Tslotnew(3,:),0,'r');
        plot3(xline,yline,zline);
        xlabel('X'); ylabel('Y'); zlabel('Z');
        axis equal;
        hold off

    end

    Nnum = Nnum + 1000; % Increment N number for next
        % operation
end

%% Tooth Roughing Operation
if string(operations(5)) == 'Yes'
    fprintf('External␣Roughing\n');
    pagenum = 8; % Page of inputs in MS Excel spreadsheet
    % Inputs
    [b_total,b_step,Ret,stockclearance_rough,safedist1,...
        safedist2,Flimit,updn,increment,incsize,...
        ToolName,ToolNumber,ToolDia,ToolLength,Tooln,...
        ToolRI,Toolv,Toolft,coolant]...
        = inputs_toothroughing(filename,pagenum);
    if b_total > ToolLength
        fprintf('The␣tool␣cutting␣length␣is␣not␣long');
```

```
    fprintf(' enough.  The cut will only extend to');
    fprintf(' a depth of %.3f mm.\r\n',ToolLength);
    b_total = ToolLength;
end
% Preliminary Stuff
[S,Flin] = cutconditions(ToolDia,Toolv,Tooln,...
    Toolft,Smax);
[Rsafe1,Rsafe2,zlev,offset_N] = prelimcalcs(z,...
    Rinner,Router,ToolDia,safedist1,safedist2,...
    b_total,b_step);
th = thicknessvec(z,Rb,Router,eta_b,beta_b,...
    ToolDia,ToolRI,stockclearance_rough); % Generate
        % a vector of the stock offset distance values

% Generate Vectors
[P,T,R,N,Q,W,Fcorr,fvec] = VecGen_Complete(z,th,...
    zlev,ToolDia,beta_b,updn,rollss,Rb,eta_b,...
    offset_N,ImpAngles,Rsafe1,Rsafe2,increment,...
    incsize,Ret);
% Convert
[Pnew,Qnew,nnew,Tnew,Rnew] = BCRotHTM(P,Q,N,T,R,...
    fvec,WCS_offsets,beta_b,MachParams,'NA00');
% Post
post_vectorial(Qnew,Rnew,W,Fcorr,Nnum,name,...
    'Tooth Roughing',ToolName,ToolNumber,S,Flin,...
    Flimit,coolant);
```

```
% TEST PLOTS
testplot = 0; % Turn on (1) or off (0)
if testplot == 1
    theta = 0:0.1:1.1*2*pi;
    basex = Rb*cos(theta);
    basey = Rb*sin(theta);
    basez = zeros(1,numel(theta));

    figure;
    plot3(basex,basey,basez,'-k'); hold all;
    % Nominal Involute Points
    plot3(P(1,:),P(2,:),P(3,:),'.k');
    % Tool Center Points
    plot3(Q(1,:),Q(2,:),Q(3,:),'.r');
    quiver3(P(1,:),P(2,:),P(3,:),N(1,:),N(2,:),...
        N(3,:),0,'b'); % Scaled Unit Normal Vectors
    quiver3(Q(1,:),Q(2,:),Q(3,:),T(1,:),T(2,:),...
        T(3,:),'r'); % Tool Orientation Vectors
    legend({'Base Circle','Nominal Involute Points',...
        'Unit Normal Vector',...
        'Tool Orientation Vector'});
    axis equal;
    axis([70 90 -10 10 -10 10]);
    grid on;
    hold off;

    figure;
```

```
plot3(basex,basey,basez,'-k'); hold all;
% Nominal Involute Points
plot3(P(1,:),P(2,:),P(3,:),'.k');
% Tool Center Points
plot3(Q(1,:),Q(2,:),Q(3,:),'.r');
% Nominal Points, Rotated
plot3(Pnew(1,:),Pnew(2,:),Pnew(3,:),'.k');
% Tool Center Points, Rotated
plot3(Qnew(1,:),Qnew(2,:),Qnew(3,:),'.r');
% Scaled Unit Normal Vectors
quiver3(P(1,:),P(2,:),P(3,:),N(1,:),N(2,:),...
    N(3,:),0,'b');
% Tool Orientation Vectors
quiver3(Pnew(1,:),Pnew(2,:),Pnew(3,:),...
    Tnew(1,:),Tnew(2,:),Tnew(3,:),'r');
% Unit Normal Vectors
quiver3(Pnew(1,:),Pnew(2,:),Pnew(3,:),...
    nnew(1,:),nnew(2,:),nnew(3,:),0,'b');
legend({'Base Circle','Nominal Involute Points',...
    'Unit Normal Vector','Tool Orientation Vector'});
axis equal;
axis([70 90 -10 10 -10 10]);
grid on;
hold off;

figure;
plot3(basex,basey,basez,'-k'); hold all;
```

```
            axis equal;
            for i = 1:numel(P(1,:))
                % Nominal Involute Points
                plot3(P(1,i),P(2,i),P(3,i),'.k');
                % Tool Center Points
                plot3(Q(1,i),Q(2,i),Q(3,i),'.r');
                % Tool Center Points, Rotated
                plot3(Qnew(1,i),Qnew(2,i),Qnew(3,i),'.r');

                pause;
            end

            hold off

        end

        Nnum = Nnum + 1000; % Increment N number for next
            % operation
    end

    %% Tooth Finishing Operation
    if string(operations(6)) == 'Yes'
        fprintf('External Finishing\n');
        pagenum = 9; % Page of inputs in MS Excel spreadsheet
        % Inputs
        [b_total,b_step,Ret,stockclearance_finish,...
            safedist1,safedist2,Flimit,updn,increment,...
```

```
    incsize ,ToolName,ToolNumber,ToolDia,ToolLength,...
    Tooln,ToolRI,Toolv,Toolft,coolant]...
    = inputs_toothfinishing(filename,pagenum);
if b_total > ToolLength
    fprintf('The tool cutting length is not long');
    fprintf(' enough.  The cut will only extend to');
    fprintf(' a depth of %.3f mm.\r\n',ToolLength);
    b_total = ToolLength;
end
% Preliminary Stuff
[S,Flin] = cutconditions(ToolDia,Toolv,Tooln,...
    Toolft,Smax);
[Rsafe1,Rsafe2,zlev,offset_N] = prelimcalcs(z,...
    Rinner,Router,ToolDia,safedist1,safedist2,...
    b_total,b_step);
th = stockclearance_finish;

% Generate Vectors
[P,T,R,N,Q,W,Fcorr,fvec] = VecGen_Complete(z,th,...
    zlev,ToolDia,beta_b,updn,rollss,Rb,eta_b,...
    offset_N,ImpAngles,Rsafe1,Rsafe2,increment,...
    incsize,Ret);
% Convert
[Pnew,Qnew,nnew,Tnew,Rnew] = BCRotHTM(P,Q,N,T,R,...
    fvec,WCS_offsets,beta_b,MachParams,'NA00');
% Post
post_vectorial(Qnew,Rnew,W,Fcorr,Nnum,name,...
```

```
        'Tooth Finishing',ToolName,ToolNumber,S,Flin,...
        Flimit,coolant);

    % TEST PLOTS
    testplot = 1; % Turn on (1) or off (0)
    if testplot == 1
        theta = 0:0.1:1.1*2*pi;
        basex = Rb*cos(theta);
        basey = Rb*sin(theta);
        basez = zeros(1,numel(theta));

%           figure;
%           plot3(basex,basey,basez,'-k'); hold all;
        % Nominal Involute Points
%           plot3(P(1,:),P(2,:),P(3,:),'.k');
        % Tool Center Points
%           plot3(Q(1,:),Q(2,:),Q(3,:),'.r');
        % Unit Normal Vectors
%           quiver3(P(1,:),P(2,:),P(3,:),N(1,:),N(2,:),...
%               N(3,:),0,'b');
        % Tool Orientation Vectors
%           quiver3(Q(1,:),Q(2,:),Q(3,:),T(1,:),T(2,:),...
%               T(3,:),'r');
%           legend({'Base Circle','Nominal Involute Points',...
%               'Unit Normal Vector','Tool Orientation Vector',...
%               'Tool Velocity Vector'});
%           axis equal;
```

```
%           axis([70 90 -10 10 -10 10]);
%           grid on;
%           hold off;

          figure;
          set(0,'defaultAxesFontSize',16);
          plot3(basex,basey,basez,'-k'); hold all;
          % Nominal Involute Points
%           plot3(P(1,:),P(2,:),P(3,:),'.k');
          % Tool Center Points
%           plot3(Q(1,:),Q(2,:),Q(3,:),'.r');
          % Nominal Points, Rotated
%           plot3(Pnew(1,:),Pnew(2,:),Pnew(3,:),'.k');
          % Tool Center Points, Rotated
%           plot3(Qnew(1,:),Qnew(2,:),Qnew(3,:),'.r');
          % Unit Normal Vectors
%           quiver3(P(1,:),P(2,:),P(3,:),N(1,:),N(2,:),...
%               N(3,:),0,'b');
%           Tool Orientation Vectors
          quiver3(Pnew(1,:),Pnew(2,:),Pnew(3,:),...
              Tnew(1,:),Tnew(2,:),Tnew(3,:),4,'r');
          % Unit Normal Vectors, Rotated
          quiver3(Pnew(1,:),Pnew(2,:),Pnew(3,:)...
              ,nnew(1,:),nnew(2,:),nnew(3,:),0,'g');
%           legend({'Base Circle','Nominal Involute Points',...
%               'Unit Normal Vector','Tool Orientation Vector',...
%               'Tool Velocity Vector'});
```

```
            axis equal;
            xlabel('X'); ylabel('Y'); zlabel('Z');
%             axis([70 90 -10 10 -10 10]);
            grid on;
            hold off;

%             figure;
%             plot3(basex,basey,basez,'-k'); hold all;
%             axis equal;
%             for i = 1:numel(P(1,:))
%                 % Nominal Involute Points
%                 plot3(P(1,i),P(2,i),P(3,i),'.k');
%                 % Tool Center Points
%                 plot3(Q(1,i),Q(2,i),Q(3,i),'.r');
%                 % Tool Center Points, Rotated
%                 plot3(Qnew(1,i),Qnew(2,i),Qnew(3,i),'.r');
%
%                 pause;
%             end
%             hold off
        end

        Nnum = Nnum + 1000; % Increment N number for next
            % operation
    end

% INTERNAL GEARS
```

```
elseif sign(z) == -1
    %% Facing Operation
    if string(operations(1)) == 'Yes'
        fprintf('Internal Face\n');
        pagenum = 3; % Page of inputs in MS Excel spreadsheet
        [R_outer,Depth_Total,Depth_Inc,Ret,helixstep ,...
            ToolName,ToolNumber,ToolDia,ToolLength,Tooln,...
            ToolRI,Toolv,Toolft,coolant]...
            = inputs_face(filename,pagenum);
        [S,Flin] = cutconditions(ToolDia,Toolv,Tooln,...
            Toolft,Smax);
        operation_pocket_cyl(R_outer,Depth_Total,Depth_Inc,...
            Ret,helixstep,ToolName,ToolNumber,ToolDia,...
            ToolLength,ToolRI,S,Flin,coolant,Nnum,name);
        Nnum = Nnum + 1000; % Increment N number for next
            % operation
    end

    %% Hog out Center
    if string(operations(2)) == 'Yes'
        fprintf('Internal Center Hogging\n');
        pagenum = 4; % Page of inputs in MS Excel spreadsheet
        [R_outer,Depth_Total,Depth_Inc,Ret,helixstep ,...
            ToolName,ToolNumber,ToolDia,ToolLength,Tooln,...
            ToolRI,Toolv,Toolft,coolant]...
            = inputs_bore_hog(filename,pagenum);
        [S,Flin] = cutconditions(ToolDia,Toolv,Tooln,...
```

```
            Toolft ,Smax);
        operation_pocket_cyl(R_outer,Depth_Total,Depth_Inc,...
            Ret,helixstep,ToolName,ToolNumber,ToolDia,...
            ToolLength,ToolRI,S,Flin,coolant,Nnum,name);
        Nnum = Nnum + 1000; % Increment N number for next
            % operation
end

%% Tooth Roughing Operation
if string(operations(3)) == 'Yes'
    fprintf('Internal Roughing\n');
    pagenum = 8; % Page of inputs in MS Excel spreadsheet

    % Inputs
    [b_total,b_step,Ret,stockclearance_rough,safedist1,...
        safedist2,Flimit,updn,increment,incsize,...
        ToolName,ToolNumber,ToolDia,ToolLength,Tooln,...
        ToolRI,Toolv,Toolft,coolant]...
        = inputs_toothroughing(filename,pagenum);
    if b_total > ToolLength
        fprintf('The tool cutting length is not long');
        fprintf(' enough.  The cut will only extend to');
        fprintf(' a depth of %.3f mm.\r\n',ToolLength);
        b_total = ToolLength;
    end
    % Preliminary Stuff
    [S,Flin] = cutconditions(ToolDia,Toolv,Tooln,...
```

```
    Toolft ,Smax);
[Rsafe1 ,Rsafe2 ,zlev ,offset_N] = prelimcalcs(z,...
    Rinner ,Router ,ToolDia ,safedist1 ,safedist2 ,...
    b_total ,b_step);
% Generate a vector of the stock offset distance
    % values
th = thicknessvec(z,Rb,Router ,eta_b ,beta_b ,ToolDia ,...
    ToolRI ,stockclearance_rough);

% Generate Vectors
[P,T,R,N,Q,W,Fcorr ,fvec] = VecGen_Complete(z,th ,...
    zlev ,ToolDia ,beta_b ,updn ,rollss ,Rb,eta_b ,...
    offset_N ,ImpAngles ,Rsafe1 ,Rsafe2 ,increment ,...
    incsize ,Ret);
% Convert
[Pnew,Qnew,nnew,Tnew,Rnew] = BCRotHTM(P,Q,N,T,R,...
    fvec ,WCS_offsets ,beta_b ,MachParams ,'NA00');
% Post
post_vectorial(Qnew,Rnew,W,Fcorr ,Nnum,name ,...
    'Tooth␣Roughing' ,ToolName ,ToolNumber ,S,Flin ,...
    Flimit ,coolant);

% TEST PLOTS
testplot = 0; % Turn on (1) or off (0)
if testplot == 1
    theta = 0:0.1:1.1*2*pi;
    basex = Rb*cos(theta);
```

```
basey = Rb*sin(theta);
basez = zeros(1,numel(theta));

figure;
plot3(basex,basey,basez,'-k'); hold all;
% Nominal Involute Points
plot3(P(1,:),P(2,:),P(3,:),'.k');
% Tool Center Points
plot3(Q(1,:),Q(2,:),Q(3,:),'.r');
% Scaled Unit Normal Vectors
quiver3(P(1,:),P(2,:),P(3,:),N(1,:),N(2,:),...
    N(3,:),0,'b');
% Tool Orientation Vectors
quiver3(Q(1,:),Q(2,:),Q(3,:),T(1,:),T(2,:),...
    T(3,:),'r');
legend({'Base Circle','Nominal Involute Points',...
    'Unit Normal Vector','Tool Orientation Vector'});
axis equal;
axis([70 90 -10 10 -10 10]);
grid on;
hold off;

figure;
plot3(basex,basey,basez,'-k'); hold all;
% Nominal Involute Points
plot3(P(1,:),P(2,:),P(3,:),'.k');
% Tool Center Points
```

```
plot3(Q(1,:),Q(2,:),Q(3,:),'.r');
% Nominal Points, Rotated
plot3(Pnew(1,:),Pnew(2,:),Pnew(3,:),'.k');
% Tool Center Points, Rotated
plot3(Qnew(1,:),Qnew(2,:),Qnew(3,:),'.r');
% Scaled Unit Normal Vectors
quiver3(P(1,:),P(2,:),P(3,:),N(1,:),N(2,:),...
    N(3,:),0,'b');
% Tool Orientation Vectors
quiver3(Pnew(1,:),Pnew(2,:),Pnew(3,:),Tnew(1,:),...
    Tnew(2,:),Tnew(3,:),'r');
% Unit Normal Vectors
quiver3(Pnew(1,:),Pnew(2,:),Pnew(3,:),nnew(1,:),...
    nnew(2,:),nnew(3,:),0,'b');
legend({'Base Circle','Nominal Involute Points',...
    'Unit Normal Vector','Tool Orientation Vector'});
axis equal;
axis([70 90 -10 10 -10 10]);
grid on;
hold off;

figure;
plot3(basex,basey,basez,'-k'); hold all;
axis equal;
for i = 1:numel(P(1,:))
    % Nominal Involute Points
    plot3(P(1,i),P(2,i),P(3,i),'.k');
```

```
            % Tool Center Points
            plot3(Q(1,i),Q(2,i),Q(3,i),'.r');
            % Tool Center Points, Rotated
            plot3(Qnew(1,i),Qnew(2,i),Qnew(3,i),'.r');

            pause;
        end

        hold off

    end

    Nnum = Nnum + 1000; % Increment N number for next
        % operation
end

%% Tooth Finishing Operation
if string(operations(4)) == 'Yes'
    fprintf('Internal␣Finishing\n');
    pagenum = 9; % Page of inputs in MS Excel spreadsheet

    % Inputs
    [b_total,b_step,Ret,stockclearance_finish,safedist1,...
        safedist2,Flimit,updn,increment,incsize,...
        ToolName,ToolNumber,ToolDia,ToolLength,Tooln,...
        ToolRI,Toolv,Toolft,coolant]...
        = inputs_toothfinishing(filename,pagenum);
```

```
if b_total > ToolLength
    fprintf('The tool cutting length is not long ');
    fprintf('enough. The cut will only extend to a');
    fprintf(' depth of %.3f mm.\r\n', ToolLength);
    b_total = ToolLength;
end
% Preliminary Stuff
[S, Flin] = cutconditions(ToolDia, Toolv, Tooln, ...
    Toolft, Smax);
[Rsafe1, Rsafe2, zlev, offset_N] = prelimcalcs(z, ...
    Rinner, Router, ToolDia, safedist1, safedist2, ...
    b_total, b_step);
th = stockclearance_finish;

% Generate Vectors
[P,T,R,N,Q,W, Fcorr, fvec] = VecGen_Complete(z, th, ...
    zlev, ToolDia, beta_b, updn, rollss, Rb, eta_b, ...
    offset_N, ImpAngles, Rsafe1, Rsafe2, increment, ...
    incsize, Ret);
% Convert
[Pnew, Qnew, nnew, Tnew, Rnew] = BCRotHTM(P, Q, N, T, R, ...
    fvec, WCS_offsets, beta_b, MachParams, 'NA00');
% Post
post_vectorial(Qnew, Rnew, W, Fcorr, Nnum, name, ...
    'Tooth Roughing', ToolName, ToolNumber, S, Flin, ...
    Flimit, coolant);
```

```
% TEST PLOTS
testplot = 0; % Turn on (1) or off (0)
if testplot == 1
    theta = 0:0.1:1.1*2*pi;
    basex = Rb*cos(theta);
    basey = Rb*sin(theta);
    basez = zeros(1,numel(theta));

    figure;
    plot3(basex,basey,basez,'-k'); hold all;
    % Nominal Involute Points
    plot3(P(1,:),P(2,:),P(3,:),'.k');
    % Tool Center Points
    plot3(Q(1,:),Q(2,:),Q(3,:),'.r');
    % Scaled Unit Normal Vectors
    quiver3(P(1,:),P(2,:),P(3,:),N(1,:),N(2,:),...
        N(3,:),0,'b');
    % Tool Orientation Vectors
    quiver3(Q(1,:),Q(2,:),Q(3,:),T(1,:),T(2,:),...
        T(3,:),'r');
    legend({'Base Circle','Nominal Involute Points',...
        'Unit Normal Vector','Tool Orientation Vector'});
    axis equal;
    axis([70 90 -10 10 -10 10]);
    grid on;
    hold off;
```

```
figure;
plot3(basex,basey,basez,'-k'); hold all;
% Nominal Involute Points
plot3(P(1,:),P(2,:),P(3,:),'.k');
% Tool Center Points
plot3(Q(1,:),Q(2,:),Q(3,:),'.r');
% Nominal Points, Rotated
plot3(Pnew(1,:),Pnew(2,:),Pnew(3,:),'.k');
% Tool Center Points, Rotated
plot3(Qnew(1,:),Qnew(2,:),Qnew(3,:),'.r');
% Scaled Unit Normal Vectors
quiver3(P(1,:),P(2,:),P(3,:),N(1,:),N(2,:),...
    N(3,:),0,'b');
% Tool Orientation Vectors
quiver3(Pnew(1,:),Pnew(2,:),Pnew(3,:),Tnew(1,:),...
    Tnew(2,:),Tnew(3,:),'r');
% Unit Normal Vectors
quiver3(Pnew(1,:),Pnew(2,:),Pnew(3,:),nnew(1,:),...
    nnew(2,:),nnew(3,:),0,'b');
legend({'Base Circle','Nominal Involute Points',...
    'Unit Normal Vector','Tool Orientation Vector'});
axis equal;
axis([70 90 -10 10 -10 10]);
grid on;
hold off;

figure;
```

```
        plot3(basex, basey, basez, '-k'); hold all;
        axis equal;
        for i = 1:numel(P(1,:))
            % Nominal Involute Points
            plot3(P(1,i),P(2,i),P(3,i),'.k');
            % Tool Center Points
            plot3(Q(1,i),Q(2,i),Q(3,i),'.r');
            % Tool Center Points, Rotated
            plot3(Qnew(1,i),Qnew(2,i),Qnew(3,i),'.r');
            pause;
        end

        hold off
    end

    Nnum = Nnum + 1000; % Increment N number for next
        % operation
end

%% Reference Ring
if string(operations(5)) == 'Yes'
    fprintf('Internal␣Reference␣Ring\n');
    pagenum = 6; % Page of inputs in MS Excel spreadsheet
    % Pull in Inputs
    [R_outer, R_inner, Depth_tot, Depth_inc,Ret,...
        ToolName,ToolNumber,ToolDia,Tooln,ToolLength,...
        ToolRI,Toolv,Toolft,coolant]...
```

```
        = inputs_refring(filename,pagenum);
    % Generate cutting feeds and speeds
    [S,Flin] = cutconditions(ToolDia,Toolv,Tooln,...
        Toolft,Smax);
    % Generate G code and write to text file
    operation_refring(R_outer,R_inner,Depth_tot,...
        Depth_inc,Ret,ToolName,ToolNumber,ToolDia,...
        ToolLength,ToolRI,S,Flin,coolant,Nnum,name);

    Nnum = Nnum + 1000; % Increment N number for next
        % operation
  end
else
end

% Home axes, etc.
footer(name);

% Close text file (stop writing and save)
fclose(name);
```

## A.2 Inputs

The following subsections give the functions to pull the inputs from the spreadsheet for each individual operation.

### A.2.1 Machine Parameters

```
function [MachParams,Smax]...
    = inputs_machine(filename,pagename)
```

```
% Jesse Groover
% 18 October 2018
% This function retrieves machine parameters for use in
% limiting the spindle speed to the machine maximum, and
% performing coordinate transformations.
params = xlsread(filename,pagename,'B3:B7');
MachParams = params(1:4);
Smax = params(5);
```

A.2.2 General File Information

```
function [z,txtfilename,auth,date,Onum,comment,desc,WCS,...
    WCSoffsets] = inputs_general(filename,pagename_general)
% Jesse Groover
% 17 October 2018
% This function retrieves general file information such as
% date, author, name, O number, etc.

[~,sigz,~] = xlsread(filename,pagename_general,'B3');
z = xlsread(filename,pagename_general,'C5');
if (string(sigz) == 'Internal')
    z = -z;
else
end
[~,txtfilename,~] = xlsread(filename,pagename_general,'B14');
txtfilename = string(txtfilename);
[~,auth,~] = xlsread(filename,pagename_general,'B18');
auth = string(auth);
```

```
[~,date,~] = xlsread(filename,pagename_general,'B19');
date = string(date);
Onum = xlsread(filename,pagename_general,'B15');
[~,comment,~] = xlsread(filename,pagename_general,'B16');
comment = string(comment);
[~,desc,~] = xlsread(filename,pagename_general,'B17');
desc = string(desc);
WCS = xlsread(filename,pagename_general,'B23');
WCSoffsets = xlsread(filename,pagename_general,'B24:B28');
```

### A.2.3 Geometry Input Information

```
function [mt,alphat,beta,b,x,k]...
    = inputs_Geometry(filename, pagenum)
% Jesse Groover
% 17 October 2018
% This function retrieves gear geometry parameters such as
% module, pressure angle, helix angle, etc.

allparams1 = xlsread(filename,pagenum,'C4:C10');
mt = allparams1(1); % Transverse Module
alphat = allparams1(3); % Transverse Pressure Angle
beta = allparams1(4); % Helix Angle
b = allparams1(5); % Facewidth
x = allparams1(6); % Profile Shift Coefficient
k = allparams1(7); % Addendum Modification Coefficient
```

### A.2.4 Facing Operation Input Information

```
function [R_outer,Depth_Total,Depth_Inc,Ret,helixstep,...
```

```
    ToolName,ToolNumber,ToolDia,ToolLength,Tooln,ToolRI,...
    Toolv,Toolft,coolant] = inputs_face(filename,pagenum)
% Jesse Groover
% 10 October 2018
% This function retrives inputs related to the facing
% operation.

allparams1 = xlsread(filename,pagenum,'B3:B7');
R_outer = allparams1(1);
Depth_Total = allparams1(2);
Depth_Inc = allparams1(3);
Ret = allparams1(4);
helixstep = allparams1(5);
[~,ToolName,~] = xlsread(filename,pagenum,'B9');
ToolName = string(ToolName);
allparams2 = xlsread(filename,pagenum,'B10:B16');
ToolNumber = allparams2(1);
ToolDia = allparams2(2);
Tooln = allparams2(3);
ToolLength = allparams2(4);
ToolRI = allparams2(5);
Toolv = allparams2(6);
Toolft = allparams2(7);
[~,coolant,~] = xlsread(filename,pagenum,'B17');
coolant = string(coolant);
```

#### A.2.5 Outer Cylinder Operation Input Information

```
function [R_final, R_init, Depth_Total, Depth_Inc, Ret,...
    ToolName, ToolNumber, ToolDia, ToolLength, Tooln,...
    ToolRI, Toolv, Toolft, coolant] = ...
    inputs_OuterCyl(filename, pagenum)
% Jesse Groover
% 16 October 2018
% This function retrieves inputs related to the outer
% cylinder cutting operation.

allparams1 = xlsread(filename,pagenum,'B3:B7');
R_final = allparams1(1); % Inner (Final) Radius
R_init = allparams1(2); % Outer (Initial) Radius
Depth_Total = allparams1(3); % Total Depth
Depth_Inc = allparams1(4); % Incremental Depth
Ret = allparams1(5); % Retract Height above part
[~,ToolName,~] = xlsread(filename,pagenum,'B9');
ToolName = string(ToolName); % Tool Name
allparams2 = xlsread(filename,pagenum,'B10:B16');
ToolNumber = allparams2(1); % Tool Number
ToolDia = allparams2(2); % Tool Diameter
Tooln = allparams2(3); % Number of Flutes
ToolLength = allparams2(4); % Cutting Length
ToolRI = allparams2(5); % Radial Immersion
Toolv = allparams2(6); % Surface Cutting Speed
Toolft = allparams2(7); % Feed per Tooth
[~,coolant,~] = xlsread(filename,pagenum,'B17');
coolant = string(coolant); % Coolant on or off
```

### A.2.6 Center Bore (External) or Hogging (Internal) Operation Input Information

```
function [R_outer,Depth_Total,Depth_Inc,Ret,helixstep,...
    ToolName,ToolNumber,ToolDia,ToolLength,Tooln,ToolRI,...
    Toolv,Toolft,coolant] = inputs_bore_hog(filename,pagenum)
% Jesse Groover
% 10 October 2018
% This function retrieves inputs related to the center bore
% cutting operation for external gears, and the center
% hogging operation for internal gears.

allparams1 = xlsread(filename,pagenum,'B3:B7');
R_outer = allparams1(1)/2; % Excel value is diameter, but we
    % need radius, so divide by 2
Depth_Total = allparams1(2);
Depth_Inc = allparams1(3);
Ret = allparams1(4);
helixstep = allparams1(5);
[~,ToolName,~] = xlsread(filename,pagenum,'B9');
ToolName = string(ToolName);
allparams2 = xlsread(filename,pagenum,'B10:B16');
ToolNumber = allparams2(1);
ToolDia = allparams2(2);
Tooln = allparams2(3);
ToolLength = allparams2(4);
ToolRI = allparams2(5);
Toolv = allparams2(6);
```

```
Toolft = allparams2(7);
[~,coolant,~] = xlsread(filename,pagenum,'B17');
coolant = string(coolant);
```

### A.2.7 Slotting Operation Input Information (External Only)

```
function [R_outer,R_inner,R_Step,ext,Ret,ToolName,...
    ToolNumber,ToolDia,ToolLength,Tooln,ToolRI,Toolv,...
    Toolft,coolant] = inputs_slots(filename,pagenum)
% Jesse Groover
% 17 October 2018
% This function retrieves input information related to
% precutting the slots on external gears.

allparams1 = xlsread(filename,pagenum,'B3:B7');
R_outer = allparams1(1)/2; % Excel value is diameter, but we
    % need radius, so divide by 2
R_inner = allparams1(2);
R_Step = allparams1(3);
ext = allparams1(4);
Ret = allparams1(5);
[~,ToolName,~] = xlsread(filename,pagenum,'B9');
ToolName = string(ToolName);
allparams2 = xlsread(filename,pagenum,'B10:B16');
ToolNumber = allparams2(1);
ToolDia = allparams2(2);
Tooln = allparams2(3);
ToolLength = allparams2(4);
```

```
ToolRI = allparams2(5);
Toolv = allparams2(6);
Toolft = allparams2(7);
[~,coolant,~] = xlsread(filename,pagenum,'B17');
coolant = string(coolant);
```

A.2.8 Tooth Roughing Operation Input Information

```
function [b_total,b_step,Ret,stockclearance,safedist1,...
    safedist2,Flimit,updn,increment,incsize,ToolName,...
    ToolNumber,ToolDia,ToolLength,Tooln,ToolRI,Toolv,...
    Toolft,coolant] = inputs_toothroughing(filename,pagenum)
% Jesse Groover
% 18 October 2018
% This function retrieves input information related to the
% roughing process for cutting gears using generation motion.

allparams1 = xlsread(filename,pagenum,'B3:B9');
b_total = allparams1(1); % Total axial depth
b_step = allparams1(2); % Axial step size
Ret = allparams1(3); % Retract height above part
stockclearance = allparams1(4); % Stock clearance to leave
safedist1 = allparams1(5); % Safe distance 1
safedist2 = allparams1(6); % Safe distance 2
Flimit = allparams1(7); % Limit for inverse time feed rate
    % values

[~,updn,~] = xlsread(filename,pagenum,'B10');
```

```
if string(updn) == 'Down'
    updn = -1;
elseif string(updn) == 'Up'
    updn = 1;
end
[~,increment,~] = xlsread(filename,pagenum,'B11');
increment = string(increment);
incsize = xlsread(filename,pagenum,'B12');

[~,ToolName,~] = xlsread(filename,pagenum,'B14');
ToolName = string(ToolName);
allparams2 = xlsread(filename,pagenum,'B15:B21');
ToolNumber = allparams2(1);
ToolDia = allparams2(2);
Tooln = allparams2(3);
ToolLength = allparams2(4);
ToolRI = allparams2(5);
Toolv = allparams2(6);
Toolft = allparams2(7);
[~,coolant,~] = xlsread(filename,pagenum,'B22');
coolant = string(coolant);
```

### A.2.9 Tooth Finishing Operation Input Information

```
function [b_total,b_step,Ret,stockclearance,safedist1,...
    safedist2,Flimit,updn,increment,incsize,ToolName,...
    ToolNumber,ToolDia,ToolLength,Tooln,ToolRI,Toolv,...
    Toolft,coolant] = inputs_toothfinishing(filename,...
```

```
    pagenum)
% Jesse Groover
% 18 October 2018
% This function retrieves input information related to the
% finishing process for cutting gears using generation
% motion.

allparams1 = xlsread(filename,pagenum,'B3:B9');
b_total = allparams1(1); % Total axial depth
b_step = allparams1(2); % Axial step size
Ret = allparams1(3); % Retract height above part
stockclearance = allparams1(4); % Stock clearance to leave
safedist1 = allparams1(5); % Safe distance 1
safedist2 = allparams1(6); % Safe distance 2
Flimit = allparams1(7); % Limit for inverse time feed rate
    % values

[~,updn,~] = xlsread(filename,pagenum,'B10');
if string(updn) == 'Down'
    updn = -1;
elseif string(updn) == 'Up'
    updn = 1;
end
[~,increment,~] = xlsread(filename,pagenum,'B11');
increment = string(increment);
incsize = xlsread(filename,pagenum,'B12');
```

```
[~,ToolName,~] = xlsread(filename,pagenum,'B14');
ToolName = string(ToolName);
allparams2 = xlsread(filename,pagenum,'B15:B21');
ToolNumber = allparams2(1);
ToolDia = allparams2(2);
Tooln = allparams2(3);
ToolLength = allparams2(4);
ToolRI = allparams2(5);
Toolv = allparams2(6);
Toolft = allparams2(7);
[~,coolant,~] = xlsread(filename,pagenum,'B22');
coolant = string(coolant);
```

#### A.2.10 Reference Ring Operation Input Information (Internal Only)

```
function [R_outer, R_inner, Depth_Tot, Depth_inc,Ret,...
    ToolName,ToolNumber,ToolDia,Tooln,ToolLength,ToolRI,...
    Toolv,Toolft,coolant] = inputs_refring(filename,pagenum)
% Jesse Groover
% 22 October 2018
% This function retrieves input information related to
% cutting the reference ring in an internal gear.  The
% reference ring is used for find the center axis of the
% gear during measurement.

allparams1 = xlsread(filename,pagenum,'B3:B7');
R_outer = allparams1(1);
R_inner = allparams1(2);
```

```
Depth_Tot = allparams1(3);
Depth_inc = allparams1(4);
Ret = allparams1(5);

[~,ToolName,~] = xlsread(filename,pagenum,'B9');
ToolName = string(ToolName);
allparams2 = xlsread(filename,pagenum,'B10:B16');
ToolNumber = allparams2(1);
ToolDia = allparams2(2);
Tooln = allparams2(3);
ToolLength = allparams2(4);
ToolRI = allparams2(5);
Toolv = allparams2(6);
Toolft = allparams2(7);
[~,coolant,~] = xlsread(filename,pagenum,'B17');
coolant = string(coolant);
```

## A.3 Geometry Calculation Function

```
function [Rb,Rp,roll,ImpAngles,eta_b,psi_b,dtoolmax,...
    Rinner,Router,beta_b]...
    = Geometry(mt,z,alpha_t_in,beta_in,x,k,plot)
% Geometry
% Jesse Groover
% 10 January 2017
% Updated 12 January 2017
% Updated and renamed to "Geometry" 13 January 2017
% Updated 19 January 2017
```

```
% Updated many times since
% Updated 18 May 2017
% Updated 5 January 2018 - Removed toolpath generation.
%     Now only geometry
% Updated 17 April 2018 - Updated to reflect standard
%     terminology (ISO 21771)
% Updated multiple times
% Final version, 28 November 2018
%
% This program generates the XY coordinates for all of the
% tool positions in milling an internal gear using both
% continuous motion toolpaths and incremented motion.  In
% order to do this, it first calculates multitudinous
% parameters related to the gear.
%
% Outputs:
% Rb - Base radius, mm
% Rp - Pitch (ref.) radius, mm
% roll - A vector containing roll angle end points
%       1 - Roll angle at inner radius (rad)
%       2 - Roll angle at outer radius (rad)
% ImpAngles - A vector containing important angles,
%       irrespective of external or internal gears.
%       1 - Inside Involute, at base circle, half angle (rad)
%       2 - Inside Involute, at reference circle, half angle
%             (rad)
%       3 - Inside Involute, at outer radius, half angle (rad)
```

```
%        4 - Outside Involute, at base circle, half angle (rad)
%        5 - Outside Involute, at reference circle, half angle
%              (rad)
%        6 - Outside Involute, at outer radius, half angle (rad)
%        7 - Tooth pitch (rad)
% eta_b - Space width half angle, in radians
% psi_b - Tooth thickness half angle, in radians
% dtoolmax - Maximum tool diameter, mm
% Rinner - Inner radius.  Root radius for external, tip
%   radius for internal.
% Router - Outer radius.  Tip radius for external, root
%   radius for internal.
% beta_b - Helix angle at the base circle in radians
%
% % Necessary geometry (Inputs)
% clear; close all;
% mt = 5; % Transverse Module (in Plane normal to rotation
%   axis), mm
% z = 15; % Number of teeth.  Negative if Internal Gear.
% alpha_t_in = 20; % Transverse Pressure Angle at reference
%   cylinder, degrees
% beta_in = 0; % Helix angle at reference cylinder, deg
% x = 0; % Profile Shift Coefficient
% k = 0; % Tip Alteration Coefficient
% plot = 1; % Plot final geometry (1), or not (0)

% Calculations
```

```
tau = 2*pi/abs(z); % Angular Pitch, rad
beta = beta_in*pi/180; % Helix angle at ref. cyl., rad
alpha_t = alpha_t_in*pi/180; % Convert Transverse Pressure
%   (input) from deg. to rad.
alpha_n = atan(tan(alpha_t)*cos(beta)); % Normal pressure
%   angle at ref. cyl., rad
beta_b = atan(tan(beta)*cos(alpha_t)); % Helix angle at base
%   circle, rad
mn = mt*cos(beta); % Normal (in plate normal to tooth
%   profile, not rot. axis) Module, mm
mx = mt/tan(beta); % Axial Module, mm (also equal to
%   mn/sin(beta), and mn/cos(gamma)
d = mt*abs(z); % Pitch Diameter, in mm
db = abs(z)*mn*cos(alpha_n)/cos(beta_b); % Base diameter,
%   in mm
Rp = d/2; % Pitch Radius, in mm
Rb = db/2; % Base radius in mm
pt = pi*mt; % Transverse Pitch, mm
pn = pi*mn; % Normal Pitch, mm
pz = (abs(z)*mn*pi)/(sin(beta)); % Lead, mm
px = pi*mx; % Axial pitch (also equal to pz/abs(z)), mm
pbt = Rb*tau; % Transverse Base Pitch, mm
pbn = pn*cos(alpha_n); % Normal Base Pitch, mm

dv = d + 2*sign(z)*mn; % Diameter of "V-Cylinder", or
%   Profile Shift Cylinder, mm
haP = 1.00*mt; % Addendum, mm
```

```
hfP = 1.25*mt; % Dedendum, mm % was 1.00*mt
da = d + 2*sign(z)*(x*mn + haP + k*mn); % Tip Cylinder
%   Diameter, mm
df = d - 2*sign(z)*(hfP - x*mn); % Root Cylinder Diameter,
%   mm
Ra = da/2; % Tip Radius, mm
Rf = df/2; % Root Radius, mm
h = abs(da-df)/2; % Tooth depth, mm
ha = haP + x*mn + k*mn; % Total Addendum, mm
hf = hfP - x*mn; % Total Dedendum, mm

inv_alpha_t = tan(alpha_t) - alpha_t; % Angular Position
%   from base point to ref. cyl.
eta = (pi - 4*x*tan(alpha_n))/(2*abs(z)); % Space width half
%   angle at ref. cyl., rad.
eta_b = eta - sign(z)*inv_alpha_t; % Space width half angle
%   at base cyl., rad
psi = (pi + 4*x*tan(alpha_n))/(2*abs(z)); % Tooth thickness
%   half angle at ref. cyl., rad
psi_b = psi + sign(z)*inv_alpha_t; % Tooth thickness half
%   angle at base cyl., rad.
alpha_t_f = acos(db/df); % Transverse Pressure Angle at
%   dedendum (tooth root), rad
inv_alpha_t_f = tan(alpha_t_f) - alpha_t_f; % Angular
%   Position from base point to dedendum, rad
alpha_t_a = acos(db/da); % Transverse Pressure Angle at
%   addendum (tooth tip), rad
```

```
inv_alpha_t_a = tan(alpha_t_a) - alpha_t_a; % Angular
%   Position from base point to addendum, rad
eta_f = eta - sign(z)*(inv_alpha_t - inv_alpha_t_f); % Space
%   width half angle at tooth root, rad;
eta_a = eta - sign(z)*(inv_alpha_t - inv_alpha_t_a); % Space
%   width half angle at tooth tip, rad;
psi_f = psi + sign(z)*(inv_alpha_t - inv_alpha_t_f); % Tooth
%   thickness half angle at dedendum (tooth root), rad
psi_a = psi + sign(z)*(inv_alpha_t - inv_alpha_t_a); % Tooth
%   thickness half angle at addendum (tooth tip), rad

st = d*psi; % Transverse tooth thickness at ref. cyl., mm
sn = st*cos(beta); % Normal tooth thickness at ref. cyl., mm
et = d*eta; % Trasnverse space width at ref. cyl., mm
en = et*cos(beta); % Normal space width at ref cyl., mm

if sign(z) == 1 % External Gears
    xi_a = tan(alpha_t_a); % Roll angle at tooth tip
        % (addendum), rad
    if df <= db % Root Circle Smaller than Base Circle
        xi_f = 0; % Roll angle at tooth root (dedendum), rad
    else
        xi_f = tan(alpha_t_f); % Roll angle at tooth root
            % (dedendum), rad
    end
    Rinner = Rf; % Inner radius for external gears (root
        % circle), mm
```

```
    Router = Ra; % Outer radius for external gears (tip
        % circle), mm
    dtoolmax = 2*Rf*sin(eta_b); % Maximum tool diameter, mm
    % ImpAngles = [Inside Inv. base, Inside Inv. ref, inside
        % inv. outer radius, outside inv. base, outside inv.
        % ref, outside inv. outer radius (all halves), tooth
        % pitch]
    ImpAngles = [psi_b, psi, psi_a, eta_b, eta, eta_a,tau];
    roll = [xi_f xi_a]; % Roll start (smaller radius) and
        % end (larger radius) angles, rad
elseif sign(z) == -1 % Internal Gears
    xi_f = tan(alpha_t_f); % Roll angle at tooth root
        % (dedendum), rad
    if da <= db % Tip Circle Smaller than Base Circle
        xi_a = 0; % Roll angle at tooth tip (addendum), rad
    else
        xi_a = tan(alpha_t_a); % Roll angle at tooth tip
            % (addendum), rad
    end
    Rrot = Rb*tan(eta_b); % Rotation radius for rounded
        % teeth (external), mm
    Rcent = sqrt((Rb^2) + (Rrot^2)); % Distance from gear
        % center to center of rotation for rounded teeth, mm
    Rinner = Ra; %cent - Rrot; % Inner Radius for internal
        % gears (tip circle), mm
    Router = Rf; % Outer Radius for internal gears (root
        % circle), mm
```

```
    dtoolmax = 2*Ra*sin(psi_a); % Maximum tool diameter, mm
    % ImpAngles = [Inside inv. base, Inside Inv. ref, inside
        % inv. outer radius, outside inv. base, outside inv.
        % ref, outside inv. outer radius (all halves), tooth
        % pitch]
    ImpAngles = [eta_b, eta, eta_f, psi_b, psi, psi_f,tau];
    roll = [xi_a xi_f]; % Roll start (smaller radius) and
        % end (larger radius) angles, rad
end

% Plot Final Gear Geometry
if plot == 1
    plotfinalgear(Rp,Rb,roll,Rinner,z,ImpAngles);
else
end
```

## A.4 Supporting Functions

The following functions support the individual smaller tasks to be completed.

### A.4.1 NC File Header

```
function header(name,auth,date,O,comment,desc,WCS)
% Jesse Groover
% 2 October 2018
% This function writes the first few lines of G code to the
% specified text file.

fprintf(name,'%%\r\nO%.0f␣(%s);\r\n',O,comment);
fprintf(name,'(%s);\r\n',auth);
```

```
fprintf(name,'(%s);\r\n',date);
fprintf(name,'(%s);\r\n(------------);\r\n',desc);
fprintf(name,'G00 G%.0f G90 G94 G49 M11 M69;\r\n',WCS);
end
```

### A.4.2 Cutting Feed and Speed

```
function [S,Flin] = cutconditions(D,v,n,ft,Smax)
% Jesse Groover
% 30 January 2018
% This function takes tool diameter, desired cutting speed,
% and number of flutes, and calculates spindle speed (RPM)
% and linear feedrate (mm/min).
%
% Inputs:
%       D - Tool Diameter (mm)
%       v - Cutting Speed (m/min)
%       n - Number of teeth on tool (teeth/rev)
%       ft - Feed per Tooth (mm/tooth)

S = 1000*v/(pi*D); % Spindle Speed, RPM
if S > Smax
    S = Smax;
else
end
Flin = S*n*ft; % Feedrate, mm/min
```

### A.4.3 Cylindrical Pocketing Operation (Facing, Center Bore, Center Hogging)

```
function operation_pocket_cyl(R_outer,Depth_Total,Depth_Inc,...
```

```
    Ret,helixstep,ToolName,ToolNumber,ToolDia,ToolLength,...
    ToolRI,S,Flin,coolant,Nnum,name)
% Updated 10/10/2018
% Jesse Groover
% This program performs a cylindrical pocket type operation.
% Geometry Parameters:
%- Outer Radius (mm)
%- Depth (mm)
% Manufacturing Parameters:
%- Tool Parameters:
% - Tool Name
% - Tool Number
% - Diameter
% - Tool Cutting Length
% - Number of Flutes
%- Process Parameters:
% - Incremental Depth (mm)
% - Incremental helix distances (mm)
% - Retract height above part (mm)
% - Radial Immersion
% - Surface Cutting Speed (m/min)
% - Feed Per Tooth (mm)
% - Coolant on ('Yes') or off ('No')

% Important Calculations
offset = 3*ToolDia/8; % Offset for spiral plunge, mm (must...
    % be less than tooldia/2)
```

```
plungedia = 2*offset + ToolDia; % Plunging (starting) hole
    % diameter
% FIXME: Print error message if plungdia is larger than
    % 2*R_outer, or find solution
% Number of radial roughing passes (spiral)
Nrad = ceil((R_outer-(plungedia/2))/(ToolRI*ToolDia));
passwidth = (R_outer-(plungedia/2))/Nrad; % Width of radial...
%roughing passes
if Depth_Total <= ToolLength
    lim = Depth_Total;
else
    lim = ToolLength;
    error = strcat('The desired cutting depth is larger',...
        ' than the extension of the tool. The cut will',...
        ' only extend to', sprintf(' %.3f mm.',lim));
    msgbox(error,'Error');
end
zstart = 0;
zpos = zstart:-Depth_Inc:-lim;
if zpos(end) ~= -lim
    zpos(end + 1) = -lim;
end
extrapasses = 2; % Number of extra helix passes in first
    % approach
Fapp = 1000; % Approach rate, mm/min

% G-Code
```

```
% Prep
fprintf(name, '\r\nN%.0f (Cylinder Pocket Operation);\r\n',...
    Nnum);
fprintf(name, '(Zero Return Z, Tool change, ');
fprintf(name, 'apply length offset);\r\n');
fprintf(name, '(%s);\r\n',ToolName);
fprintf(name, 'G00 G91 G28 Z0;\r\n'); % Zero-Return Z-axis
fprintf(name, 'G00 G91 G28 X0 Y0;\r\n'); % Zero-Return X and
    % Y Axes
fprintf(name, 'T%.0f M06;\r\n',ToolNumber); % Tool change
fprintf(name, 'G43 H%.0f;\r\n',ToolNumber); % Apply length
    % offset
fprintf(name, 'M11;\r\nM69;\r\n'); % Unclamp B and C Axes
fprintf(name, 'G90 B0 C0;\r\n'); % Go to zero point for B and
    % C axes
fprintf(name, 'M68;\r\n'); % Reclamp B and C Axes

% z = [];
for i = 2:numel(zpos) % i = axial (Z) roughing pass

    % Initialize Plunging Helix
    if i == 2
        fprintf(name, '(Plunging Helix, to a ');
        fprintf(name, 'total plunge depth of %.3f mm);\r\n',...
            zpos(i));
        fprintf(name, '(Approach and spindle and');
            fprintf(name, ' coolant on);\r\n');
```

```
    fprintf(name,'G00 G91 G28 Z0;\r\n'); % Zero-Return
        % Z-axis
    % Position in X and Y
    fprintf(name,'G00 G90 X0 Y%.3f;\r\n',offset);
    fprintf(name,'G01 G90 Z%.3f F%.3f;\r\n',Ret,Fapp);
    % Approach in Z
    fprintf(name,'M03 S%.3f;\r\n',S); % Spindle on
    if coolant == 'Yes'
        fprintf(name,'M08;\r\n'); % Flood Coolant On
    else
    end

    z = extrapasses*helixstep; % The starting z height
    % Go to starting point
    fprintf(name,'G01 G90 Z%.3f F%.3f;\r\n',z,Flin);
else
    fprintf(name,'(Plunging Helix to a ');
    fprintf(name,'plunge depth of %.3f mm);\r\n',zpos(i));
    fprintf(name,'G01 G90 X0 Y%.3f F%.3f;\r\n',offset,...
        Flin); % Go to starting point
    % Position in X and Y
    z = zpos(i-1); % The starting height
end

% Set helix z heights
clear zinc;
zinc = z:-helixstep:zpos(i);
```

```
if zinc(end) ~= zpos(i)
    zinc(end + 1) = zpos(i);
end
zinc(end + 1) = zinc(end);

% Actual helix operation
for j = 2:numel(zinc)
    fprintf(name,...
        'G90 G03 X0 Y%.3f Z%.3f I0 J%.3f F%.3f;\r\n',...
        offset,zinc(j),-offset,Flin); % Helix downwards
end

% Working outwards
fprintf(name,'(Working outwards by walking out in Y, ');
fprintf(name,'spinning C 360 deg. %.0f Passes);\r\n',...
    Nrad);
r = offset + passwidth; % Set initial radial tool
    % position
for j = 1:Nrad
    Frot = (Flin/r)*(180/pi); % Rotary feedrate, in
        % deg/min
    % Walk forward
    fprintf(name,'G01 G90 Y%.3f F%.3f;\r\n',r,Flin);
    % Rotate table
    fprintf(name,'G01 G91 C380.0 F%.3f;\r\n',Frot);
    if j < Nrad
        r = r + passwidth;
```

```
            else
            end
        end
end

% Retracting
fprintf(name,...
    '(Retract, spindle and coolant off, zero');
    fprintf(name,' return Z);\r\n');
fprintf(name,'G00 G90 Z%.3f;\r\n',Ret); % Retract
fprintf(name,'M05;\r\n'); % Spindle off
fprintf(name,'M09;\r\n'); % Coolant off
fprintf(name,'G91 G28 Z0;\r\n'); % Zero Return Z Axis
fprintf(name,'G91 G28 B0 C0;\r\n'); % Zero Return B and C
    % Axes
fprintf(name,'M01;\r\n'); %

%fclose(name);
end
```

### A.4.4 Outer Cylinder Operation

```
function operation_OuterCyl(R_final, R_init, Depth_Total,...
    Depth_Inc, Ret, ToolName,ToolNumber,ToolDia,...
    ToolLength,ToolRI,S,Flin,coolant,Nnum,name)
% Updated 10/10/2018
% Jesse Groover
% This program performs a cylindrical pocket type operation.
```

```
% Geometry Parameters:
%- Outer Radius (mm)
%- Depth (mm)
% Manufacturing Parameters:
%- Tool Parameters:
%  - Tool Name
%  - Tool Number
%  - Diameter
%  - Tool Cutting Length
%  - Number of Flutes
%- Process Parameters:
%  - Incremental Depth (mm)
%  - Incremental helix distances (mm)
%  - Retract height above part (mm)
%  - Radial Immersion
%  - Surface Cutting Speed (m/min)
%  - Feed Per Tooth (mm)
%  - Coolant on ('Yes') or off ('No')

% R_final = 100;
% R_init = 150;
% Depth_Total = 10;
% Depth_Inc = 2;
% Ret = 5;
% ToolName = 'Test Tool';
% ToolNumber = 13;
% ToolDia = 12.7;
```

```
% ToolLength = 15;
% Tooln = 4;
% ToolRI = 0.5;
% Toolv = 243;
% Toolft = 0.0254;
% coolant = 'Yes';
% Nnum = 1000;
% [name] = openfile('test.txt');

Fapp = 1000; % Approach rate, mm/min
safedistfact = 1.5;

passwidth = ToolRI*ToolDia;
R_vec = R_init:-passwidth:R_final;
if R_vec(end) ~= R_final
    R_vec(end + 1) = R_final;
end
R_vec = R_vec + ToolDia/2; % Vector of tool center point
    % positions (Radii (mm))

if Depth_Total <= ToolLength
    lim = Depth_Total;
else
    lim = ToolLength;
    error = strcat('The desired cutting depth is larger',...
        ' than the extension of the tool.  The cut will',...
        ' only extend to',sprintf(' %.3f mm.',lim));
```

```
    msgbox(error, 'Error');
end
zstart = 0;
zpos = zstart:-Depth_Inc:-lim; % Vector of incremental Z
    % heights at which to perform passes
if zpos(end) ~= -lim
    zpos(end + 1) = -lim;
end



% G-Code
% Prep
fprintf(name, '\r\nN%.0f (Outer Cylinder Rotary ',Nnum);
    fprintf(name, 'Operation);\r\n');
fprintf(name, '(Zero Return Z, Tool change, ');
fprintf(name, 'apply length offset);\r\n');
fprintf(name, '(%s);\r\n',ToolName);
fprintf(name, 'G00 G91 G28 Z0;\r\n'); % Zero-Return Z-axis
fprintf(name, 'G00 G91 G28 X0 Y0;\r\n'); % Zero-Return X and
    % Y Axes
fprintf(name, 'T%.0f M06;\r\n',ToolNumber); % Tool change
fprintf(name, 'G43 H%.0f;\r\n',ToolNumber); % Apply length
    % offset
fprintf(name, 'M11;\r\nM69;\r\n'); % Unclamp B and C Axes
fprintf(name, 'G90 B0 C0;\r\n'); % Go to zero point for B and
    % C axes
fprintf(name, 'M68;\r\n'); % Reclamp B and C Axes
```

```
% Main Operation
for i = 1:numel(R_vec) % Loop by Radius
    xstart = safedistfact*sqrt(2*R_vec(i)*ToolRI*ToolDia-...
        (ToolRI*ToolDia)^2); % Safe X position
    Frot = (Flin/R_vec(i))*(180/pi); % Rotary feedrate, in
        % deg/min
    for j = 2:numel(zpos) % Loop by Z Height
        % Initialize Approach
        if (i == 1 && j == 2)
            fprintf(name,'(Approach and spindle and');
                fprintf(' coolant on);\r\n');
            % Zero-Return Z-axis
            fprintf(name,'G00 G91 G28 Z0;\r\n');
            fprintf(name,'G00 G90 X%.3f Y%.3f;\r\n',...
                -xstart,R_vec(i)); % Position in X and Y
            fprintf(name,'G01 G90 Z%.3f F%.3f;\r\n',Ret,...
                Fapp);
            % Approach in Z
            fprintf(name,'M03 S%.3f;\r\n',S); % Spindle on
            if coolant == 'Yes'
                % Flood Coolant On
                fprintf(name,'M08 (Coolant On);\r\n');
            else
            end
        else
            % Go to starting point in X and Y
```

```
        fprintf(name,'G00 G90 X%.3f Y%.3f',-xstart,...
            R_vec(i));
            fprintf(name,'F%.3f (Reset);\r\n',Flin);

    end
    % Go to appropriate Z height
    fprintf(name,'G01 G90 Z%.3f F%.3f',zpos(j),Flin);
        fprintf(name,'(Approach in Z);\r\n');
        % Walk in to starting point linearly
    fprintf(name,'G90 X0 Y%.3f',R_vec(i));
        fprintf(name,'F%.3f (Walk in);\r\n',Flin);
    % Rotate
    fprintf(name,'G91 C-380.0 F%.3f (Rotate);\r\n',...
        Frot);
    % Walk safe distance away
    fprintf(name,'G90 X%.3f F%.3f (Walk out);\r\n',...
        xstart, Flin);
    % Retract
    fprintf(name,'G00 G90 Z%.3f (Retract);\r\n',Ret);

  end
end

% Retracting
fprintf(name,'(Retract, spindle and coolant off,');
  fprintf(name,' zero return Z);\r\n');
fprintf(name,'M05;\r\n'); % Spindle off
```

```
fprintf(name,'M09;\r\n'); % Coolant off
fprintf(name,'G91␣G28␣Z0;\r\n'); % Zero Return Z Axis
fprintf(name,'G91␣G28␣B0␣C0;\r\n'); % Zero Return B and C Ax
fprintf(name,'M01;\r\n'); %

%fclose(name);
end
```

### A.4.5 Slotting Operation

```
function [P,T,R,n,Q,W,Fcorr,fvec] = operation_slots_radial...
    (znum,Rb,beta_b,eta_b,b,R_outer,R_inner,R_step,ext,...
    ToolDia,ToolRI,Ret)
% Jesse Groover
% This function generates the toolpaths for a slotting
% operation, with the tool oriented radially with respect to
% the base cylinder.

% Troubleshooting Inputs
% ToolDia = 3.175;
% ToolRI = 1.0;
% znum = 15;
% Ret = 10;
%
% Rb = 100; % Base Radius
% beta_b = 0*pi/180; % Helix Angle at base circle
% eta_b = 0.0898; % Tooth Space Half Angle
% b = 10; % Facewidth
```

```
% ext = 5;
%
% R_outer = 110;
% R_inner = 90;
% R_step = 5;
R_vec = R_outer:-R_step:R_inner;
if R_vec(end) ~= R_inner
    R_vec(end+1) = R_inner;
end

P = []; T = []; R = []; n = []; Q = []; W = []; Fcorr = [];
fvec = [];

for i = 1:abs(znum) % Tooth Number
    offset_N = (i-1)*2*pi/abs(znum) + pi/abs(znum);
    for j = 1:numel(R_vec) % Radius
        offset_th_vec = eta_b-ToolDia*0.5/R_vec(j):...
            -ToolDia*ToolRI/R_vec(j):0;
        if offset_th_vec(end) ~= 0
            offset_th_vec(end+1) = 0;
        end
        for k = 1:numel(offset_th_vec) % Thickness
            % Side 1
            f = 1;
            offset_tot = offset_N + f*(eta_b...
                - offset_th_vec(k));
            zpoint = [-b-ext ext];
```

```
% Approach
[Pnew,Tnew,Rnew,nnew,Qnew,Wnew,Fcorrnew,fvecnew]...
    = vecgen_slot_approach(f,Rb,beta_b,...
    offset_tot,R_vec(j),R_outer,Ret,zpoint(1),...
    ToolDia);
P = [P,Pnew];
T = [T,Tnew];
R = [R,Rnew];
n = [n,nnew];
Q = [Q,Qnew];
W = [W,Wnew];
Fcorr = [Fcorr,Fcorrnew];
fvec = [fvec,fvecnew];


% Cut Side 1
[Pnew,Tnew,Rnew,nnew,Qnew,Wnew,Fcorrnew,fvecnew]...
    = vecgen_slot_cutside(f,Rb,beta_b,...
    offset_tot,R_vec(j),zpoint,ToolDia);
P = [P,Pnew];
T = [T,Tnew];
R = [R,Rnew];
n = [n,nnew];
Q = [Q,Qnew];
W = [W,Wnew];
Fcorr = [Fcorr,Fcorrnew];
fvec = [fvec,fvecnew];
```

```
% Side 2 - Three Points
f = -1;
offset_tot = offset_N + f*(eta_b...
    - offset_th_vec(k));
zpoint = [ext -b-ext];
% Cut Side 2
[Pnew,Tnew,Rnew,nnew,Qnew,Wnew,Fcorrnew,fvecnew]...
    = vecgen_slot_cutside(f,Rb,beta_b,...
    offset_tot,R_vec(j),zpoint,ToolDia);
P = [P,Pnew];
T = [T,Tnew];
R = [R,Rnew];
n = [n,nnew];
Q = [Q,Qnew];
W = [W,Wnew];
Fcorr = [Fcorr,Fcorrnew];
fvec = [fvec,fvecnew];

% Retract (Use approach function, just one point
% radially farther out)
[Pnew,Tnew,Rnew,nnew,Qnew,Wnew,Fcorrnew,fvecnew]...
    = vecgen_slot_approach(f,Rb,beta_b,...
    offset_tot,R_vec(j),R_outer,Ret,zpoint(end),...
    ToolDia);
P = [P,Pnew];
T = [T,Tnew];
```

```
            R = [R,Rnew];
            n = [n,nnew];
            Q = [Q,Qnew];
            W = [W,Wnew];
            Fcorr = [Fcorr,Fcorrnew];
            fvec = [fvec,fvecnew];

        end
    end
end
```

A.4.5.1 Slotting Point Vector

```
function [P] = Pvec_slot_radial(Rb,beta_b,offset,Rpoint,...
    zpoint)
% 23 October 2018
% Jesse Groover
% This function generates a point vector on a helicoid
% surface.

c = Rb/tan(beta_b);
offset_point = offset + zpoint/c; % Offset angle, taking
    % into account initial offset, and Z height

P = [Rpoint.*cos(offset_point); % Cooresponding X coordinate
    Rpoint.*sin(offset_point); % Cooresponding Y coordinate
    zpoint];
```

A.4.5.2 Slotting Normal Vector

```
function [n] = nvec_slot_radial(f,Rb,beta_b,offset,...
    Rpoint,zpoint)
% Jesse Groover
% 23 October 2018
% This function generates the surface unit normal vector at
% a point P on a helicoid surface.

c = Rb/tan(beta_b);
offset_point = offset + zpoint./c; % Offset angle, taking
    % into account initial offset, and Z height
beta_point = atan((Rpoint./Rb).*tan(beta_b)); % Helix angle
    % at this radius

n = f*[sin(offset_point).*cos(beta_point);
    -cos(offset_point).*cos(beta_point);
    sin(beta_point)*ones(1,numel(zpoint))];
```

#### A.4.5.3 Slotting Tool Orientation Vector

```
function [T] = Tvec_slot_radial(Rb,beta_b,offset,zpoint)
% Jesse Groover
% 23 October 2018
% This function generates the tool orientation vector at a
% particular point on a helicoid surface. The tool
% orientation vector points directly from the point on the
% surface outward radially.

c = Rb/tan(beta_b);
```

```
offset_point = offset + zpoint./c; % Offset angle, taking
    % into account initial offset, and Z height

T = [cos(offset_point);
    sin(offset_point);
    0*ones(1,numel(zpoint))];
```

A.4.5.4 Slotting Approach

```
function [Pnew,Tnew,Rnew,nnew,Qnew,Wnew,Fcorrnew,fvecnew]...
    = vecgen_slot_approach(f,Rb,beta_b,offset,R_vec,...
    R_outer,Ret,zpoint,ToolDia)
% Jesse Groover
% 23 October 2018
% This function generates the vectors for the two approach
% points for a slot precutting operation.

Pnew = Pvec_slot_radial(Rb,beta_b,offset,R_outer + Ret,...
    zpoint);
Tnew = Tvec_slot_radial(Rb,beta_b,offset,zpoint);
Rnew = zeros(2,numel(zpoint));
nnew = nvec_slot_radial(f,Rb,beta_b,offset,R_vec,zpoint);
Qnew = Pnew + 0.5*ToolDia*nnew;
Wnew = [0; % Rapid or feed
    94; % Feedrate type
    90]; % Abs or Inc
Fcorrnew = ones(1,numel(zpoint));
fvecnew = f*ones(1,numel(zpoint));
```

#### A.4.5.5 Slotting, Cutting One Side

```
function [Pnew,Tnew,Rnew,nnew,Qnew,Wnew,Fcorrnew,fvecnew]...
    = vecgen_slot_cutside(f,Rb,beta_b,offset,R_vec,...
    zpoint,ToolDia)
% Jesse Groover
% 23 October 2018
% This function generates the points for one pass of cutting
% a helicoid surface.

Pnew = Pvec_slot_radial(Rb,beta_b,offset,R_vec,zpoint);
Tnew = Tvec_slot_radial(Rb,beta_b,offset,zpoint);
Rnew = zeros(2,numel(zpoint));
nnew = nvec_slot_radial(f,Rb,beta_b,offset,R_vec,zpoint);
Qnew = Pnew + 0.5*ToolDia*nnew;
Wnew = [0,1; % Rapid or feed
    94,94; % Feedrate type
    90,90]; % Abs or Inc
Fcorrnew = ones(1,numel(zpoint));
fvecnew = f*ones(1,numel(zpoint));
```

### A.4.6 Homogeneous Transformations Matrices for Coordinate Transformations

```
function [Pnew,Qnew,nnew,Tnew,Rnew] = BCRotHTM(P,Q,n,T,R,...
    fvec,WCS_offsets,beta_b,MachParams,OrientVec)
% Jesse Groover
% 24 May 2018
% This function takes a bunch of vectors and performs
% coordinate rotations for five axis milling.
```

```
%
% The P, N, and T, vectors must be rotated such that the T
% vector is oriented vertically, and the N vector lies in a
% XZ plane.
%
% The vectors are all rotated about Z (C rotation) first, to
% put either the T or N vector in the XZ plane.  Then they
% are all rotated about Y (B rotation) to orient the T
% vector vertically.
%
% UPDATE 23 OCTOBER 2018: V VECTOR IS NO LONGER USED AT ALL
%
% These transformations must be done for each individual
% point (each column in each of the vectors)

L = [MachParams(3) - WCS_offsets(1);
     MachParams(2) - WCS_offsets(2);
     MachParams(4) - WCS_offsets(3)];

% Add a fourth row of ones, for HTM operations
P(4,:) = ones(1,numel(P(1,:)));
Q(4,:) = ones(1,numel(Q(1,:)));
T(4,:) = ones(1,numel(T(1,:)));
n(4,:) = ones(1,numel(n(1,:)));
Rnew = R;

for i = 1:numel(Q(1,:))
```

```
% Calculate C Rotation Angle
if beta_b ~= 0 || all(OrientVec == 'Tool') % If the
        % helix angle is not zero, or if it is desired
        % to use the tool specifically for orientation...
    Rnew(1,i) = atan2(T(2,i),T(1,i)); % Calculate C axis
        % position, and store in R matrix.
else % Otherwise, use the surface normal vector for
        % orientation (generation motion only)
    Rnew(1,i) = atan2(n(2,i),n(1,i)); % Same as above,
        % but use n vector
    if fvec(i) == -1
        Rnew(1,i) = Rnew(1,i) + pi;
    end
end

% Rotate C, to align normal vectors in XZ plane
H1 = eye(4);
H1(1:3,1:3) = rot_z(-(Rnew(1,i)));%- pi/2)); % Rotate
    % XYZ Toolpaths

% Apply HTMs to individual matrices
Pnew(:,i) = H1*P(:,i);
Qnew(:,i) = H1*Q(:,i);
Tnew(:,i) = H1*T(:,i);
%       Vnew(:,i) = H1*V(:,i);
nnew(:,i) = H1*n(:,i);
```

```
    % Calculate B axis position, and store in R matrix
    Rnew(2,i) = atan2(Tnew(1,i),Tnew(3,i));

    % HTM Generation: Collapse all Z axis data to the zero
        % plane
    H2 = eye(4);
    H2(1:3,4) = -L;

    % HTM Generation: B rotation of coordinates
    H3 = eye(4);
    H3(1:3,1:3) = rot_y(-Rnew(2,i));

    % HTM Generation: Undo Z collapse
    H4 = eye(4);
    H4(1:3,4) = L;

    % Apply the above HTMs
    Pnew(:,i) = H4*H3*H2*Pnew(:,i);
    Qnew(:,i) = H4*H3*H2*Qnew(:,i);
    Tnew(:,i) = H3*Tnew(:,i);
    nnew(:,i) = H3*nnew(:,i);

end
```

#### A.4.6.1 Rotation Matrix for Rotation About Z

```
function [Rz] = rot_z(A)
% Jesse Groover
```

```
% 17 March 2018
% This function provides the 3x3 rotation matrix for a
% rotation about Z.

Rz = [cos(A) -sin(A) 0;
    sin(A) cos(A) 0;
    0 0 1];
```

A.4.6.2 Rotation Matrix for Rotation About Y

```
function [Ry] = rot_y(A)
% Jesse Groover
% 17 March 2018
% This function provides the 3x3 rotation matrix for a
% rotation about Y.

Ry = [cos(A) 0 sin(A);
    0 1 0;
    -sin(A) 0 cos(A)];
```

A.4.7 Post Processor (For Writing to Text File)

```
function post_vectorial(Q,R,W,Fcorr,Nnum,name,process,...
    tooldesc,toolnum,S,Flin,Flimit,coolant)
% Jesse Groover
% 10 January 2018
% Updated 12 February 2018
% Updated 26 June 2018
% This function takes the P matrix, which contains toolpath
% information,and writes it to the desired textfile in a
```

```
% complete operation format, including tool change, spindle
% start/stop, etc.
%
% The format of the P matrix is as follows: SUPERSEDED
% P = [X coordinates ...                                    (1)
%       Y coordinates ...                                   (2)
%       Z coordinates ...                                   (3)
%       C coordinates ...                                   (4)
%       B coordinates ...                                   (5)
%       Feedrate values ...                                 (6)
%       Extra space 1 (I)...                                (7)
%       Extra space 2 (J)...                                (8)
%       Flag operation type (1-G01, 2-G02, 3-G03)...        (9)
%       Flag feed type (1-G93, 2-G94, 3-G00)...             (10)
%       Flag abs or inc (1-G90, 2-G91)...                   (11)
%       Ones...];                                           (12)
%
% UPDATE: This function now takes the Q matrix (after
% coordinate transformation), the R matrix (also after
% coordinate transformation), and the W matrix, and writes
% them to the appropriate NC file.

fprintf(name, 'N%3.0f (%s);\r\n',Nnum,process);
% fprintf(name,'(%s);\r\n',tooldesc);

toolchange(toolnum,toolnum,tooldesc,name); % Tool change
spindlefwd(name,S,coolant); % Turn spindle on, and coolant
```

```
    % if desired
% Process the actual operation toolpaths
post_operation_vectorial(name,Q,R,W,Fcorr,Flin,Flimit);
spindleoff(name); % Spindle off
```

#### A.4.7.1 Tool Change

```
function toolchange(toolnum,Hval,tooldesc,name)
% Jesse Groover
% 10 January 2018
% This function commands a tool change, and applies the tool
% length offset.

fprintf(name,'(%s);\r\n',tooldesc);        % Tool description
    % string
fprintf(name,'G91 G28 Z0;\r\n');           % Zero Return Z
fprintf(name,'T%.0f M6;\r\n',toolnum);     % Change tool
fprintf(name,'G43 H%.0f;\r\n',Hval);       % Apply tool length
    % offset
```

#### A.4.7.2 Spindle On, Forward Direction

```
function spindlefwd(name,S,coolant)
% Jesse Groover
% 17 October 2018
% This function writes the command to turn the spindle on at
% the specified spindle speed to the NC file.
fprintf(name,'M3 S%.0f;\r\n',S);
if coolant == 'Yes'
    fprintf(name,'M08;\r\n');
```

```
else
end
```

A.4.7.3 Write Each Individual Motion

```
function post_operation_vectorial(name,Q,R,W,Fcorr,...
    Flin,Flimit)
% Jesse Groover
% 12 February 2018
% This function takes the Q, R and W matrices, which contain
% all the toolpath information, and writes that information
% to the desired text file.

% Initalize Status Bar
f = waitbar(0,'Outputting to Text File');

for j = 1:numel(Q(1,:))
    % Update Status Bar
    status = j/numel(Q(1,:));
    waitbar(status,f);

    if j == 1 % The very first line
        % Operation Type
        switch W(1,j)
            case 0; fprintf(name,'G00 '); % Rapid
            case 1; fprintf(name,'G01 '); % Linear
            case 2; fprintf(name,'G02 '); % Linear, CW
            case 3; fprintf(name,'G03 '); % Linear, CCW
```

```
    otherwise
end
% Feedrate Type
switch W(2,j)
    case 93; fprintf(name,'G93 '); % Inverse Time
    case 94; fprintf(name,'G94 '); % Conventional
    case 95; fprintf(name,'G95 '); % Feed per
        % Revolution
    otherwise
end
% Abs or Inc
switch W(3,j)
    case 90; fprintf(name,'G90 '); % Absolute
    case 91; fprintf(name,'G91 '); % Incremental
    otherwise
end
% Positional Commands
fprintf(name,'X%.3f Y%.3f Z%.3f C%.3f B%.3f',...
    Q(1,j),Q(2,j),Q(3,j),R(1,j)*180/pi,...
    R(2,j)*180/pi);
% Feedrate
if W(1,j) ~= 0 % If not a rapid...
    fprintf(name,'F%.0f;\r\n',Flin); % ...write a
        % feedrate
else % If it is a rapid...
    fprintf(name,';\r\n'); % ...End of block
end
```

```
elseif j ~= 1 % All subsequent lines
    % Operation Type
    if W(1,j) ~= W(1,j-1)
        switch W(1,j)
            case 0; fprintf(name,'G00 '); % Rapid
            case 1; fprintf(name,'G01 '); % Linear
            case 2; fprintf(name,'G02 '); % Circular, CW
            case 3; fprintf(name,'G03 '); % Circular, CCW
        end
    else
    end
    % Feedrate Type
    if W(2,j) ~= W(2,j-1)
        switch W(2,j)
            case 93; fprintf(name,'G93 '); % Inverse Time
            case 94; fprintf(name,'G94 '); % Conventional
            case 95; fprintf(name,'G95 '); % Feed per
              % Revolution
        end
    else
    end
    % Absolute or Incremental
    if W(3,j) ~= W(3,j-1)
        switch W(3,j)
            case 90; fprintf(name,'G90 '); % Absolute
            case 91; fprintf(name,'G91 '); % Incremental
```

```
        end
    else
    end

    % Positional Commands
    % X
    if W(3,j) == 90 && Q(1,j) ~= Q(1,j-1)
        fprintf(name,'X%.3f ',Q(1,j));
    elseif W(3,j) == 91
        fprintf(name,'X%.3f ',Q(1,j));
    else
    end
    % Y
    if W(3,j) == 90 && Q(2,j) ~= Q(2,j-1)
        fprintf(name,'Y%.3f ',Q(2,j));
    elseif W(3,j) == 91
        fprintf(name,'Y%.3f ',Q(2,j));
    else
    end
    % Z
    if W(3,j) == 90 && Q(3,j) ~= Q(3,j-1)
        fprintf(name,'Z%.3f ',Q(3,j));
    elseif W(3,j) == 91
        fprintf(name,'Z%.3f ',Q(3,j));
    else
    end
    % C
```

```
if W(3,j) == 90 && R(1,j) ~= R(1,j-1)
    fprintf(name,'C%.3f ',R(1,j)*180/pi);
elseif W(3,j) == 91
    fprintf(name,'C%.3f ',R(1,j)*180/pi);
else
end
% B
if W(3,j) == 90 && R(2,j) ~= R(2,j-1)
    fprintf(name,'B%.3f ',R(2,j)*180/pi);
elseif W(3,j) == 91
    fprintf(name,'B%.3f ',R(2,j)*180/pi);
else
end

% Feedrate
if W(1,j) ~= 0 % If NOT Rapid
    if W(2,j) == 94 || W(2,j) == 95 % Conventional
            % or Feed/rev
        if W(2,j) ~= W(2,j-1) || W(2,j-1) ~= 94 ...
                || W(2,j-1) ~= 95 % Not the same as
                    % last time
            fprintf(name,'F%.3f;\r\n',Flin);
        else
            fprintf(name,';\r\n');
        end
    elseif W(2,j) == 93
        Finv = Flin*Fcorr(j);
```

```
                if Finv > Flimit
                    Finv = Flimit;
                else
                end
                fprintf(name,'F%.3f;\r\n',Finv);
            else
                fprintf(name,';\r\n');
            end
        else %
            fprintf(name,';\r\n');
        end

    else
    end
end

% Close status bar
close(f);
```

A.4.7.4 Spindle Off

```
function spindleoff(name)
% Jesse Groover
% 26 April 2018
% This function writes the commands to turn off the spindle
% and coolant, and writes an optional stop.

fprintf(name,'G00␣G90␣Z100.0;\r\n');
```

```
fprintf(name, 'M9;\r\n');
fprintf(name, 'M5;\r\n');
fprintf(name, 'M01;\r\n');
```

A.4.8 Preliminary Calculations

```
function [Rsafe1,Rsafe2,zlev,offset_N] = prelimcalcs(z,...
    Rinner,Router,ToolDia,safedist1,safedist2,b_total,b_step)
% Jesse Groover
% 24 October 2018
% This function performs some of the preliminary operations
% to cutting gear teeth by generation motion.

Rsafe1 = saferadius(z,Rinner,Router,ToolDia,safedist1);
Rsafe2 = saferadius(z,Rinner,Router,ToolDia,safedist2);

zlev = -1*(b_step:b_step:b_total);
if zlev(end) ~= -b_total
    zlev(end + 1) = -b_total;
end
offset_N = [0:2*pi/abs(z):2*pi] + pi/abs(z);
```

A.4.8.1 Safe Distance

```
function Rsafe = saferadius(z,Rinner,Router,tooldia,safedist)
% Jesse Groover
% 25 June 2018
% This function calculates the safe radius for the tool
% commanded position for both internal and external gears.
```

```
if sign(z) == 1
    Rsafe = Router + safedist + tooldia/2;
elseif sign(z) == -1
    Rsafe = Rinner - safedist - tooldia/2;
else
end
```

A.4.9 Vector of Offset Distances

```
function thvec = thicknessvec(z,Rb,Router,eta_b,beta_b,...
    ToolDia,ToolRI,stockclearance)
% Jesse Groover
% 18 October 2018
% This function generates a vector of the offset distances
% from the nominal involute.

thlarge = stockremovalmax(z,Rb,Router,0,eta_b,beta_b,...
    ToolDia); % Normal distance from nominal involute
thvec = stockclearance:ToolRI*ToolDia:thlarge; % Normal
    % distance from nominal involute
if thvec(end) ~= thlarge
    thvec(end+1) = thlarge;
end
thvec = vecsreverse(thvec);
```

A.4.9.1 Maximum Stock Thickness

```
function th = stockremovalmax(z,Rb,Ra,gamma,eta_b,...
    beta_b,tooldia)
% Jesse Groover
```

```
% 21 June 2018
% This function calculates the appropriate maximum stock
% thickness for both internal and external gears.

if sign(z) == 1 % External Gears
    th = Rb*cos(beta_b)*(tan(acos(Rb/Ra)) - acos(Rb/Ra)...
        + gamma);
elseif sign(z) == -1 % Internal Gears
    th = (Rb*eta_b - tooldia/2)*cos(beta_b);
end
```

#### A.4.9.2 Vector Reversal

```
function Pnew = vecsreverse(P)
% Jesse Groover
% 19 June 2018
% This function takes a matrix, and reverses the column
% order.

N = numel(P(1,:));

Pnew = zeros(size(P));

for i = 1:N
    Pnew(:,i) = P(:,(N-i+1));
end
```

### A.4.10 Generation of Tool Points and Orientations on Involute Flank

```
function [P,T,R,N,Q,W,Fcorr,fvec]...
```

```
    = VecGen_Complete(z,th,zlev,tooldia,beta_b,updn,...
    rollss,Rb,eta_b,offset_N,ImpAngles,Rsafe,Rsafe2,...
    increment,incstepsize,Ret)
% Jesse Groover
% 23 October 2018
% This function generates the tool centerpoints and
% orientations on a cylindrical involute gear.  All values
% are in part coordinates.

P = []; T = []; R = [];
N = []; Q = []; W = []; Fcorr = []; fvec = [];

% Initialize Status Bar
Numloops = abs(z)*numel(th)*numel(zlev);
count = 0;
stat = waitbar(0,'Generating Tool Paths');
for k = 1:abs(z) % Loop by tooth number

    for j = 1:numel(th) % Loop by Thickness
        d = th(j) + tooldia/(2*cos(beta_b)); % Total
            % Distance from nominal surface

        for i = 1:numel(zlev) % Loop by Z Height
            % Update Status Bar
            count = count + 1;
            status = count/Numloops;
            waitbar(status,stat);
```

```
% Generate Vectors - Side 1
f1 = sign(z)*1;
direc = f1*updn*sign(z); % Inward or Outward
    % (should be a function of up milling or
    % down milling)
rollssnew = roll_limits(z,rollss,Rb,d,zlev(i),...
    eta_b,beta_b,f1);
if direc == -1
    rollssnew = vecsreverse(rollssnew);
else
end
offset_nom = offset_N(k) + f1*ImpAngles(4);
offset1 = offset_axial(offset_nom,zlev(i),Rb,...
    beta_b);

% Approach - Use starting roll angle, and an
    % appropriate offset value
[Pnew,Tnew,Rnew,Nnew,Qnew,Wnew,Fnew,fnew] =...
    VecGen_Approach(z,Rb,Rsafe,d,rollssnew,...
    direc,f1,offset1,zlev(i),beta_b,Ret);
P = [P,Pnew]; % Nominal Points
T = [T,Tnew]; % Tool orientation vector
R = [R,Rnew]; % BC rotation positions
N = [N,Nnew]; % Surface normal vector, with
    % length applied
Q = [Q,Qnew]; % Toolpoint coordinates
```

```
W = [W,Wnew]; % Post processing parameters
Fcorr = [Fcorr,Fnew]; % Feedrate correction
    % factors
fvec = [fvec,fnew]; % Flag for side of gear

% Actual Involute, side 1
[Pnew,Tnew,Rnew,Nnew,Wnew,Fnew,fnew] =...
    VecGen_OnePass(z,rollssnew,Rb,d,zlev(i),...
    beta_b,f1,direc,increment,incstepsize,...
    offset1);
P = [P,Pnew]; % Nominal Points
T = [T,Tnew]; % Tool orientation vector
R = [R,Rnew]; % BC rotation positions
N = [N,Nnew]; % Surface normal vector, with
    % length applied
Qnew = Pnew + Nnew;
Q = [Q,Qnew]; % Toolpoint coordinates
W = [W,Wnew]; % Post processing parameters
Fcorr = [Fcorr,Fnew]; % Feedrate correction
    % factors
fvec = [fvec,fnew]; % Flag for side of gear

% Prep for side 2
f2 = sign(z)*-1;
direc = f2*updn*sign(z); % Inward or Outward
    % (should be a function of up milling or
        % down milling)
```

```
rollssnew = roll_limits(z,rollss,Rb,d,...
    zlev(i),eta_b,beta_b,f2);
if direc == -1
    rollssnew = vecsreverse(rollssnew);
else
end
offset_nom = offset_N(k) + f2*ImpAngles(4);
offset2 = offset_axial(offset_nom,zlev(i),Rb,...
    beta_b);

% Transition
[Pnew,Tnew,Rnew,Nnew,Qnew,Wnew,Fnew,fnew] =...
    VecGen_Transition(z,Rb,Rsafe2,d,rollss,...
    direc,f1,f2,offset1,offset2,zlev(i),beta_b);
P = [P,Pnew]; % Nominal Points
T = [T,Tnew]; % Tool orientation vector
R = [R,Rnew]; % BC rotation positions
N = [N,Nnew]; % Surface normal vector, with
    % length applied
Q = [Q,Qnew]; % Toolpoint coordinates
W = [W,Wnew]; % Post processing parameters
Fcorr = [Fcorr,Fnew]; % Feedrate correction
    % factors
fvec = [fvec,fnew]; % Flag for side of gear

% Generate Vectors - Side 2
[Pnew,Tnew,Rnew,Nnew,Wnew,Fnew,fnew] = ...
```

```
    VecGen_OnePass(z,rollssnew,Rb,d,zlev(i),...
    beta_b,f2,direc,increment,incstepsize,...
    offset2);
P = [P,Pnew]; % Nominal Points
T = [T,Tnew]; % Tool orientation vector
R = [R,Rnew]; % BC rotation positions
N = [N,Nnew]; % Surface normal vector, with
    % length applied
Qnew = Pnew + Nnew;
Q = [Q,Qnew]; % Toolpoint coordinates
W = [W,Wnew]; % Post processing parameters
Fcorr = [Fcorr,Fnew]; % Feedrate correction
    % factors
fvec = [fvec,fnew]; % Flag for side of gear

% Retract
[Pnew,Tnew,Rnew,Nnew,Qnew,Wnew,Fnew,fnew] = ...
    VecGen_Retract(z,Rb,Rsafe,d,rollssnew,...
    direc,f2,offset2,zlev(i),beta_b,Ret);
P = [P,Pnew]; % Nominal Points
T = [T,Tnew]; % Tool orientation vector
R = [R,Rnew]; % BC rotation positions
N = [N,Nnew]; % Surface normal vector, with
    % length applied
Q = [Q,Qnew]; % Toolpoint coordinates
W = [W,Wnew]; % Post processing parameters
Fcorr = [Fcorr,Fnew]; % Feedrate correction
```

```
                % factors
            fvec = [fvec,fnew]; % Flag for side of gear

        end

    end

end

% Close status bar
close(stat);
```

A.4.10.1 Roll Angle Limits

```
function rollssnew = roll_limits(z,rollss,Rb,d,zlev,...
    eta_b,beta_b,f)
% Jesse Groover
% 21 June 2018
% This function calculates the new roll angle limits for
% both internal and external gears.  This function in
% particular contains the logic whereby other functions
% actually calculate the particular limits. Currently, the
% function roll_lim_spacecent is used, which uses a brute
% force method to determine the roll angle limit.

% Multiplication Factor for final roll angle, for external
    % gears only.  Ensures tool fully comes off tip of tooth.
ExtraFactor = 1.1;
```

```
if sign(z) == 1 % External gear
    rollssnew(1) = max(roll_lim_spacecent(z,Rb,d,eta_b),...
        rollss(1));
    rollssnew(2) = ExtraFactor*rollss(2);
elseif sign(z) == -1 % Internal gear
    rollssnew(1) = rollss(1);
    % Subtract from end point
    rollssnew(2) = min([rollss(2),...
        roll_lim_spacecent(z,Rb,d,eta_b)]);

    % Limit outer roll angle for inaccessible flank
    if beta_b > 0 && f < 0 % Pos helix, RH flank
        rollssnew(2) = rollssnew(2) + zlev*...
            abs(tan(beta_b))/Rb;
    elseif beta_b < 0 && f > 0 % Neg helix, LH flank
        rollssnew(2) = rollssnew(2) + zlev*...
            abs(tan(beta_b))/Rb;
    else
    end
end
```

#### A.4.10.2 Roll Angle at Space Width Half Angle

```
function xi = roll_lim_spacecent(z,Rb,Nxy,eta_b)
% Jesse Groover
% 21 June 2018
% This function uses brute force iteration to find the roll
```

```
% limit for the given constraints.

% xi + sign(z)*eta_b = atan(xi + sign(z)*Nxy/Rb)

xi = 0;
dxi = 0.001;
eps = 0.001;
countlim = 10000;

delta = abs((xi + sign(z)*eta_b) - atan(xi + sign(z)*Nxy/Rb));

if sign(z) == 1 && Nxy/Rb <= eta_b
    xi = 0;
elseif sign(z) == -1 && Nxy/Rb >= eta_b
    xi = 0;
else
    count = 0;
    while delta > eps
        xi = xi + dxi;
        delta = abs((xi + sign(z)*eta_b) - atan(xi...
            + sign(z)*Nxy/Rb));
        count = count + 1;
        if count > countlim
            fprintf('Limit Reached.\n\n');
            break
        end
    end
```

```
end
```

#### A.4.10.3 Offset Angle due to Helix Angle

```
function offset_ax = offset_axial(offset_N,zlev,Rb,beta_b)
% Jesse Groover
% 25 June 2018
% This function takes an initial offset angle, the base
% circle, the helix angle at the base circle, and the
% zlevel, and calculates the corrected offset angle to
% account for the helix angle at each Z height.

offset_ax_mod = (zlev./Rb).*tan(beta_b);

offset_ax = offset_N + offset_ax_mod;
```

#### A.4.10.4 Approach

```
function [Pnew,Tnew,Rnew,Nnew,Qnew,Wnew,Fnew,fnew]...
    = VecGen_Approach(z,Rb,Rsafe,dist,rollss,direc,f,...
    offset,zlev,beta_b,Ret)
% Jesse Groover
% 23 June 2018
% This function generates the appropriate vector matrices
% for the approaching motions to cut an involute.

P = [Pvec(Rb,rollss(1),offset,Ret,f),Pvec(Rb,rollss(1),...
    offset,zlev,f)]; % Nominal Points
n = [nvec(z,rollss(1),offset,beta_b,f),nvec(z,rollss(1),...
    offset,beta_b,f)]; % Surface unit normal vector
```

```
T = [Tvec(rollss(1),offset,beta_b,f),Tvec(rollss(1),...
    offset,beta_b,f)]; % Tool orientation vector
% BC rotation positions
R = [zeros(2,numel(rollss(1))),zeros(2,numel(rollss(1)))];

N = dist.*n; % Surface normal vector, with length applied

Qnom = P + N;
Rad = sqrt(Qnom(1,1)^2 + Qnom(2,1)^2);
Q1 = (Rsafe/Rad)*Qnom(1:2,:);
Q1(3,:) = Qnom(3,:);

Pnew = [P]; %,P];
Nnew = [N]; %,N];
Tnew = [T]; %,T];
Rnew = [R]; %,R];

Qnew = [Q1]; %,Qnom];

Wnew = [0, 0; % 1;
    94, 94; % 94;
    90, 90]; % 90];

Fnew = [0,0]; % 1];

fnew = [f,f];
```

A.4.10.5 Cutting an One Pass of an Involute Surface

```
function [P,T,R,N,W,F,fnew] = ...
    VecGen_OnePass(z,rollss,Rb,d,zlev,beta_b,f,direc,...
    increment,stepsize,offset)
% Jesse Groover
% 23 October 2018
% This function generates the tool center points and
% orientations for cutting a single involute flank at a
% single axial (z) height.

if increment == 'Arc Length'
    [arc] = convert_roll2arc(Rb,rollss);
    rollvec = invstep_arclength(arc(1), arc(2), stepsize,...
        0,Rb);
elseif increment == 'Roll Angle'
    stepsize = stepsize*pi/180; % Convert the step size in
        % degrees to radians
    rollvec = invstep_rollangle(rollss(1),rollss(2),...
        stepsize,0);
end



% Generate Vectors
P = Pvec(Rb,rollvec,offset,zlev,f); % Nominal Points
n = nvec(z,rollvec,offset,beta_b,f); % Surface unit normal
    % vector
```

```
T = Tvec(rollvec,offset,beta_b,f); % Tool orientation vector
R = zeros(2,numel(rollvec)); % BC rotation positions

N = d.*n; % Surface normal vector, with length applied

W = [1, ones(1,numel(rollvec)-1);
     94, 93*ones(1,numel(rollvec)-1);
     90, 90*ones(1,numel(rollvec)-1)]; % abs or inc

F = [1,zeros(1,numel(P(1,:))-1)];

for i = 2:numel(P(1,:))
    F(i) = 1/(Rb*(abs(rollvec(i) - rollvec(i-1)))*...
        sqrt(2 - 2*sqrt((rollvec(i) + sign(z)*d/Rb)^2 + 1)*...
        cos(atan(rollvec(i) + sign(z)*d/Rb)) +...
        (rollvec(i) + sign(z)*d/Rb)^2));
end

fnew = f*ones(1,numel(rollvec));
```

#### A.4.10.6 Converting from Roll Angle to Arc Length

```
function [arc] = convert_roll2arc(Rb, roll)
% Jesse Groover
% 19 March 2018
% This function converts roll angle to arc length

arc = Rb.*(roll.^2)./2;
```

#### A.4.10.7 Generating an Array of Roll Angle Values with Equal Arc Length Increments

```
function C = invstep_arclength(start, stop, step, offset,Rb)
% Jesse Groover
% 9 January 2018
% This function calculates machine C-axis positions for an
% involute, starting at the base circle and traveling
% outward to some point. The increments are calculated by
% equal arc length increments, then converted to a roll
% angle.
%
% Syntax:
% C = invstep_arclength(start,stop,step,offset,Rb)
% start - The starting arc length from the base circle,
    % in mm
% stop - The ending arc length from the base circle, in mm
% step - Arc length step size, in mm
% offset - The starting machine C axis position at the base
    % circle, in rad
% Rb - Base radius, in mm

if stop < start
    step = -step;
else
end
```

```
S = start:step:stop; % Arc length positions in mm

C = (offset + sqrt(2.*S./Rb)); % C coordinates (roll angle)
    % in rad

if C(end) ~= offset + sqrt(2*stop/Rb)
    C(end+1) = offset + sqrt(2*stop/Rb);
else
end
```

A.4.10.8 Generating an Array of Roll Angle Values with Equal Roll Angle Increments

```
function C = invstep_rollangle(startroll,endroll,step,...
    offset)
% Jesse Groover
% 9 January 2018
% This function calculates machine C-axis positions for an
% involute, starting at the base circle and traveling
% outward to some point. The increments are calculated by
% equal roll angle increments.
%
% Syntax:
% C = invstep_rollangle(start,stop,step,offset)
% start - The starting arc length from the base circle,
    % in mm
% stop - The ending arc length from the base circle, in mm
% step - Arc length step size, in mm
```

```
% offset - The starting machine C axis position at the base
    % circle, in rad

if endroll < startroll
    step =- step;
else
end

C = (offset + startroll:step:endroll); % C axis positions
    % in rad

if C(end) ~= (offset+endroll)
    C(end+1) = offset+endroll;
else
end
```

A.4.10.9 Transitioning from One Side to the Other

```
function [Pnew,Tnew,Rnew,Nnew,Qnew,Wnew,Fnew,fnew] = ...
    VecGen_Transition(z,Rb,Rsafe,dist,rollss,direc,f1,f2,...
    offset1,offset2,zlev,beta_b)
% Jesse Groover
% 23 June 2018
% This function generates the appropriate vector matrices
% for the approaching motions to cut an involute.

if sign(z) == 1
    Pnew = [];
```

```
    Nnew = [];
    Tnew = [];
    Rnew = [];

    Qnew = [];
    Wnew = [];
    Fnew = [];
    fnew = [];
elseif sign(z) == -1
    % Retract
    P = Pvec(Rb,rollss(1),offset1,zlev,f1); % Nominal Points
    n = nvec(z,rollss(1),offset1,beta_b,f1); % Surface unit
        % normal vector
    T = Tvec(rollss(1),offset1,beta_b,f1); % Tool
        % orientation vector
    R = zeros(2,numel(rollss(1))); % BC rotation positions

    Qnom = P + dist*n;
    Rad = sqrt(Qnom(1)^2 + Qnom(2)^2);
    Q1 = Qnom*Rsafe/Rad;

    N = dist.*n; % Surface normal vector, with length
        % applied

    Pnew = [P];
    Nnew = [N];
    Tnew = [T];
```

```
Rnew = [R];

Qnew = [Q1];
Wnew = [0;
    94;
    90];
Fnew = [0];
fnew = [f1];

% Approach
P = Pvec(Rb, rollss(1), offset2, zlev, f2); % Nominal Points
n = nvec(z, rollss(1), offset2, beta_b, f2); % Surface unit
    % normal vector
T = Tvec(rollss(1), offset2, beta_b, f2); % Tool
    % orientation vector
R = zeros(2, numel(rollss(1))); % BC rotation positions

Qnom = P + dist*n;
Rad = sqrt(Qnom(1)^2 + Qnom(2)^2);
Q1 = Qnom*Rsafe/Rad;

N = dist.*n; % Surface normal vector, with length
    % applied

Pnew = [Pnew, P];
Nnew = [Nnew, N];
Tnew = [Tnew, T];
```

```
    Rnew = [Rnew,R];

    Qnew = [Qnew,Q1];
    Wnew = [Wnew,[0;94;90]];

    Fnew = [Fnew,0];
    fnew = [fnew,f2];

else
end
```

A.4.10.10 Retracting

```
function [Pnew,Tnew,Rnew,Nnew,Qnew,Wnew,Fnew,fnew] = ...
    VecGen_Retract(z,Rb,Rsafe,dist,rollss,direc,f,offset,...
    zlev,beta_b,Ret)
% Jesse Groover
% 23 June 2018
% This function generates the appropriate vector matrices
% for the approaching motions to cut an involute.

P = [Pvec(Rb,rollss(2),offset,zlev,f),Pvec(Rb,rollss(2),...
    offset,Ret,f)]; % Nominal Points
n = [nvec(z,rollss(2),offset,beta_b,f),nvec(z,rollss(2),...
    offset,beta_b,f)]; % Surface unit normal vector
T = [Tvec(rollss(2),offset,beta_b,f),Tvec(rollss(2),...
    offset,beta_b,f)]; % Tool orientation vector
% BC rotation positions
```

```
R = [zeros(2,numel(rollss(2))),zeros(2,numel(rollss(2)))];

N = dist.*n; % Surface normal vector, with length applied

Qnom = P + N;
Rad = sqrt(Qnom(1,1)^2 + Qnom(2,1)^2);
Q1 = (Rsafe/Rad)*Qnom(1:2,:);
Q1(3,:) = Qnom(3,:);

Pnew = [P];
Nnew = [N];
Tnew = [T];
Rnew = [R];

Qnew = [Q1];
if sign(z) == 1
    Wnew = [0,0;
        94,94;
        90,90];
    Fnew = [0,0];
elseif sign(z) == -1
    Wnew = [1,1;
        94,94;
        90,90];
    Fnew = [1,0];
else
end
```

```
fnew = [f,f];
```

A.4.10.11 P Vector

```
function P = Pvec(Rb,rollvec,offset,zlev,f)
% Jesse Groover
% 24 May 2018
% This function generates the nominal involute points, in
% part coordinates.

% Nominal Points
P = [Rb.*(cos(f*rollvec + offset) + f*rollvec.*...
     sin(f*rollvec + offset));
     Rb.*(sin(f*rollvec + offset) - f*rollvec.*...
     cos(f*rollvec + offset));
     zlev*ones(1,numel(rollvec))];
```

A.4.10.12 n Vector

```
function n = nvec(z,rollvec,offset,beta_b,f)
% Jesse Groover
% 24 May 2018
% This function generates the unit normal vectors for an
% involute, in part coordinates.

% Surface unit normal vector
n = f*sign(z)*[sin(f*rollvec + offset).*cos(beta_b);
     -cos(f*rollvec + offset).*cos(beta_b);
     sin(beta_b)*ones(1,numel(rollvec))];
```

#### A.4.10.13 T Vector

```
function T = Tvec(rollvec,offset,beta_b,f)
% Jesse Groover
% 24 May 2018
% This function generates the tool orientation vector for an
% involute, in part coordinates

% Tool orientation vector
T = [-sin(f*rollvec + offset)*sin(beta_b);
    cos(f*rollvec + offset)*sin(beta_b);
    cos(beta_b)*ones(1,numel(rollvec))];
```

### A.4.11 Reference Ring Operation (Internal Only)

```
function operation_refring(R_outer, R_inner, Depth_Total,...
    Depth_Inc,Ret,ToolName,ToolNumber,ToolDia,ToolLength,...
    ToolRI,S,Flin,coolant,Nnum,name)
% Jesse Groover
% 26 October 2018
% This function writes the commands for a round ring to be
% machined, using the rotary (C) axis.

% Generate radius values
Rvec = (R_inner + ToolDia/2):ToolDia*ToolRI:(R_outer -...
    ToolDia/2);
if Rvec(end) ~= R_outer - ToolDia/2
    Rvec(end+1) = R_outer - ToolDia/2;
end
```

```
% Error message if tool not long enough
if Depth_Total <= ToolLength
    lim = Depth_Total;
else
    lim = ToolLength;
    error = strcat('The desired cutting depth is larger',...
        ' than the extension of the tool.  The cut will',...
        ' only extend to',sprintf(' %.3f mm.',lim));
    msgbox(error,'Error');
end

% Generate Z depth values
zstart = -Depth_Inc;
zpos = zstart:-Depth_Inc:-lim;
if zpos(end) ~= -lim
    zpos(end + 1) = -lim;
end

% Approach rate, mm/min
Fapp = 1000;

% C approach angle, deg
Capp = 45;

% G-Code
% Prep
```

```
fprintf(name,'\r\nN%.0f',Nnum);
    fprintf(name,' (Reference Cylinder Operation);\r\n');
fprintf(name,'(Zero Return Z, Tool change, apply length');
    fprintf(name,' offset);\r\n');
fprintf(name,'(%s);\r\n',ToolName);
fprintf(name,'G00 G91 G28 Z0;\r\n'); % Zero-Return Z-axis
fprintf(name,'G00 G91 G28 X0 Y0;\r\n'); % Zero-Return X and
    % Y Axes
fprintf(name,'T%.0f M06;\r\n',ToolNumber); % Tool change
fprintf(name,'G43 H%.0f;\r\n',ToolNumber); % Apply length
    % offset
fprintf(name,'M11;\r\nM69;\r\n'); % Unclamp B and C Axes
fprintf(name,'G90 B0 C0;\r\n'); % Go to zero point for B and
    % C axis
fprintf(name,'M68;\r\n'); % Reclamp B Axis

% Operation
for i = 1:numel(zpos)
    Frot = (Flin/Rvec(1))*(180/pi); % Rotary feedrate, in
        % deg/min
    if i == 1 % First time around
        fprintf(name,'(Approach and spindle and coolant');
            fprintf(name,' on);\r\n');
        % Position for first iteration
        fprintf(name,'G00 G90 X0 Y%.3f C0;\r\n',Rvec(1));
        % Go to Retract height
        fprintf(name,'G01 G90 Z%.3f F%.3f;\r\n',Ret,Fapp);
```

```
        fprintf(name,'M03 S%.3f;\r\n',S); % Spindle on
        if coolant == 'Yes'
            fprintf(name,'M08;\r\n'); % Flood Coolant On
        else
        end
        fprintf(name,'G01 G90 Z%.3f C%.3f F%.3f;\r\n',...
            zpos(1),Capp,Frot);
    else % All other times
        fprintf(name,'G01 G90 X0 Y%.3f C0 F%.3f;\r\n',...
            Rvec(1),Flin); % Reposition for next iteration
        fprintf(name,'G01 G90 Z%.3f C%.3f F%.3f;\r\n',...
            zpos(i),Capp,Frot); % Plunge
    end

    for j = 1:numel(Rvec)
        Frot = (Flin/Rvec(j))*(180/pi); % Rotary feedrate,
            % in deg/min
        % Rotate table
        fprintf(name,'G01 G91 C380.0 F%.3f;\r\n',Frot);
        if j ~= numel(Rvec) % If not on the last iteration,
                % walk out to next radius
            fprintf(name,'G01 G90 Y%.3f F%.3f;\r\n',...
                Rvec(j),Flin); % Walk forward
        end
    end
end
```

```
% Retracting
fprintf(name,...
    '(Retract, spindle and coolant off, zero return');
    fprintf(name,' Z);\r\n');
fprintf(name,'G00 G90 Z%.3f;\r\n',Ret); % Retract
fprintf(name,'M05;\r\n'); % Spindle off
fprintf(name,'M09;\r\n'); % Coolant off
fprintf(name,'G91 G28 Z0;\r\n'); % Zero Return Z Axis
fprintf(name,'G91 G28 X0 Y0;\r\n'); % Zero Return X and Y
    % Axes
fprintf(name,'G91 G28 C0;\r\n'); % Zero Return C Axis
fprintf(name,'M01;\r\n'); %

%fclose(name);
end
```

### A.4.12 Footer

```
function footer(name)
% Jesse Groover
% 28 October 2018
% This function writes the final commands to the NC file.
% The spindle and coolant are turned off, all axes returned
% to home position, the rotary axes clamped, and the program
% ended.

fprintf(name,'(FOOTER - Retract, finish program);\r\n');
% fprintf(name,'G94 G00 Z%.3f;\r\n',retract); % Retract
```

```
fprintf(name, 'M09;\r\n'); % Coolant off
fprintf(name, 'M05;\r\n'); % Spindle off
fprintf(name, 'G91 G28 Z0;\r\n'); % Home Z Axis
fprintf(name, 'G91 G28 X0 Y0 C0 B0;\r\n'); % Home other axes
fprintf(name, 'M10;\r\n'); % Reclamp C Axis
fprintf(name, 'M68;\r\n'); % Reclamp B Axis
fprintf(name, 'M30;\r\n'); % End Program
fprintf(name, '%%\r\n');
end
```

# VITA

Jesse Michael Groover was born on 23 July, 1992, in Raleigh North Carolina, to parents David and Denise Groover. The second of eight children, he was home-schooled through high school, and spent his youth playing the bluegrass banjo, and piddling with his father's woodworking tools in their home shop. In 2016, he graduated with his Bachelors Degree in Mechanical Engineering from the University of North Carolina at Charlotte. In December 2018, he plans to graduate with his Masters Degree in Mechanical Engineering from the same university.

He is survived by numerous family and friends, and this thesis.