The Pennsylvania State University

The Graduate School

College of Engineering

UNDERSTANDING THE INFLUENCE OF PARAMETER MODULATION IN MACHINING-BASED PROCESSING

A Thesis in
Mechanical Engineering
by
Joshua M. Norman

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2013

The thesis of Joshua M. Norman was reviewed and approved* by the following:

Christopher J. Saldana
Assistant Professor of Industrial Engineering
Thesis Adviser

Edward C. De Meter
Professor of Industrial and Mechanical Engineering
Thesis Reviewer

Karen Thole
Professor of Mechanical Engineering
Department Head of Mechanical and Nuclear Engineering

*Signatures are on file in the Graduate School

Abstract

It is widely believed that the parameters of cutting speed and undeformed chip thickness can be used to influence the specific energy consumption of traditional machining processes. The purpose of this study was to further investigate the degree to which these two parameters affect the system load response (e.g. force, power consumption) – and in turn the specific energy consumption – when they are applied quasi-statically as well as dynamically. This was carried out experimentally using a CNC planer (to study the effects of cutting velocity) and a CNC lathe (to study the effects of undeformed chip thickness). Furthermore, this study purposed to ascertain whether the loading response of a system to dynamic parameter manipulation can be predicted based on a characterization of the system's response to static changes in the same parameter. This involved discretizing the time-varying cases and treating them as a continuous series of static cutting instances, each of which could then be assigned a predicted force value. Obviously, such a technique would only be effective if there is close agreement between the static and dynamic loading responses of the given system.

As a result of empirical testing, close agreement was found between the loading responses for static and dynamic variations in cutting speed for both AA6061-T6 and OFE Copper. When similar variations were applied to AA6061-T6 and Ti3Al2.5V in the direction of undeformed chip thickness, close agreement between static and dynamic loading responses was again found, but for low to moderate values of the processing parameter. These observations suggest that, for many materials, the nature of material response in low-frequency, modulation-assisted machining (MAM) is similar to that which occurs in conventional machining. Based on this similarity, it was also found that the general shape of the characterized load response to static parameter manipulation could be used to predict whether MAM would consume more or less specific energy than its conventionally cut counterpart. Specifically, MAM will generally require slightly more energy for dynamic changes in cutting velocity but significantly less energy for dynamic changes in undeformed chip thickness, as compared to conventional cutting.

It is suggested that further research be performed which incorporates dynamic changes in cutting speed and rake angle into the energy prediction model for MAM as applied in the direction of undeformed chip thickness. Finally, a further study may also make use of an intermediate variable, such as thickness of the primary deformation zone, as the predicted quantity for the dynamic cases. Such might provide insight as to the existence of a common

physical mechanism underlying the energetics of both modulated machining configurations and would act as an extension of the research performed by Kececioglu [2,14].

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to extend my truest thanks to my adviser, Dr. Saldana, whose wisdom and guidance over the past year have been paramount to the execution of this project. I also want to recognize the contributions of labmate Cesar Moreno, and support specialists Dan Supko, Randy Wells, and David Shelleman to the design, fabrication, and testing of various components used in this study. My gratitude also goes out to Dr. De Meter for his extremely valuable input as a result of the reviewing process.

I also want to thank my wife, Kristi, for being an anchor of support and encouragement for me these past several years. Most importantly, I want to recognize my Lord and Saviour Jesus Christ–my faith in whom has sustained me throughout the course of my studies and has given me purpose both within as well as beyond the realm of academia.

Chapter 1: Introduction

Analyses of material removal processes such as turning, drilling and milling often involve consideration of the underlying deformation mechanics. Traditional machining methods are unique in that their modes of material removal require direct contact between the tool and workpiece. The behavior of this interaction affects many system parameters: notably, tool loading and wear. Such parameters determine the local energy consumption of the process, which in turn affects decisions regarding equipment selection and cost effectiveness.

Although the term “traditional machining” encompasses a variety of processes, the method of material removal is quite similar throughout. In each case, the cutting edge of a tool oriented at some rake angle (α) with a relief angle (β) engages a workpiece at a given cutting speed (V_{DC}) and undeformed chip thickness (h_0) giving rise to the formation of a narrow band of shearing which extends from the tool’s cutting edge to the surface of the workpiece at an angle φ (see Fig. 1.1) [1]. The thickness of this band may vary along its length and has been known in some cases to be thicker near the tool edge than near the surface of the workpiece [2]. In any case, this region is referred to as the primary deformation zone (PDZ) and is responsible for the majority of strain imposed on the workpiece throughout the process of chip formation [3-5]. The card deck analogy developed by Piispanen has long been applied in order to quantify the degree of deformation that occurs in the PDZ [6]. This strain is known to be a function of α and the ratio between the undeformed and deformed chip thicknesses h_0 and h_c, respectively [1]. The area at which the tool contacts the workpiece is relatively small and is known as the intimate contact region (ICR) [3,7,8]. The portion of the newly formed chip that has been in close proximity to the ICR will also undergo deformation as a result of tool/workpiece friction. This band has been referred to by some as the secondary deformation zone [3,4]. The strain imposed in these two zones is widely accepted as being responsible for the overwhelming majority of mechanical work required in the metal cutting process [1,5]. It is believed that the PDZ accounts for approximately three-fourths of this energy requirement and the secondary deformation zone is responsible for the remaining amount [1].

The processing parameters of undeformed chip thickness, cutting velocity, and rake angle are known to affect the magnitude of specific energy required for material removal [1,9-11]. This presents an important question: can changes in processing effect changes in the physical phenomena controlling energy consumption and if so, to what degree? Technologies exist which can modify such processing parameters both statically as well as dynamically. Many have observed that statically reducing the undeformed chip thickness results in a greater degree of specific energy consumption, known as the “size effect” in metal cutting [4,10,12,13]. Kececioglu believed that it was the size of the shear zone itself, rather than the more general variable of undeformed chip thickness, which was responsible for the size effect [14]. In any case, some have attributed this size-dependent phenomenon to be directly correlated with the number of defects in the material that will pass through the PDZ and facilitate slip plane formation [12,15]. Thus, smaller volumes of a given material logically contain fewer of such defects than do larger volumes. In a similar vein, cutting velocity—which directly correlates with strain rate—is also known to influence the mechanisms of deformation. Work performed by Kececioglu on this subject is well known. He determined that increases in strain rate correspond with increases in shear flow stress [2], and thus to a greater degree of energy consumption in the PDZ [1]. This strain-rate dependency is also referred to by Shaw who goes on to state that increases in temperature tend to have the opposite effect on flow stress. Furthermore, a counter-balancing effect has been observed in which high cutting speeds can elicit temperatures sufficient enough to weaken a material and bring about adiabatic shear in the PDZ – a phenomenon known as “thermal softening.” However, such is most commonly observed at cutting speeds on the order of some meters/sec [1], which is several orders of magnitude greater than those used in this study and is therefore not expected to be a significant factor here.

Experiments which indicate the existence of such phenomenon are typically performed by making multiple cuts in a material, each at a different value of the static parameter under investigation. However, such parameters can also be modified dynamically during the course of the cut. Considerable work in this regard has been performed by Chandrasekar et al through the use of modulation-assisted machining (MAM) techniques [3,7,16-21]. MAM has demonstrated an ability to provide a high level of control over dynamic tool displacement through the use of coupled linear and oscillatory motion. Because these motions are induced using multiple actuation systems, they can be applied along different axes. The relative orientation of these axes determines which processing parameter will be dynamic in nature.

1.1 Problem statement:

The purpose of the present study is to further investigate the effects that undeformed chip thickness and cutting velocity have on the magnitude of specific energy consumption for orthogonal cutting processes. Specifically, these parameters will be applied in varying degrees of magnitude both statically (through conventional cutting methods) as well as dynamically (by incorporating MAM) in order to assess their respective effects on the energy-related phenomena described above.

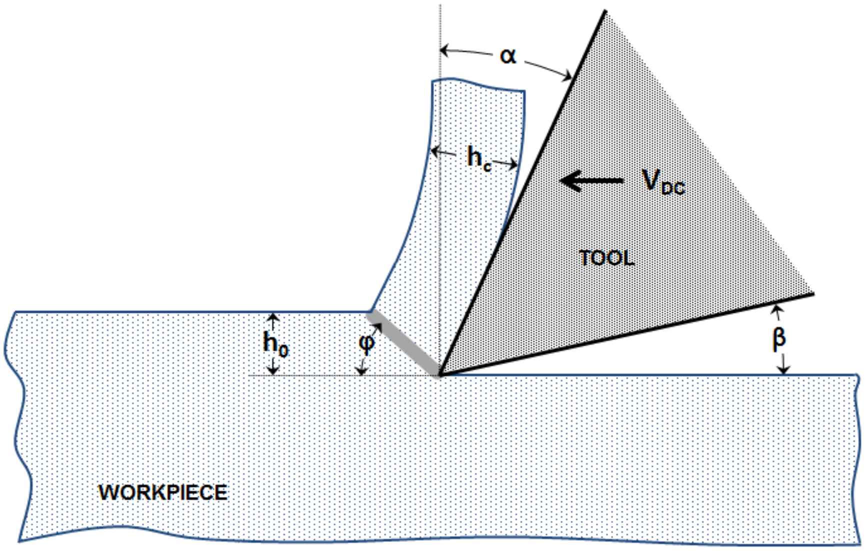


Figure 1.1: Schematic of orthogonal cutting and its commonly associated parameters.

Chapter 2: Background

The importance of processing parameters on the energetics of machining via the tool-workpiece interaction was discussed in the previous chapter. Conventional, or static parameter, cutting processes have long been studied [1,2,5,6,8,10,12]. Only in the last several decades has the technology been available which makes dynamic parameter manipulation not only possible, but effective. The intentional use of oscillatory tool displacement appears to have become popular following the work of Kumabe in 1979, though he was arguably not the first to use it [19,22,23]. Application of this method was initially focused on the high-frequency (>5 kHz) range, which became known as "vibration-assisted machining" (VAM). Joshi notes the difference between this and the related technique of MAM which accesses low-frequencies (<1 kHz) [23]. These methods most commonly make use of a 1D oscillation although they can be extended to 2D as was done by Shamoto in his well-known elliptical vibration studies [16,19,21,23,24]. The direction of the 1D motion is arbitrary, though two cases are of special interest to the present study, namely "velocity-direction" and "feed-direction." These two types of MAM affect the processing parameters of cutting velocity and undeformed chip thickness, respectively. Prior research performed on these techniques will now be discussed and a review of their underlying mechanics will be presented in the following chapter.

2.1 Velocity-direction MAM:

The tool-workpiece interface is a region characterized by high pressure and as such, is difficult to directly lubricate. A study performed by Moscoso et al provided visual proof that lubricant is not allowed access to the ICR during cutting [3]. However, by inducing a sinusoidal time-varying cutting velocity of sufficient magnitude, it has been shown that the tool will separate from the workpiece and thus allow lubricant to access this regime. Evidence of ICR lubrication was seen directly using a transparent tool visualization technique and indirectly by observation of the tool-work friction coefficient (see Fig. 2.1). The perturbative motion described here can be achieved through the use of a piezoelectric actuator and is often coupled with a linear motion provided by a motor [3]. It is important to note however that tool-work separation is only possible for cutting speeds that are less than the maximum modulation velocity. In many cases, the limited ability of the piezoelectric actuator renders velocity-direction MAM incapable of achieving material removal rates higher than 0.5 m/s [19].

Although some studies have suggested a positive correlation between cutting velocity and specific energy consumption, all of these have been performed using static parameter manipulation (e.g. multiple cuts each made at different cutting speeds). The experiment performed by Moscoso et al which employed a time-varying cutting speed cannot be compared to the conventional case without incorporating the effect of lubricant on the tool-work interaction. Thus, whereas specific advantages of velocity-direction MAM have been identified which may justify industry's acceptance of the process, the effects of its dynamic nature on the energetics of dry metal cutting are less adequately understood.

2.2 Feed-direction MAM:

Ductile metals such as aluminum, titanium, and nickel alloys present unique challenges to the machining industry due to their tendency to produce continuous chips [7]. This aspect is especially detrimental to the drilling process, as large chips are difficult to evacuate from the cutting site and thus promote tool breakage. In order to encourage chip removal, high-pressure fluids may be used to lower the temperature at the cutting site, break chips into smaller pieces, and evacuate them from the hole [7,25,26]. This method is effective, as tools used in the presence of such metal removal fluids (MRFs) have been found to last up to 250 times as many cycles. However, MRFs can be expensive; estimates have placed them at up to 16% of the total cost for some machining processes. Efforts have been made to remove the need for such fluids by instead applying various coatings to the tools. Such tools have been found to last approximately 100 times as many cycles as the baseline case. However, this technique is only advantageous under a narrow range of processing conditions [25].

Previous studies have demonstrated MAM's ability to form discrete chips when the direction of modulation is parallel to the undeformed chip thickness [7,18,19,21]. The application of this feed-direction MAM to the drilling process is analogous to the commonly used "peck-drilling" technique in which the machine controller periodically moves the tool away from the workpiece, effectively breaking the chip while simultaneously lubricating the ICR. Traditional peck drilling has its limits however. In the case of manual machines, this may come in the form of the operator's speed of motion; for CNC machines, it might be due to the inertial effects that must be overcome by the motor. Peck-drilling can be applied at a much higher rate through the use of a piezoelectric actuator positioned near the tool since its intrinsic kinematic and dynamic properties allow it to overcome both aforementioned limitations.

Although the discussion thus far has focused on drilling, feed-direction MAM can similarly be incorporated into the turning process. Consider a cutting tool used to continuously turn down the length of a tube for example. This could also result in discrete chips, depending on the ratio of the modulation frequency to the rotational frequency of the primary spindle. Although chip evacuation is less of a concern in such a case, feed-direction MAM has been used to produce metal particulates of controlled size and shape (see Fig. 2.2) [18,19,21]. Such particulate may be the intended product of the cutting operation, or may be a valuable byproduct. Furthermore, the mode by which the tool disengages from the workpiece in feed-direction MAM allows separation to be achieved even at high cutting speeds, making this case more viable from an industrial standpoint [19].

As was the case in velocity-direction MAM, studies have already been performed which demonstrate the ability of feed-direction MAM to carry out prescribed motions under load and to affect tool-work separation and cutting forces [7,16-18,21-23,27]. However, there again exists some gap in knowledge surrounding the subject of the effect that undeformed chip thickness has on energy consumption when applied dynamically and whether its magnitude can be predicted based on knowledge of its dependence on static variations in this processing parameter.

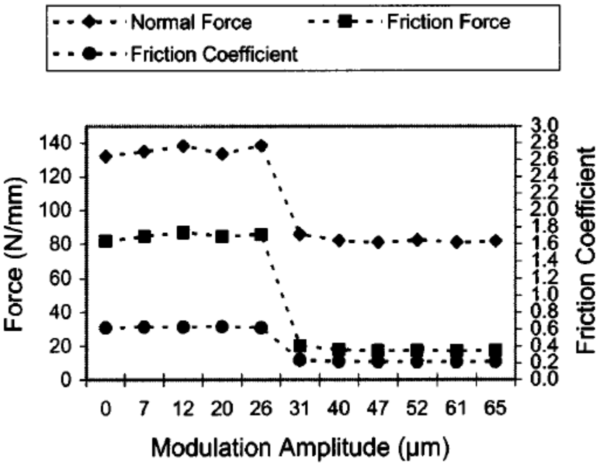


Figure 2.1: Cutting forces and friction coefficient values for a range of velocity-direction modulation amplitudes. In all cases, V_{DC}=10mm/s, h_0=0.10mm, fm=75Hz. The normal and friction forces reported here were the average plateau values [3].

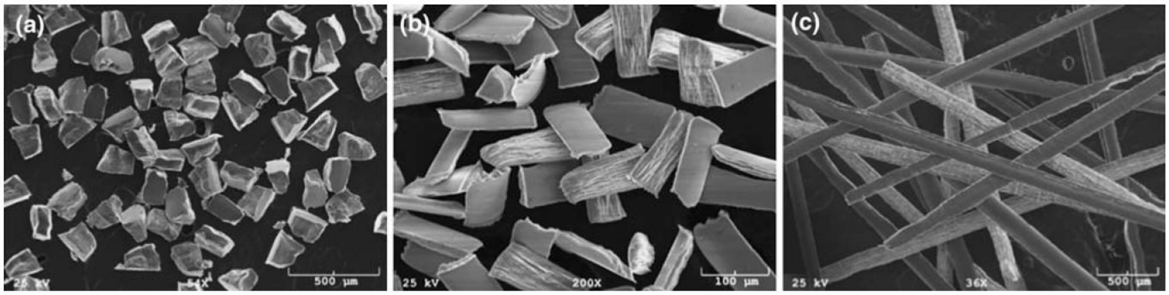


Figure 2.2: Examples of chips produced using feed-direction MAM [18].

Chapter 3: Mechanics of MAM

This chapter serves to review the underlying mechanics of MAM-based processing methods. It should be noted that many of the equations used throughout this chapter have also been reported in Ref. [27] which was frequently referred to during the course of the present study.

3.1 Velocity-direction MAM:

Velocity-direction MAM is a special case of MAM in which a time-varying velocity in the cutting direction is superimposed on the cutting speed (see Fig. 3.1). Throughout this study, the term displacement is used to refer to the position of the tool with respect to the workpiece. With these conditions established, the effective displacement (D_{eff}) and velocity (V_{eff}) can be written in terms of time (t), the amplitude of modulation (K), the modulation frequency (f_m), and the static component of the effective cutting velocity (V_{DC}) as

$$D_{eff} = K\sin(2\pi f_m t) + V_{DC}t \qquad \text{[Eq. 3.1]}$$

$$V_{eff} = 2\pi f_m K\cos(2\pi f_m t) + V_{DC}\ . \qquad \text{[Eq. 3.2]}$$

It should be noted that the absence of any phase shift in the sine and cosine terms infers that $\sin(2\pi f_m t)=0$ at time t=0 (e.g. zero displacement at t=0). Also, since the value of $\sin(2\pi f_m t)$ is arbitrary at the edge of the workpiece, the location represented by zero displacement may be offset from this edge. Finally, since tool/workpiece displacement and velocity are defined relative to one another, an experimental setup may allow for the tool and/or the workpiece to be in motion relative to some fixed object.

The effective velocity will travel between maximum and minimum values of $V_{DC}+2\pi f_m K$ and $V_{DC}-2\pi f_m K$, respectively. Depending on the relationship between V_{DC}, f_m, and K, one of three situations will occur. In each of them, the average cutting velocity will be equal to V_{DC}; thus, the displacement trace will always be represented by a sinusoidal variation superimposed on a line of constant slope and the velocity trace by a sinusoidal variation offset from the x-axis. The first case has been named “forward advancement” since the tool not only remains in contact with the workpiece but also has a positive velocity (e.g. advances) throughout the entire length of cut (see Fig. 3.2a). Forward advancement occurs when $V_{DC}>2\pi f_m K$. The second is known as the “critical case” since the tool and workpiece remain in contact but periodically have zero instantaneous relative velocity (see Fig. 3.2b). The critical case occurs when $V_{DC}=2\pi f_m K$. The third condition is referred to as “separation” since the tool and workpiece periodically separate from one another throughout the cut (see Fig. 3.2c). Separation occurs when V_{DC}<2πf_mK. When this condition is met, each complete modulation period will have three notable times: t’, t’’, and t’’’. Conceptually, t’ denotes the time(s) at which separation begins, t’’ the time(s) at which the distance of separation is a maximum, and t’’’ the time(s) at which the tool and workpiece regain contact. Graphically, t’ is the time at which the displacement-time trace sees a local maximum, t’’ the time of a local minimum, and t’’’ the time at which the displacement is next equal to its value at time t’. See Fig. 3.3 for a pictorial description of these critical times.

The time(s) between which the tool moves away from the workpiece (t’ ≤ t ≤ t’’) can be solved analytically by setting V_{eff} in Eq 3.2 equal to zero, resulting in

$$t^{\prime,\prime\prime} = \frac{1}{2\pi f_m} cos^{-1}\left(\frac{-V_{DC}}{2\pi f_m K}\right) . \qquad \text{[Eq. 3.3]}$$

Recalling from geometry that the function cos(X) over the range 0 to 2π graphically looks like Figure 3.4, then t’ and t’’ are related by

$$f_m t^{\prime\prime} = 1 - f_m t^{\prime} . \qquad \text{[Eq. 3.4]}$$

Solving for the difference between the local maximum and subsequent local minimum yields the equation for maximum effective distance of separation, given by

$$\Delta D_{eff,max} = 2K\left|\sqrt{1-\left(\frac{V_{DC}}{2\pi f_m K}\right)^2}\right| + \frac{V_{DC}}{\pi f_m}\left[cos^{-1}\left(\frac{-V_{DC}}{2\pi f_m K}\right) - \pi\right] . \qquad \text{[Eq. 3.5]}$$

Note that Eq. 3.3 and Eq. 3.5 will only yield real values if separation occurs.

3.2 Feed-direction MAM:

Feed-direction MAM is a special case of MAM in which a time varying displacement in the feed direction is superimposed parallel to the undeformed chip thickness (see Fig. 3.5). In the case of turning, this induced modulation has a constant phase shift (φ) between passes of

$$\varphi = 2\pi\left[\frac{f_m}{f_w} - INT\left(\frac{f_m}{f_w}\right)\right] \qquad \text{[Eq. 3.6]}$$

where fw is the angular speed of the workpiece (e.g. spindle speed) and INT() denotes the integer part of the value. The wavelength (λ) is expressed as

$$\lambda = \pi \mathrm{d}\frac{f_w}{f_m} \qquad \text{[Eq 3.7]}$$

where d is the average diameter of the workpiece. The overall displacement (Z) of the tool in its n^th pass can be represented spatially as

$$Z_n(x) = (n-1)s_0 + \left(\frac{x}{\pi d}\right)s_0 + A\sin\left(\frac{2}{d}\frac{f_m}{f_w}x + (n-1)\varphi + \varphi_0\right) \qquad \text{[Eq. 3.8]}$$

where x represents the instantaneous location of the tool along the circumference of the workpiece (0≤x≤2πd) [27]. The instantaneous undeformed thickness (h_0) can be found by taking the difference between the displacements of the n^th and n-1^th passes

$$h_0 = Z_n - Z_{n-1}\ . \qquad \text{[Eq. 3.9]}$$

Under certain conditions it is possible for h_0 to be analytically less than or equal to zero which is theoretically synonymous with tool/workpiece separation and the formation of a discrete chip. It is important to note that chip thickness, and thus the conditions for separation, depends not only on the amplitude of modulation but also on the phase shift between passes. The minimum amplitude required to achieve separation as a function of the phase shift can be expressed as

$$K_{min} = \frac{s_0}{2\sin\left(\frac{\varphi}{2}\right)} \qquad \text{[Eq 3.10]}$$

It is evident from Fig. 3.6 that separation can be achieved with minimum modulation amplitude when φ=π, which has been named the “optimum modulation condition” [17,19].

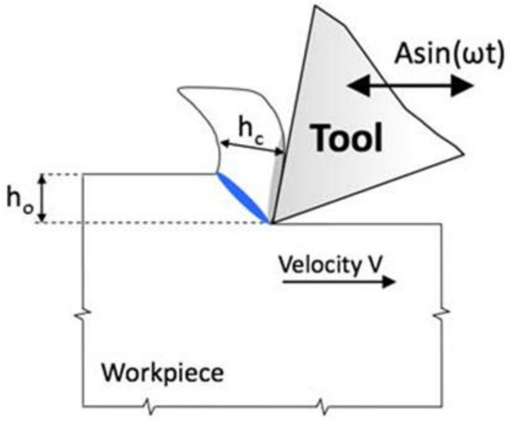


Figure 3.1: Orthogonal cutting geometry for velocity-direction MAM [19]. Note that the amplitude A is referred to as K in the present study.

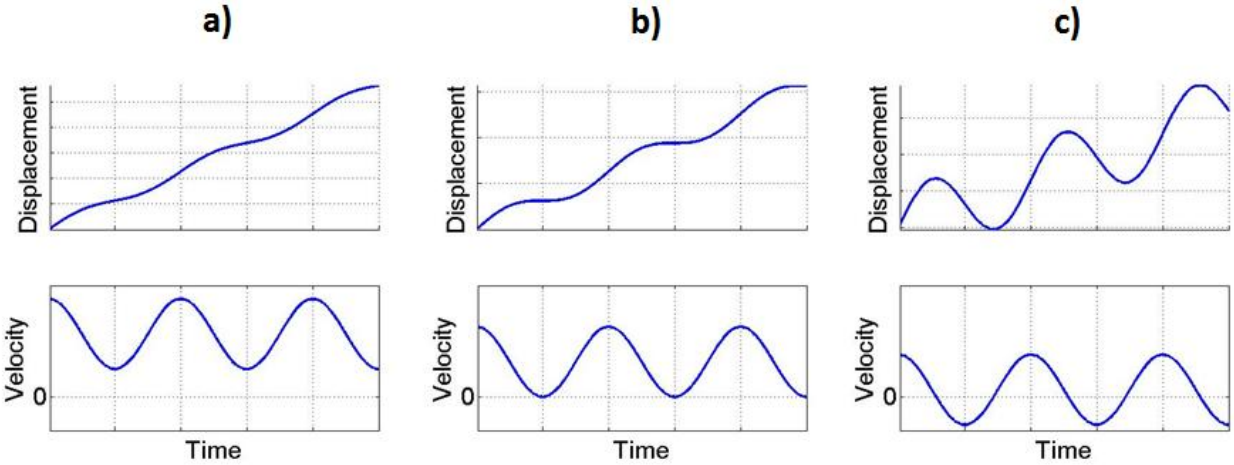


Figure 3.2: Tool displacement and velocity vs time for cases of: (a) forward advancement, (b) critical condition, and (c) separation.

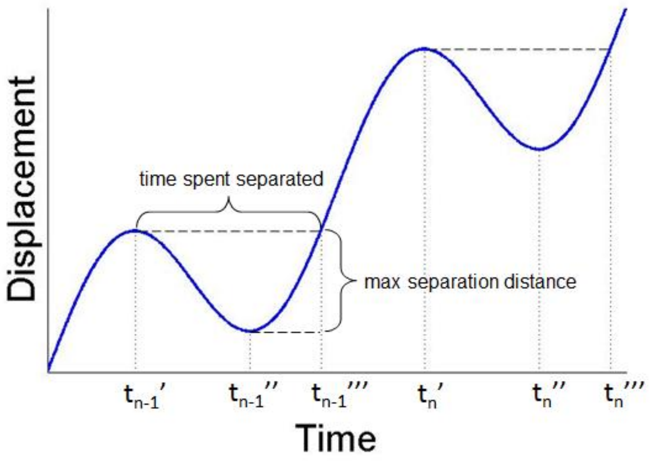


Figure 3.3: Pictorial representation of critical times as defined using the displacement-time trace.

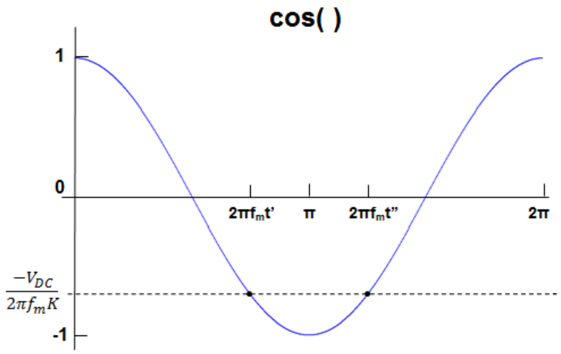


Figure 3.4: Graphical example of Eq. 3.3.

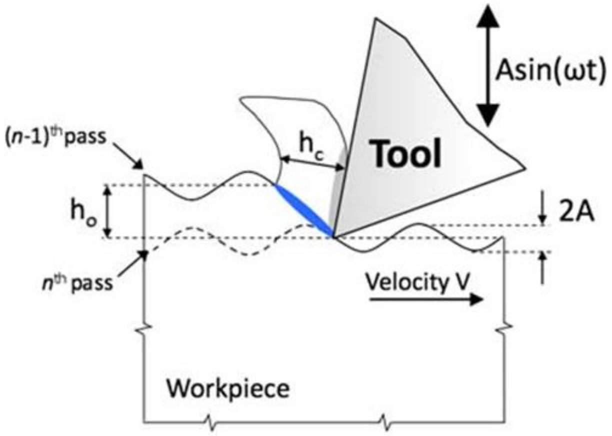


Figure 3.5: Orthogonal cutting geometry for feed-direction MAM [19]. Note that the amplitude A is referred to as K in the present study.

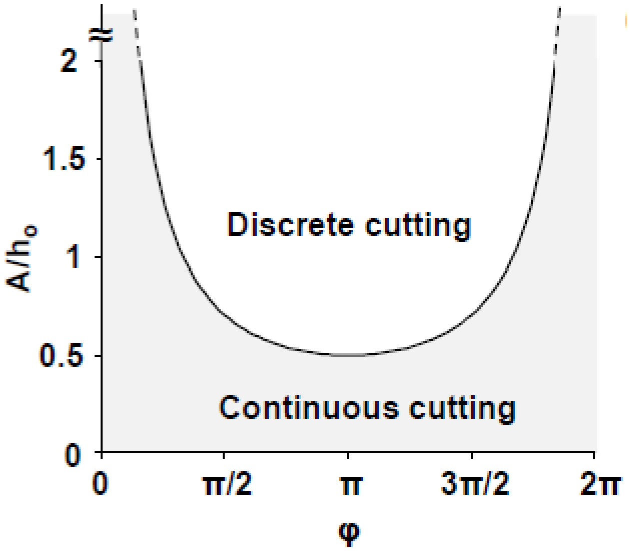


Figure 3.6: Classification of cutting type as a function of modulation amplitude and phase shift between passes [7]. Note that the amplitude parameter A is referred to as K in the present study.

Chapter 4: Experimental Methods

This purpose of this section of the thesis is to describe the experimental configurations, materials and data acquisition methods used in executing the experimental work of the present study. As the thesis involved the exploration of power- and energy-related effects for two types of parameter modulation methods, namely velocity-direction modulation and feed-direction modulation, this chapter is logically divided along those lines.

4.1 Velocity-direction MAM

A. Experimental setup To study the effects that cutting velocity has on specific energy consumption for both conventional cutting and velocity-direction MAM, a custom computer-controlled planer was constructed (Fig. 4.1). The device couples a linear motion with a sinusoidally varying modulated motion and imparts them on a workpiece. The linear motion is provided by a linear stage (Nippon Bearing Corp., Model BG5520B) which is driven by a closed-loop, PID controlled servo motor (Parker USA, Model MPP1002D3E-NPSN). Positioning data is provided to a motion controller (Parker USA, Model 6K2) by an in-built optical rotary encoder with effective resolution of 8000 counts/rev (post-quadrature). These digital signals were also monitored externally during each experiment to measure the constant, or DC, component of workpiece velocity. The motion controller enables the servo drive (Parker USA, Model Aries AR-20AE) to regulate the power supplied to the servo motor. The rotational motion provided by the motor is converted to linear motion via a rotary coupling to the ball-screw of the linear stage. The primary shaft of the ball-screw has a pitch of 20 mm, resulting in an overall theoretical linear positioning accuracy of ±2.5 um when considering the optical encoder capability.

The sinusoidal, or AC, component of the workpiece velocity was provided by a piezo-ceramic actuator (APC International, Model Pst 150/14/100 VS20). One end of the actuator was fixed to the ball-screw driven platform while the other was attached to a work-fixturing platform that enabled motion along a single degree of freedom in the cutting direction. The work-fixturing platform was a well-greased linear slide (Nippon Bearing Corp., Model SGW35) whose motion axis was carefully aligned with the larger linear stage using a dial indicator. The work-fixturing platform facilitated holding of a workpiece using set screws. The displacement of this platform relative to the linear stage (e.g. the purely AC component of the motion) was monitored using a capacitance probe (Capacitec, Model HPC-40) which was affixed to the ball-screw driven platform. The oscillatory motion was generated by charging the piezo-ceramic actuator with a sinusoidal voltage which originated from a waveform generator (Agilent, Model 33220A) and was amplified using a power supply (Kepco, Model BOP 100-4M). Using this system, voltages ranging from 0-100 V can be provided to the actuator.

The cutting tool was produced from a 12.7 mm x 25.4 mm x 177.8 mm high speed steel blank and was mounted on a cast-iron tool holder. The tool was machined using wire electrical discharge machining and had an effective rake angle of 30° and a relief angle of ~5°. The rake and flank surfaces of the tool were finish ground to produce a sharp cutting edge (Fig. 4.2). The tool holder was mounted on a three-axis force transducer (Kistler, Model 9257A) with the orientation of two of these axes, F_X and F_Y, aligned along and perpendicular to the cutting direction, respectively. The transducer was mounted on a ball-screw driven dovetail slide (Setco USA, Model M2PLWY.8) with an in-built linear encoder (Heidenhain) attached to a digital readout device. The rough positioning of the tool was set by adjusting the height using the hand crank on the dovetail slide. The final value of undeformed chip thickness was verified using a second capacitance probe (Capacitec, Model HPC-150). The undeformed chip thickness in these experiments was 20 um with an uncertainty of 2.5%. It should be noted that the capacitance probes require a common reference ground. This was provided by the steel table.

Arguably the most difficult challenge in implementing a precision machining tool is minimizing system compliance due to the force generated by the tool/work interaction. The fundamental principles of beam theory state that the deflection of an end loaded beam can be expressed as $\delta=(PL^3)/(3EI)$ where δ is the linear deflection, P the cutting force, L the length of the beam, E the Young's modulus, and I the area moment of inertia. Several design iterations of the tool and its holder were necessary to reduce compliance. The final experimental platform was tested for compliance at various locations over a range of cutting forces. A capacitance probe was placed normal to the respective surfaces indicated by Fig. 4.3 and the values of force and the corresponding local compliances are recorded in Table 4.1.

To obtain the desired cutting velocities under the load ranges investigated, the motor required tuning. Two sets of PID gains were used: one for conventional cutting and the other for MAM. Although it is common practice to tune such gains as tightly as possible to yield the fastest system response, it was found that high gains effected unsteady motions. This was validated by comparing the encoder's internally measured digital motion profile with an externally measured position provided by the capacitance probe. The capacitance probe was positioned such that it measured normal to the direction of travel. A series of motions at a range of velocities were tested and the two displacement profiles were compared. To simulate the cutting loads expected in the study, the motion profiles were measured during actual cutting tests. The results of these tests can be seen in Figs. 4.4-4.5. The values for the high gain settings on the motion controller were (P, D, I) = (25.0, 4.4, 0), while the low gain settings were (P, D, I)=(8.3, 3.6, 0). It was empirically observed that more linear profiles were possible using the low-gain settings; thus, all conventional machining experiments were performed with these gains. For modulation-assisted machining, a time-varying load caused counter-rotation of the motor resulting in departure from its linear motion. In these cases, high-gain settings were used as they provided better response for these time-varying loads. Figures 4.6-4.7 provide evidence that the magnitude of this counter-rotation was lessened by the use of high-gains. The sinusoidal motion of the platform itself was also validated, as is clear in Fig. 4.8.

In the context of the planer machine, the term displacement is used to refer to the location of an arbitrary point on the work holding platform relative to the cutting tool. An increasing displacement indicates that the workpiece is moving in the cutting direction. This demonstrates the concept that it is not the value of displacement that is of primary importance here, but rather the nature of how the displacement trace varies over time. The reason this parameter is displayed in the results and not merely bypassed en route to obtaining cutting velocity has to do with its ability to relate conceptually with tool-work separation (refer to Fig. 3.3). Recall that the effective displacement trace represents a synthesis of temporally linear and nonlinear motion. The linear component is due to the (approximately) constant velocity (V_{DC}) imposed on the stage by the motor and the nonlinear component is due to the sinusoidal velocity (V_{AC}) imposed by the piezoelectric actuator. The effective velocity (Veff) is simply the superimposition of V_{AC} on V_{DC}. For cases of conventional cutting, V_{AC}=0 and thus Veff=V_{DC}. It should be noted that the velocity terms seen in the figures are best-fit approximations of the data and are therefore noise-free. Also, recall that V_{DC} has a time varying component built into it due to the nature of the motor to counter-rotate when under sufficient load. This is factored into the best-fit models.

B. Data acquisition Two channels on the dynamometer were used to sense the cutting and tangential forces according to the orientations seen in Fig. 4.9. The signal from these channels was amplified using charge amplifiers (Kistler, Model 5004). The charge from the capacitance probes was also amplified (Capacitec, Model 4100-SL). These data were then output from their respective amplifiers and received by a data acquisition system (National Instruments, Model cDAQ-9178) that simultaneously accessed an analog input board (NI-9215) and a digital input board (NI-9411) at a sampling rate of fs=10 kHz. The differential digital signals from the rotary encoder operated at TTL voltage levels and thus required no amplification. To facilitate simultaneous acquisition of data from these two input boards, it was necessary to record the raw digital signals from the encoder and to decode the quadrature signal offline using a specially-designed Matlab script (see appendix D). This was because Labview's in-built linear encoder subroutine was observed to cause severe data lag, presumably due to the computational cost of dynamically decoding the quadrature signal. The Matlab quadrature decoding script was validated by commanding the linear stage to first move a pre-programmed distance (25mm) at a constant speed (10mm/s) and then to move the same distance in reverse at half the original speed. Results from this test can be seen in Fig. 4.10. The synchronization of the two input boards was validated by measuring equivalently-sourced electrical signals from the encoder using both the analog and digital modules (Fig. 4.11). As is evident in the figure, these modules are able to simultaneously measure the rise and fall of the encoding signal with a high degree of accuracy.

C. Experiments The materials tested included AA6061-T6 and oxygen-free electronic (OFE) copper (Cu101). Workpieces were constructed from 3.175 mm thick flat bar stock (McMaster Carr) and were machined to a size of approximately 20 mm x 30 mm. The workpiece thickness was reduced to 1.0 mm for a step height of 1.5 mm. The initial thickness of 3.175 mm was maintained throughout the remainder of the workpiece to provide structural integrity (Fig. 4.12). In all cases, the undeformed chip thickness (h_0) was 20 um, resulting in a cutting force of Fp≈15-20 N for both materials. The system compliance values given in Table 4.1 are representative of those expected under these force levels. Note that the cutting width is at least 10 times greater than the cutting depth, which is generally accepted as facilitating plane strain deformation [1].

Five cutting speeds (V_{DC}) were used to characterize load response as a function of velocity: 0.05, 0.25, 0.50, 0.75, and 1.00 mm/s. Three trials were performed at each of these speeds (except V_{DC}=0.50 which had 8 trials) and the overall average cutting power for each speed was found. In order to compare conventional cutting with velocity-direction MAM, five modulated trials were performed at V_{DC}=0.50 mm/s, fm=10 Hz, and K=19 V for aluminum and 16 V for copper. Note that the amplitude of modulation varies between materials since it was designed so as to effect approximately 75% of the critical MAM condition. Table 4.2 provides a comprehensive list of the cutting conditions mentioned here. The order in which these cuts were performed was randomized and reference-establishing cuts were made between each trial. These reference cuts were performed at V_{DC}=1 mm/s and h_0=20 um. For the conventional machining tests, the piezo-actuator was provided with a DC offset voltage such that the only difference between the MAM tests was the lack of an AC motion. This DC offset merely placed the actuator in an extended condition to ensure that the actuator stiffness would be similar between the conventional and MAM conditions used in this study.

4.2 Feed-Direction MAM

All experiments for feed-direction MAM were performed in Ref. [27] and a new analysis of the same data sets is provided here using the analysis framework described in Chapter 5.

A. Experimental setup The MAM experiments were performed using a computer numerical control (CNC) lathe (Miyano, Model BNC42) outfitted with a custom-built piezo-actuation device capable of achieving a sinusoidal motion. A boring tool (0deg rake, 5deg relief) was mounted to this actuator and oriented such that the configuration also approximated orthogonal cutting. While the use of a piezo-ceramic actuator to cause the sinusoidal motion of a tool is similar to that in the preceding section, the key difference in this series of experiments is that the sinusoidal motion is effected in a direction transverse to the cutting velocity and in the direction of tool feed. A three-axis load washer (Kistler, Model 9027A) was mounted between the boring tool holder and the piezo device such that it was capable of sensing the cutting loads. The spindle speed and feed rate (e.g., undeformed chip thickness) were controlled by entering the respective parameters into the controller of the CNC lathe. Finally, the oscillatory motion of the tool was controlled in a similar manner as described above for velocity-direction modulation, that is, by means of a similar waveform generator and voltage amplifier.

The term displacement is used in the context of the CNC lathe to refer to the location of the boring bar’s cutting edge relative to that of the workpiece. Increasing values of displacement indicate that the turret is feeding the tool into the workpiece. Since the undeformed chip thickness, or instantaneous feed, is the difference between the displacement profiles of the current and previous pass, it is also sinusoidal in nature (with an offset equal to the feed rate). The only deviation from this shape comes in the form of a saturation limit at h_0=0, thus preventing what would be negative values of undeformed chip thickness. Note that in the case of conventional cutting, the amplitude of this sine wave is zero, thus, the instantaneous undeformed chip thickness is equivalent to the feed rate of the turret (s).

B. Experiments Turning experiments were carried out using AA6061-T6 and Ti3Al2.5V tubes which had an outer diameter of 25.4 mm. The wall thicknesses for the aluminum and titanium tubes were 1.24 mm and 1.295 mm, respectively. Each experiment was performed with a spindle speed of 1200 rpm (giving cutting speeds of 1.52 and 1.51 m/s for aluminum and titanium, respectively). Static feed rates varied between 0.005 mm/rev and 0.050 mm/rev, which should be noted as being faciliatory to plane strain conditions [1]. A total feed length of 0.5 mm was used for the 0.005 mm/rev case and 2.0 mm was used for all other cases. Modulation frequencies of 10, 50, 100, and 110 Hz were each used, effecting fm/fw ratios of 0.5, 2.5, 5.0, and 5.5, respectively. Data was sampled at a rate of fs=5 kHz. The piezo-ceramic actuator received a sinusoidal voltage signal of amplitude 60V in all cases. A list of the conditions tested using the lathe is given in Table 4.3. Note that only one trial was performed for each condition.

Note: see Table 4.4 for a list of various mechanical and thermal properties for the workpiece materials used in the present study.

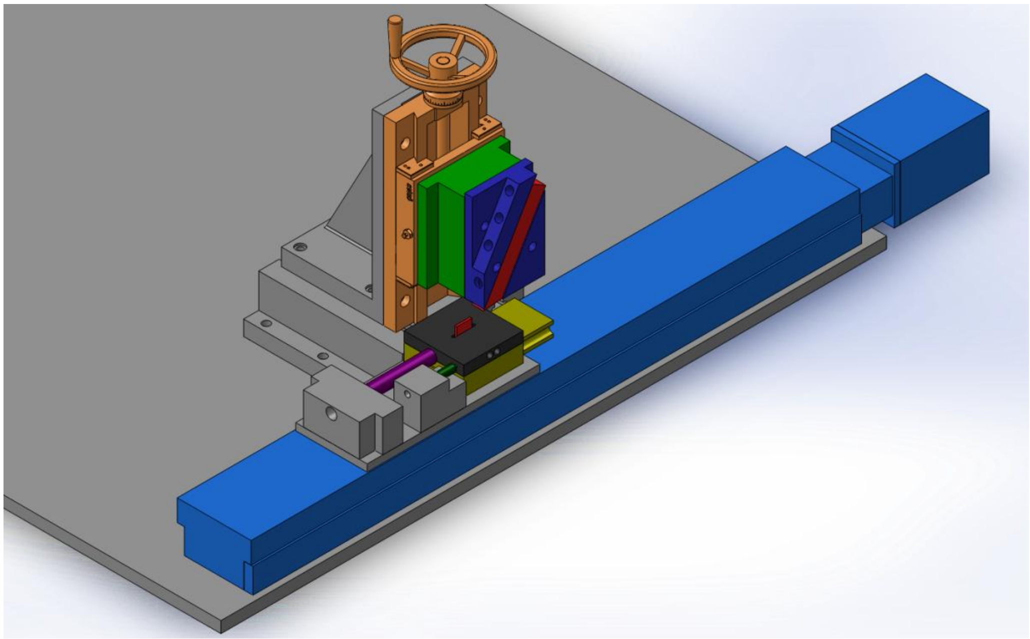


Figure 4.1: Solid model of computer-controlled planer. Components include linear stage (light blue), piezo-actuator (violet), capacitance probe (dark green), linear slide (yellow), work fixturing platform (black), workpiece and tool (red), tool holder (dark blue), force transducer (light green), vertical stage (tan).

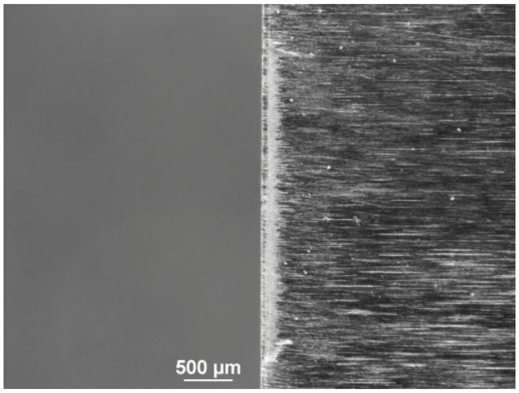


Figure 4.2: Profile of cutting edge prior to experimental trials.

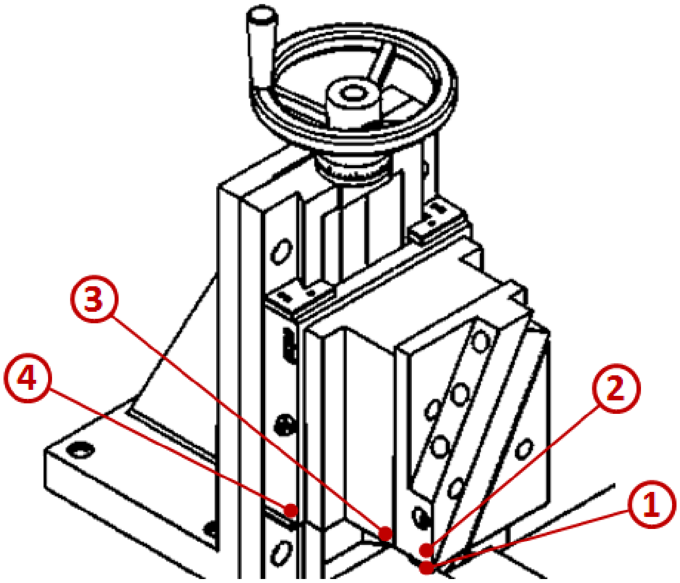


Figure 4.3: Locations along tool assembly at which compliance was checked as reported in Table 4.1.

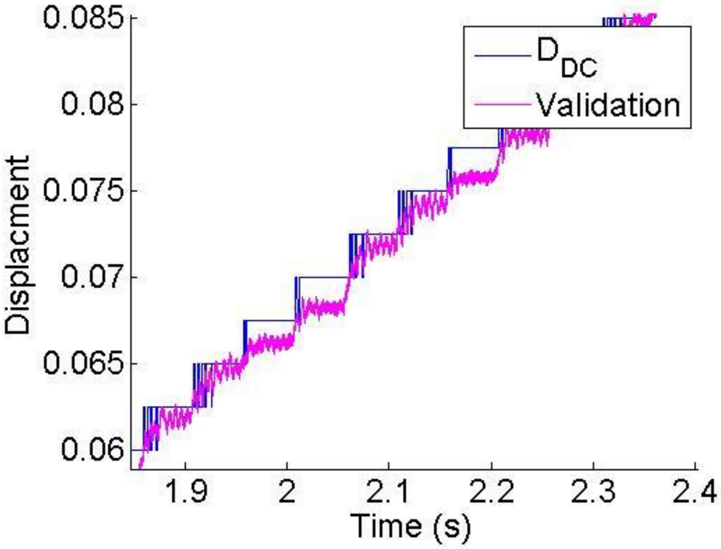


Figure 4.4: Tested “linear” motion of stage for V_{DC}=0.05 mm/s with high gain settings (P, D, I) = (25.0, 4.4, 0).

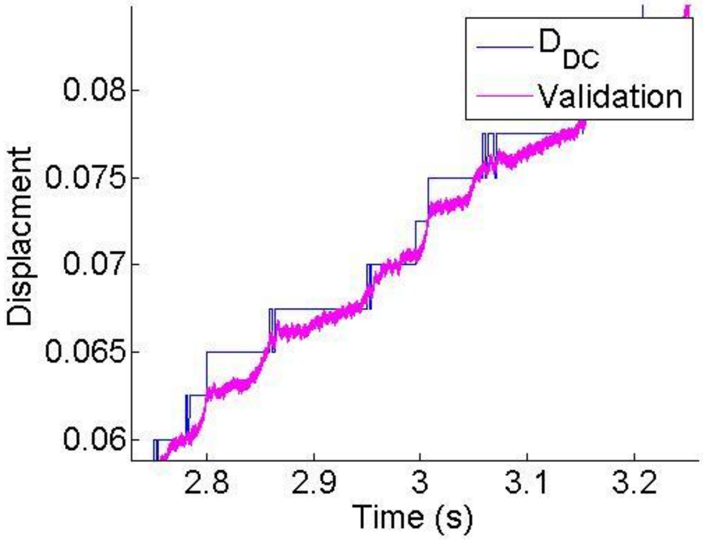


Figure 4.5: Tested “linear” motion of stage for V_{DC}=0.05 mm/s with low gain settings (P, D, I) = (8.3, 3.6, 0).

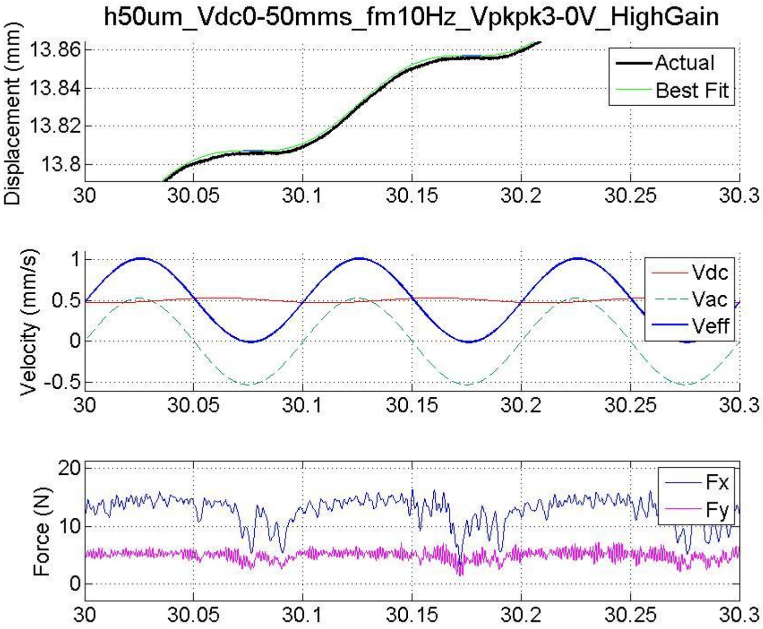


Figure 4.6: Analysis of modulated cut using the high gain settings. Note the constancy of Vdc.

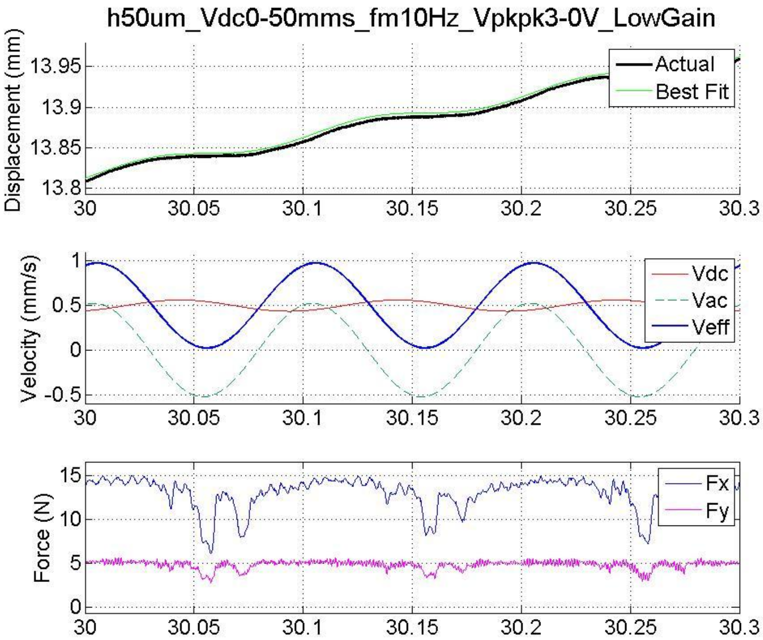


Figure 4.7: Analysis of modulated cut using the low gain setting. Note the lack of constancy in Vdc.

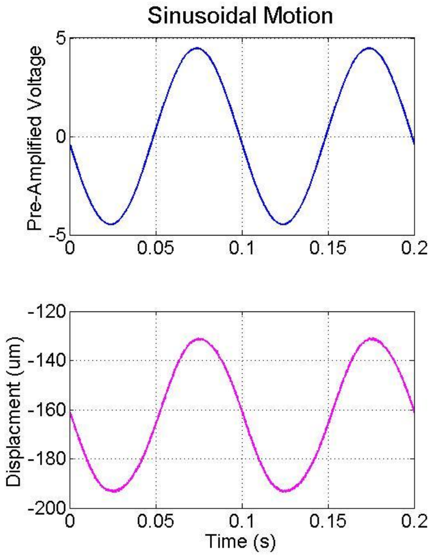


Figure 4.8: Pre-amplified 10 Hz sinusoidal voltage signal and resulting motion of work holder platform. Post amplified signal sent to piezo actuator was 90 V peak to peak.

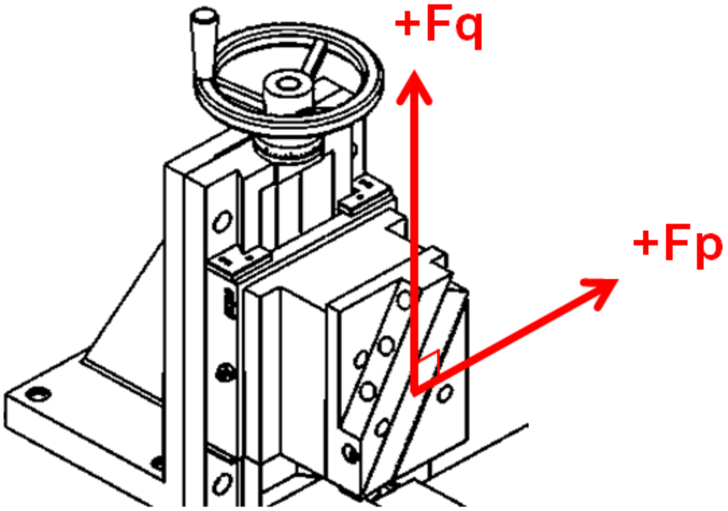


Figure 4.9: Orientation of cutting direction force, Fp, and its transverse counterpart, Fq.

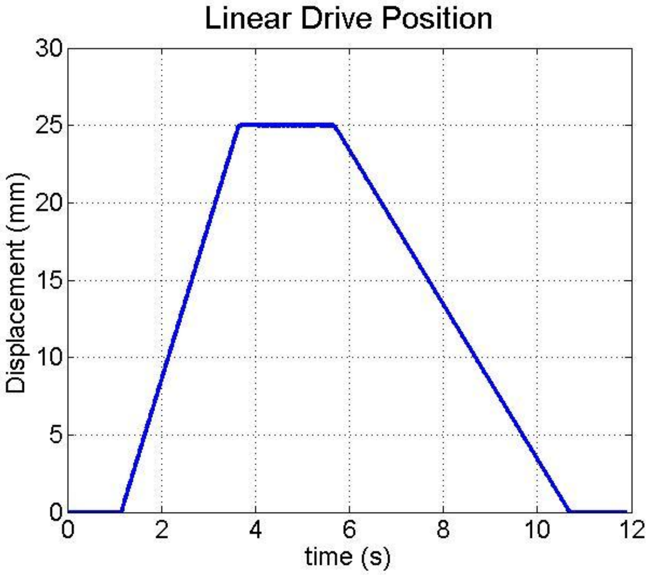


Figure 4.10: Decoded motion profile of linear stage resulting from a commanded motion of distance D=25mm at velocity V=10mm/s followed by a motion of distance D=-25mm at velocity V=5mm/s.

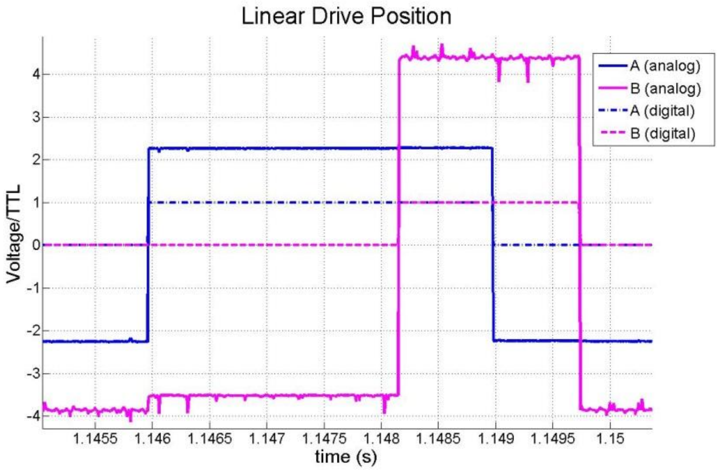


Figure 4.11: Analog and digital voltage measurements of identical signals from encoder. The two tracks, A and B, are a result of the use of quadrature encoding.

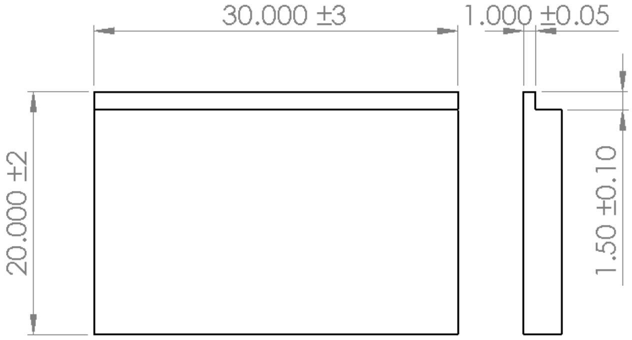
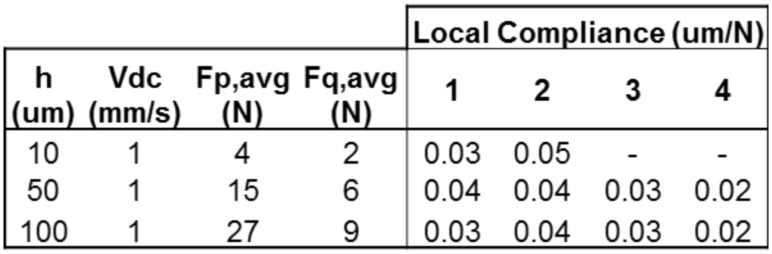


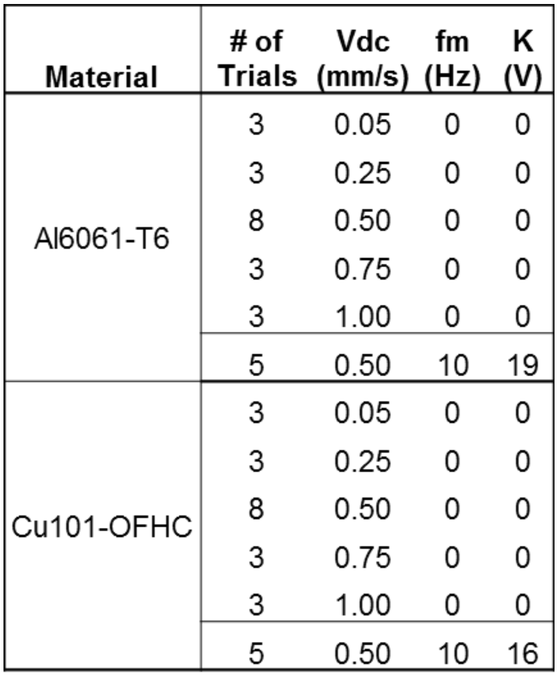
Figure 4.12: Workpiece drawing: note that the planer experimental cutting trials removed material from the 1mmx1.5mm cross-section.

Table 4.1: Average forces and corresponding local compliance values (normalized by Fp,avg) of planer setup for various conventional cutting conditions. Note that Fp and Fq are in the direction of and tangent to the cutting velocity, respectively.



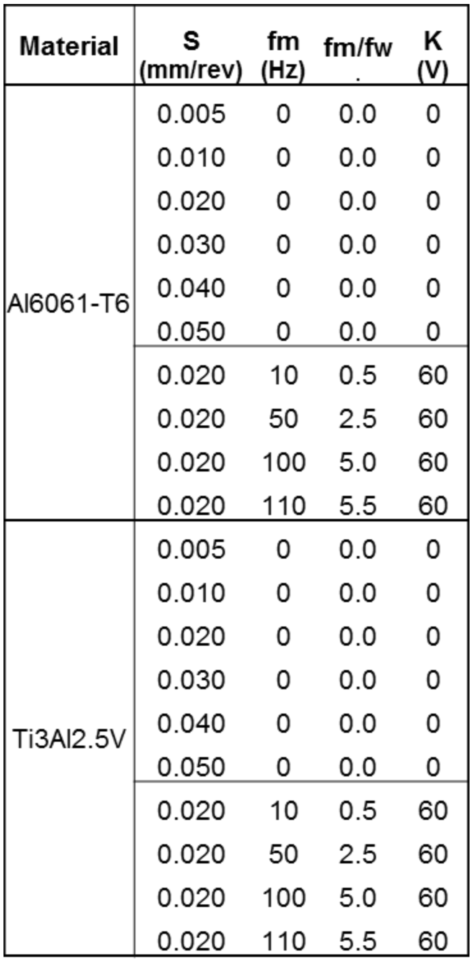
				Local Compliance (um/N)			
h (um)	Vdc (mm/s)	Fp,avg (N)	Fq,avg (N)	1	2	3	4
10	1	4	2	0.03	0.05	-	-
50	1	15	6	0.04	0.04	0.03	0.02
100	1	27	9	0.03	0.04	0.03	0.02

Table 4.2: List of workpiece materials and cutting conditions tested using CNC planer (velocity-direction MAM).



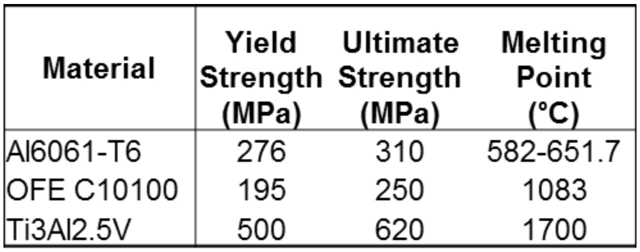
Material	# of Trials	Vdc (mm/s)	fm (Hz)	K (V)
Al6061-T6	3	0.05	0	0
	3	0.25	0	0
	8	0.50	0	0
	3	0.75	0	0
	3	1.00	0	0
	5	0.50	10	19
Cu101-OFHC	3	0.05	0	0
	3	0.25	0	0
	8	0.50	0	0
	3	0.75	0	0
	3	1.00	0	0
	5	0.50	10	16

Table 4.3: List of workpiece materials and cutting conditions tested using CNC lathe (feed-direction MAM). Note that a spindle speed of 1200rpm was used throughout.



Material	S (mm/rev)	fm (Hz)	fm/fw .	K (V)
Al6061-T6	0.005	0	0.0	0
	0.010	0	0.0	0
	0.020	0	0.0	0
	0.030	0	0.0	0
	0.040	0	0.0	0
	0.050	0	0.0	0
	0.020	10	0.5	60
	0.020	50	2.5	60
	0.020	100	5.0	60
	0.020	110	5.5	60
Ti3Al2.5V	0.005	0	0.0	0
	0.010	0	0.0	0
	0.020	0	0.0	0
	0.030	0	0.0	0
	0.040	0	0.0	0
	0.050	0	0.0	0
	0.020	10	0.5	60
	0.020	50	2.5	60
	0.020	100	5.0	60
	0.020	110	5.5	60

Table 4.4: Various mechanical and thermal property values for the workpiece materials used in this study.



Material	Yield Strength (MPa)	Ultimate Strength (MPa)	Melting Point (°C)
Al6061-T6	276	310	582-651.7
OFE C10100	195	250	1083
Ti3Al2.5V	500	620	1700

Chapter 5: Analysis Methodology

A fundamental characteristic of modulation-assisted processing methods is that controllable process parameters are modulated over a range of values so as to affect desirable changes in baseline performance. In the present study, controllable machining parameters for modulation-assisted machining include cutting speed and undeformed chip thickness. In conventional machining configurations, performance measures such as instantaneous power dissipation and energy consumption are usually sensitive to these inputs within a parameter space whose breadth is material-dependent. In this regard, it is unsurprising that many have reported that the use of modulation has a marked effect on performance for a number of machining configurations [3,7,9,16-24,26-30]. Explanations of these effects usually cite fundamental differences between thermo-mechanical response in modulation-assisted versus conventional processing methods; these include but are not limited to: lower yield stresses due to enhanced dislocation generation [31], reduced friction due to enhanced contact lubrication [3] and incipient straining associated with smaller removal volumes [32]. While each is plausible for explaining the observed differences, it often is not recognized that the time-varying nature of the machining parameters may also affect overall performance. In this regard, insufficient consideration has been given to explicitly resolving whether parameter modulation simply modifies the temporal distribution of the output measures (e.g., force) or whether it fundamentally alters the nature of thermo-mechanical response underlying deformation in material removal.

The purpose of this section of the thesis is to address how the character of functional relationships between output measures (e.g., instantaneous power) and controllable machining parameters in conventional processing configurations can be used to understand the role of parameter modulation in affecting performance in modulation-assisted configurations.

5.1 Discretization of the cutting process

The cutting process in modulation-assisted machining can be modeled as a discretization of that which occurs in conventional cutting. In the limit wherein parameters controlling modulation (e.g., frequency, amplitude) tend toward zero, machining parameters (e.g., undeformed chip thickness, cutting velocity) are constant and this process approximates one of continuous material removal. In contrast, for configurations wherein parameters controlling modulation are non-zero, modulation can be used to affect significant changes to characteristics of the material removal process, such as periodic contact separation and/or oscillatory loading. It is unclear whether the presumption holds that the cutting process in these latter scenarios is simply a discretization of that which occurs in conventional cutting or whether fundamental differences exist for the unit events in modulation-assisted machining. Instantaneous power consumption and overall energy expenditure are output measures that together can be used to probe this question. If the underlying nature of both of these output measures is fundamentally different with the application of modulation, then the notion of discretization of the conventional machining process is suspect.

Instantaneous power consumption is useful in evaluating maximum power requirements and energy expenditure during cutting. The expended energy is simply obtained by integration of the power distribution over time. In conventional machining, cutting power (and force) saturates rapidly after the start of chip formation to a plateau whose magnitude is related to a host of controllable input parameters including speed, undeformed chip thickness, rake angle, etc. This nominal cutting power is generally maintained until chip formation ceases. In the case of milling, the power consumption may vary dynamically over time since each tooth of the tool will typically engage and disengage the workpiece once per spindle revolution. However, this can still be viewed as a static parameter process, since the processing parameters – most notably cutting speed and undeformed chip thickness – are held relatively constant throughout the duration of each cut.

In contrast, the power consumption profile in modulation-assisted machining is time-varying since the processing parameters are themselves intrinsically dynamic. In this view, the nature of load response can be expected to be similar in modulation-assisted machining as in conventional machining, where instantaneous force dissipation and power consumption are both dictated by the instantaneous cutting parameters. In this regard, it is important to establish the manner in which cutting power in conventional machining changes as a function of input parameters, particularly those modulated in the current study: undeformed chip thickness and cutting velocity.

5.2 Effects of parameter modulation on power

The effects of parameter modulation can be generalized to represent behavior for any modulation configuration. Consider an arbitrary machining parameter, X, whose value can be modulated with respect to time in modulation-assisted machining. For example, in feed modulation, this parameter would be the undeformed chip thickness. For a modulation configuration that directly affects X, the instantaneous value of X is given by X=Ksin(2πf_mt)+X_0 where K is modulation amplitude (in units of X), f_m is modulation frequency, t is time and X_0 is the initial value of X. Considering the sinusoidal nature of this relationship, the value of X can vary over a range [-K+X_0, K+X_0]. Consider also that instantaneous power consumption has historically been taken as the cutting force multiplied by the instantaneous cutting speed [1]. If a dependent relationship exists between parameter X and instantaneous power (denoted as P), it follows that P will also be time-varying in the case of modulation and will vary between some maximum and some minimum value. A straightforward method to determine this instantaneous power is to correlate it with load response at an equivalent value of X in conventional machining. This dependent relationship, P(X), can be established empirically by evaluating the instantaneous value of P for a range of input X's in conventional machining configurations. For example, if X represents undeformed chip thickness, the P(X) relationship can be evaluated by measuring machining power for a range of undeformed chip thicknesses in conventional machining. Such a method inherently assumes that the underlying thermo-mechanical response is the same in modulation-assisted machining as it is in conventional machining. While the validity of this assumption has yet been established, it can be tested through comparisons made of predicted power dissipation and actual power dissipation.

5.3 Effects of parameter modulation on energy

If the validity of the above approach is established, knowledge of the P(X) relationship in conventional machining enables first-pass prediction of instantaneous power consumption in modulated machining conditions wherein the value of X is controllable. Furthermore, the total energy expended during machining can be found according to $\int P(t)dt$. However, direct comparisons between energy expended in conventional machining and that consumed in modulation-assisted machining are non-trivial. In conventional machining, the power consumption during a cut is relatively time-invariant (represented as P_0) and the machining parameter value is constant (X_0). This can be compared with modulation configurations, where power dissipation is given by P(X) where X is a function of time. Thus, one-to-one comparison with P_0 requires determination of a weighted power value, P_{eff}, for modulation-assisted machining. In this case, there exists an effective modulation parameter X_{eff} such that P_{eff}=P(X_{eff}). It is not difficult to show that the nature of P(X) ultimately influences the values of P_{eff} and X_{eff}. Consider first the expected value of the modulated parameter, E[X]~X_0, for the probability density function (Fig. 5.2) of the driving sinusoidal function. The power dissipated at this parameter value is equivalent to that dissipated in conventional machining such that

$P(E[X])=P(X_0)=P_0$. In contrast, P_{eff} is given by the expected value of the power function, $E[P(X)]=P_{eff}$. The value of P_{eff} relative to P_0 can be shown to be dependent on the curvature of P(X) through Jensen's inequality. Jensen's inequality states that for a random variable 'y' that takes values in an interval over which a given function f is convex, then $E(f(y)) \geq f(E(y))$ where E(y) is the geometric mean of y over its range. Further, the inequality reverses direction when the function f is concave and the two sides are equivalent when the function f is linear [33]. This can be graphically visualized in Fig. 5.3. With regard to power dissipation in modulation-assisted machining, this indicates that: (i) $P_{eff} \geq P_0$ if the P(X) curve is convex, (ii) $P_{eff} \leq P_0$ if the P(X) curve is concave and (iii) $P_{eff}=P_0$ if the P(X) curve is linear.

From the above analysis, one may be able to determine *a priori* the effects that parameter modulation has on energy consumption through an understanding of response under a range of conventional, or static, machining conditions. This requires knowledge as to whether the corresponding static P(X) relationship is convex, concave, or linear over the range of the modulated parameter space. For configurations wherein cutting speed or undeformed chip thickness is modulated, P(X) often is linearly related to the modulated parameter X over a broad parameter space [10,34]. However, several studies have pointed to non-linear behavior at extreme conditions. Specifically, the P(X) curve can exhibit convex or concave nature at low values of cutting speed and/or undeformed chip thickness, respectively [10,11].

5.4 Empirical characterization of load response and power dissipated

While P_{eff} will ultimately be useful in exploiting the conventional machining and MAM energy inequality, the validity of using a P(X) relationship, calibrated on a range of conventional machining data, to predict instantaneous power dissipation in modulation-assisted machining must first be established. This, as described above, can come from a direct comparison of predictions using P(X) against empirically obtained measurements in MAM. To establish the P(X) relationship, the instantaneous value of P can be measured for various values of X in conventional machining and a least-squares fit of the data can be made to approximate P(X). In practice, power oft is not measured directly in machining experiments; it usually is inferred from measurements of force response F made through dynamometry. Recall that the cutting power in machining is historically given as the product of the cutting force Fp and the cutting velocity V. It is simple to posit that the force dissipated as a function of X, e.g., Fp(X), can be used to establish P(X). An understanding of an appropriate general form to fit Fp(X), where X is cutting velocity or undeformed chip thickness, can come from literature.

A. Velocity-direction MAM Research performed by Maan and co-workers indicates that, for steels cut at low speeds, there exists an approximately logarithmic relationship between specific energy and cutting speed [11]. This can be seen in Fig. 5.4, where the specific energy is plotted against the log-scaled cutting speed. This results in a linearized relationship between the variables which is most readily fit according to e(X) = A'*ln(X) + B' where e is specific energy in J/mm^3, X is cutting velocity in mm/s, and A' and B' are fitting parameters. From this, the cutting force (Fp) can be easily back-formulated into the equation $\mathrm{Fp} = \mathrm{A}\ln[\mathrm{X}] + \mathrm{B}$ where the fitting constants A and B are equivalent to h_0w_0A' and h_0w_0B'. The parameters h_0 and w_0 represent undeformed chip thickness and workpiece width, respectively. The parameters of A and B can be found through the least-squares technique. Specifically, if M*p=r where M is a matrix of size (n x q) containing elements of the fitting function, p is a vector of size (q x 1) containing the fitting parameters and r is a vector of size (n x 1) containing the data to be fit, then the fitting parameters can be solved according to $p = (M^TM)^{-1}*M^T*r$. After the fitting parameters are determined, the power dissipation relationship is simply written as $P = A(X * 10^{-3})\ln[X] + B(X * 10^{-3})$ where X has units of mm/s and P has units of Watts. The curvature of this relationship can be evaluated through determination of the second partial derivative of P with respect to X, thus yielding the quantity A/X. Considering that A can be any real number, the P(X) will be convex when A>0, linear when A=0, and concave when A<0.

B. Feed-direction MAM An understanding of the relationship between cutting force and undeformed chip thickness comes from Ref. [10], where a roughly linear relationship between undeformed chip thickness (h_0) and cutting force (Fp) is found for h_0>0.1 mm in plain carbon steel (see Figure 5.5). In the same figure, it was observed that at h_0<0.1 mm, the cutting force tended toward zero. The linear portion of this trace is classically referred to as an oblique asymptote and occurs in functions whose denominator is of order exactly 1 greater than that of the numerator. In order to ensure intersection with the origin, the asymptote's y-intercept must be cancelled out by a term in the function that must also independently go to zero for large values of the abscissa (so as not to affect the linear asymptote). From this, a general form for Fp(X) can come as $Fp = AX + B - \frac{B}{CX+1}$ where X is undeformed chip thickness (units of um) and A, B, and C are fitting parameters, each of which must be a real number greater than or equal to zero. The equation of this asymptote can be found by making X sufficiently large; this effectively drives the B/(C*X+1) term to zero and leaves the linear part of this expression, A*X + B, unaffected. It is important to note that the equation for force is linear with respect to parameters A and B, but non-linear with respect to C. An approach to quantify these parameters may be as follows: the values for A and B can be determined by fitting a line through two empirically-obtained data points, then C can then be determined using the Newton-Raphson iterative algorithm to minimize the residual sum of squares. Given Fp(X), the P(X) equation can be written as $P = V_{DC}\left[AX + B - \frac{B}{CX+1}\right]$ where V_{DC} has units of m/s and P has units of Watts. Computing the second partial derivative of P with respect to X yields $(-2*V*B*C^2)/(C*X+1)^3$. Considering that A, B and C must each be real numbers greater than or equal to zero, P(X) in this form will be linear when B and/or C is zero and will be concave when B and C are non-zero.

5.5 Numerical analysis methods

Data processing was handled using two custom Matlab scripts: one designed for data collected from the planer setup and the other for the lathe. These scripts are provided in Appendices G and H. The main features of the codes include the determination of a best-fit function for X, force prediction based on X and the determination of both actual and predicted specific energy consumption. Both numerical routines automatically identify the times at which cutting is initiated and when it ends based on the magnitude of Fp and plots these boundaries using green and red dashed lines, respectively. The use of Fp to characterize these critical time values is justified due to the close correspondence between the Fp and Fq traces and the lower signal to noise ratio present in Fp as compared to Fq. In data collected using both the planer and lathe machines, it is common for the force values seen at the beginning and end of the cuts to be somewhat different from those seen in the middle, or “steady-state,” portion of the cut. Thus, each routine evaluates cutting velocity, power, and energy based on a specified window of data. For the planer, this window begins 5 mm into the cut and ends 20 mm into the cut. For the lathe, the window begins 0.1 mm into the cut and ends 0.1 mm before the cut finishes. The beginning and ending of each window is represented using thin green and red dotted lines, which naturally occur inside the boundary limits of the overall cut. Lastly, the routines also use the Fp trace to determine the times at which each plateau begins and ends and identifies these times using green dots and red x's, respectively, which are positioned along the abscissa.

A primary difference between the two codes is the numerical integration scheme by which cutting energy is calculated. From traditional metal cutting mechanics, the total energy (E) consumed in a material removal process is E=Fp*Veff*Δt [1]. For both the planer and lathe data, a numerical technique for rectangular integration can be expressed symbolically as $E=\Sigma[Fp_i*dx_i]=\Sigma[Fp_i*Veff_i*dt_i]$. Note that the use of majuscule E is important here as the energy has not been normalized by volume. Due to the small dt’s used (1e-4 and 2e-4 s for planer and lathe data, respectively), trapezoidal integration was found to yield nearly identical results (<0.01% difference). For feed-direction modulation, the rectangular integration technique simplifies to $E=V_{DC}*dt*\Sigma[Fp_i]$ since Veff and dt are constant as a function of time. However, this is not the case for velocity-direction MAM wherein Veff is time-varying. In this case, energy is calculated according to $E=dt*\Sigma[Fp_i*Veff_i]$. Once energy has been obtained, the quantity can be normalized by volume.

Both routines were validated by first analyzing simulated, noise-free control data. Estimated values for amplitude and frequency of X, as well as actual and predicted specific energy were compared against pre-programmed values (e.g., Figs. 5.7-5.10) and the results of such are given in Tables 5.1-5.2. The theoretical values for amplitude and frequency of X were determined analytically while the specific energy values were determined using offline numerical integration using time steps (dt) at least three orders of magnitude smaller than those of the analysis codes. As can be seen in the figures and in the tables, the computational scheme used here tracks well with the expected result in the simulated cases presented. Finally, it should be noted that saturation limits were designed into the routine such as to prevent negative values for predicted power. This is justified by the assumption that the machining system is perfectly rigid; thus, cutting forces can only exist as positive values and are concurrent with the forward advancement of the tool through the work.

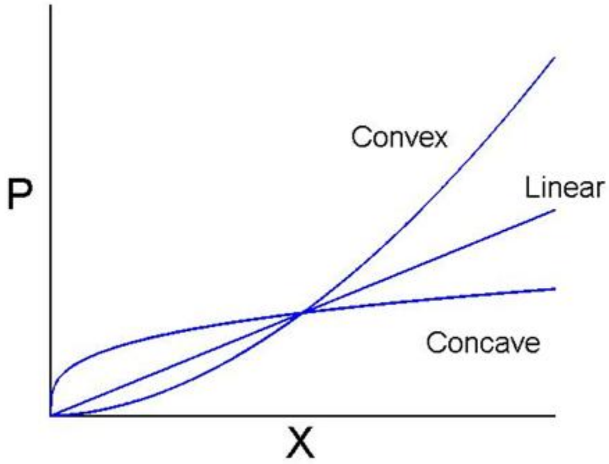


Figure 5.1: General shape of convex, linear and concave functions.

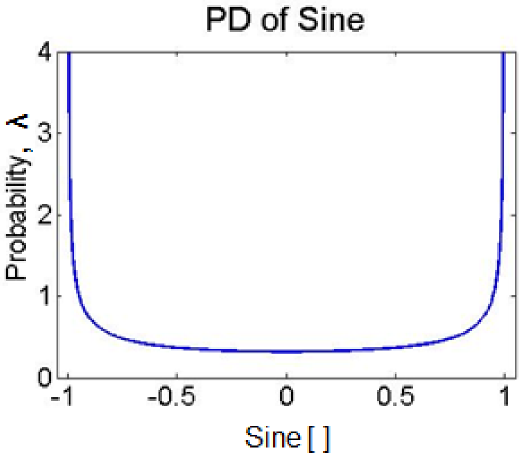


Figure 5.2: Probability density, λ, of the sine function.

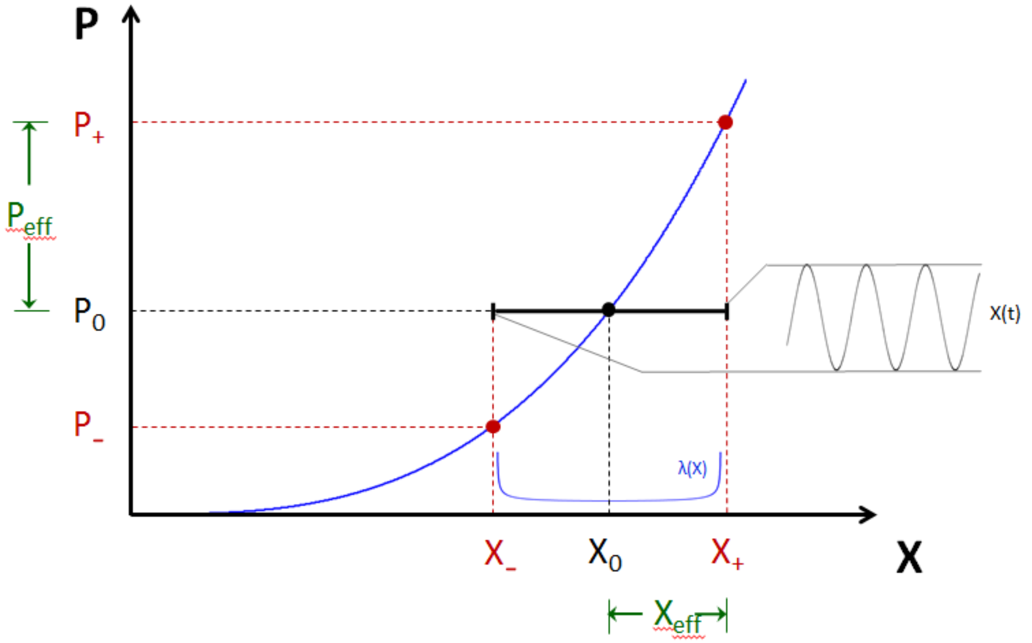


Figure 5.3: Graphical explanation of P/X variables and their relative magnitudes for a convex function.

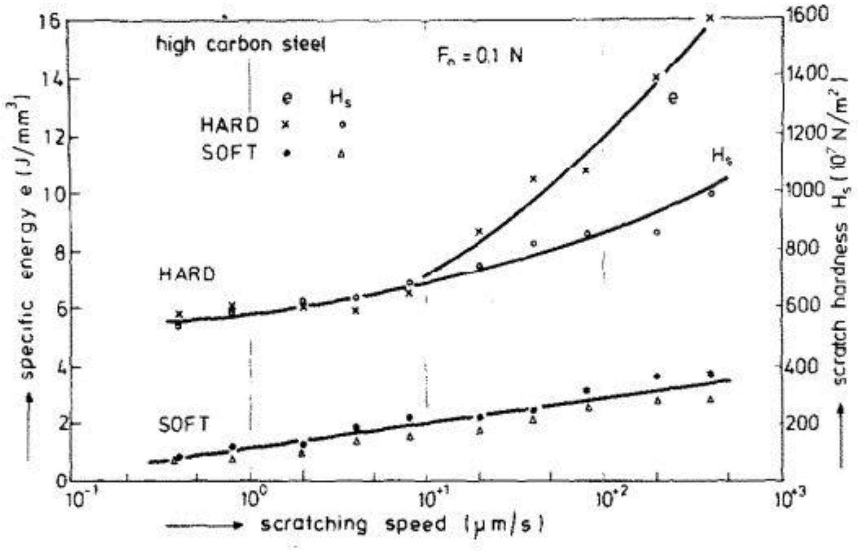


Figure 5.4: Reference indicating logarithmic relationship between cutting speed and specific energy consumption [11].

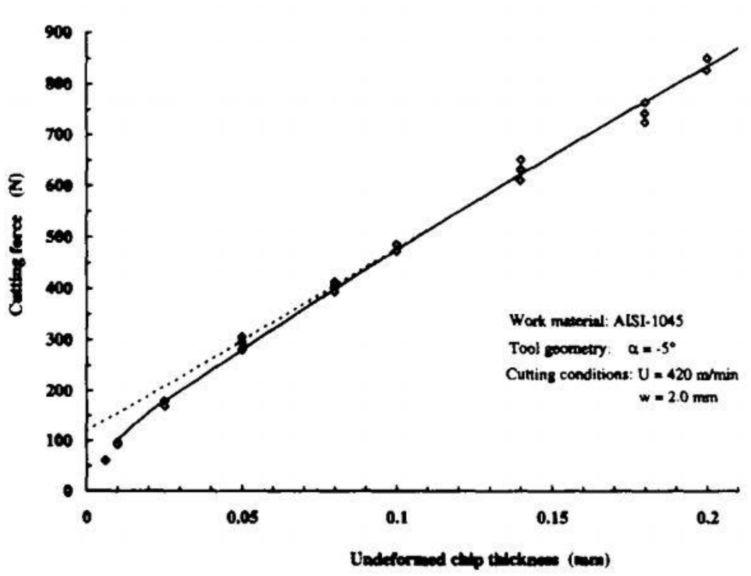


Figure 5.5: Reference indicating oblique asymptote for force-undeformed chip thickness relationship at high values of the abscissa and tending toward the origin at low values [10].

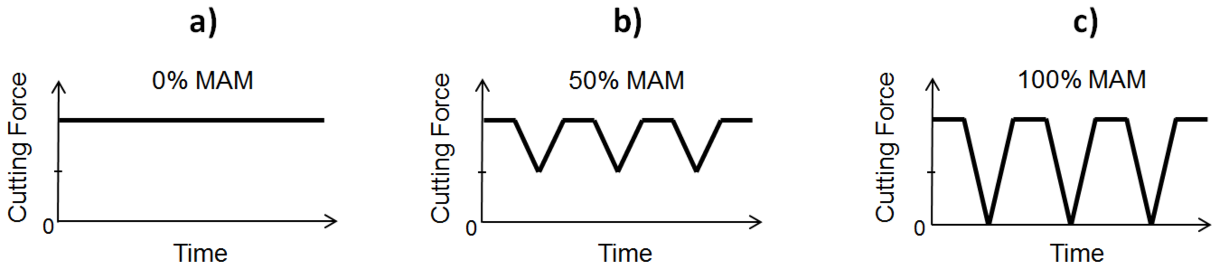


Figure 5.6: Examples of 0% (a), 50% (b) and 100% (c) MAM, calculated based on range of cutting force.

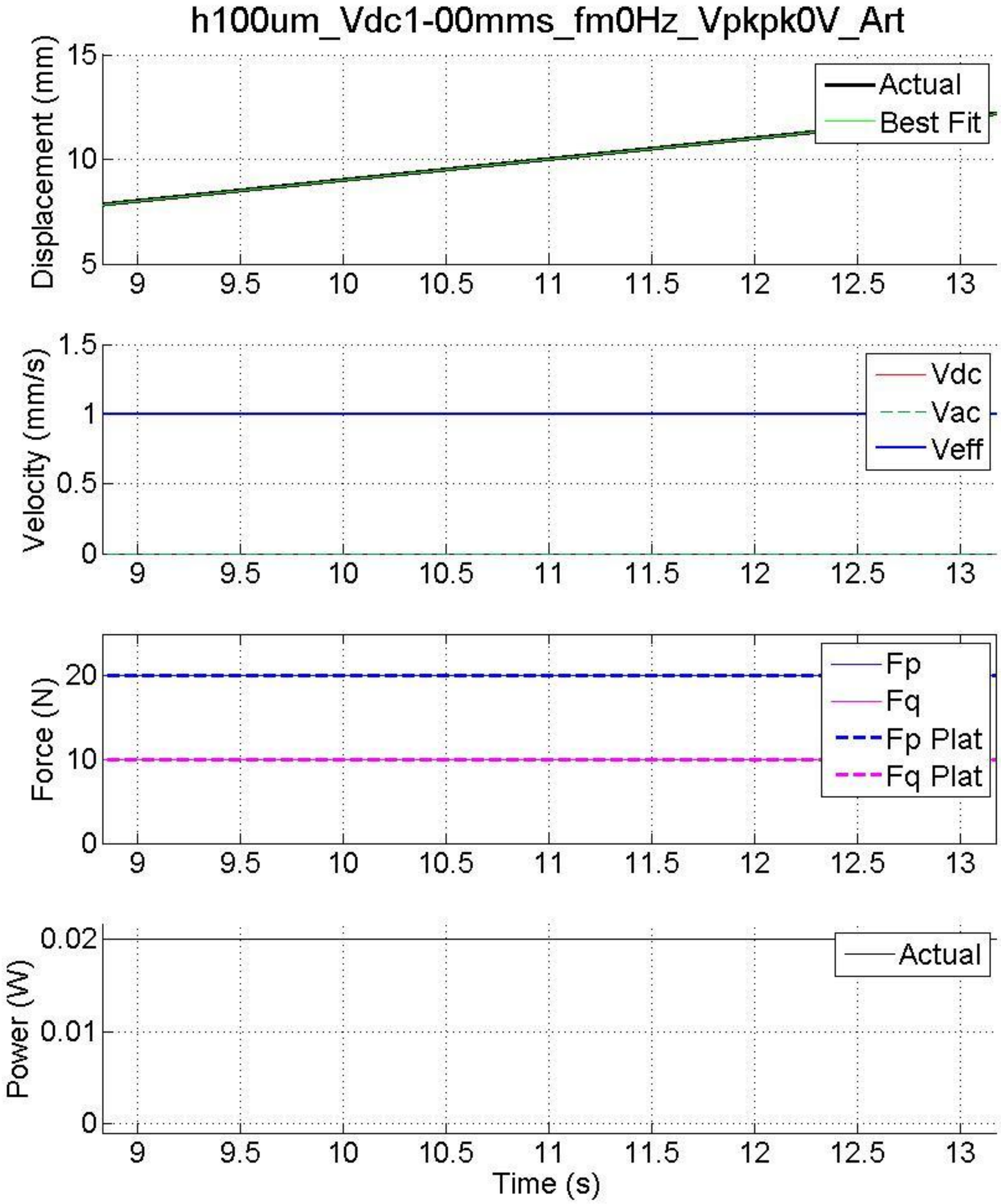


Figure 5.7: Results from numerical analysis of artificial data simulating a conventional cut at Vdc=1.00mm/s using planer machine.

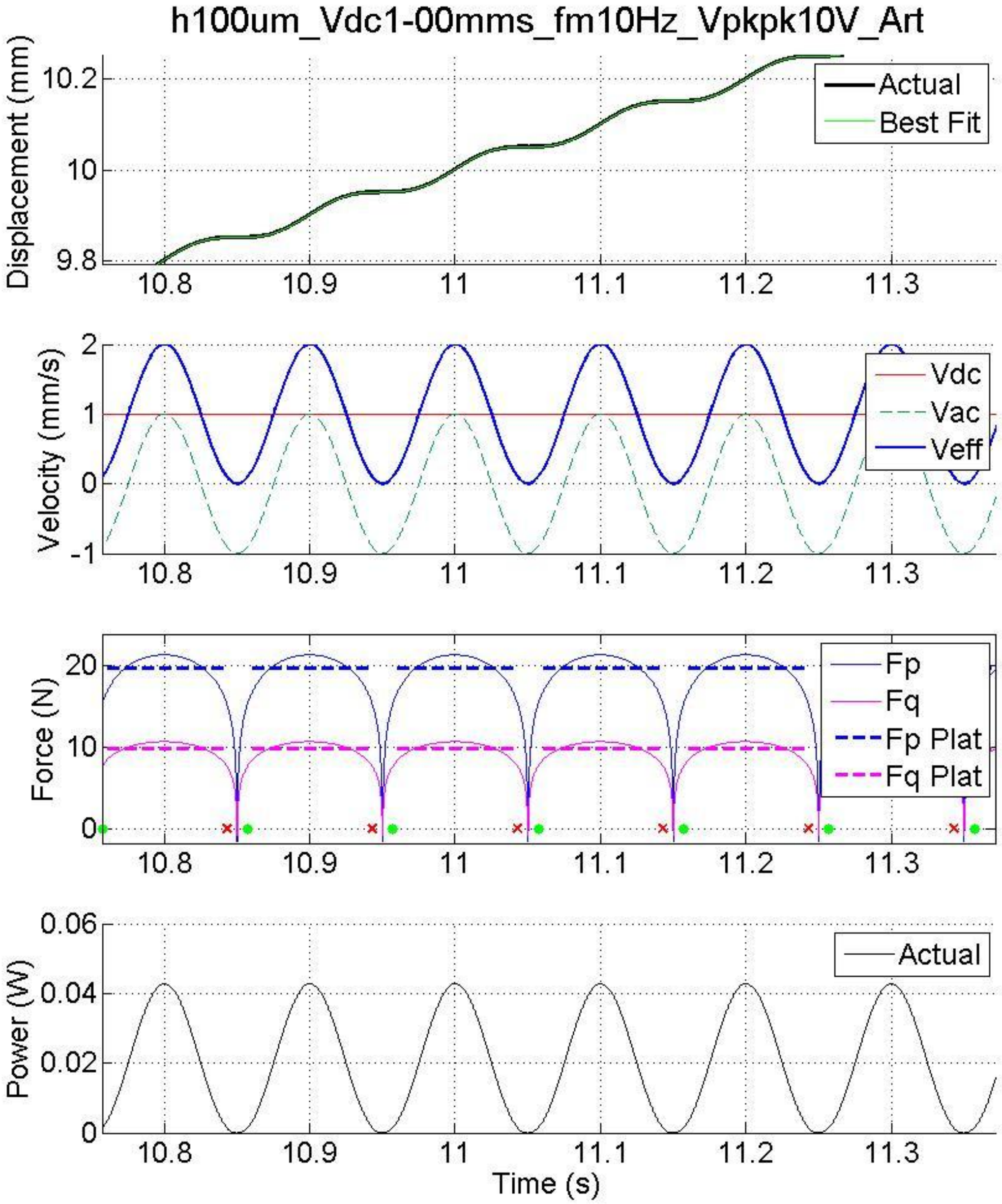


Figure 5.8: Results from numerical analysis of artificial data simulating velocity-direction MAM at Vdc=1.00mm/s, fm=10Hz, K=15.92um, 100% MAM using planer machine.

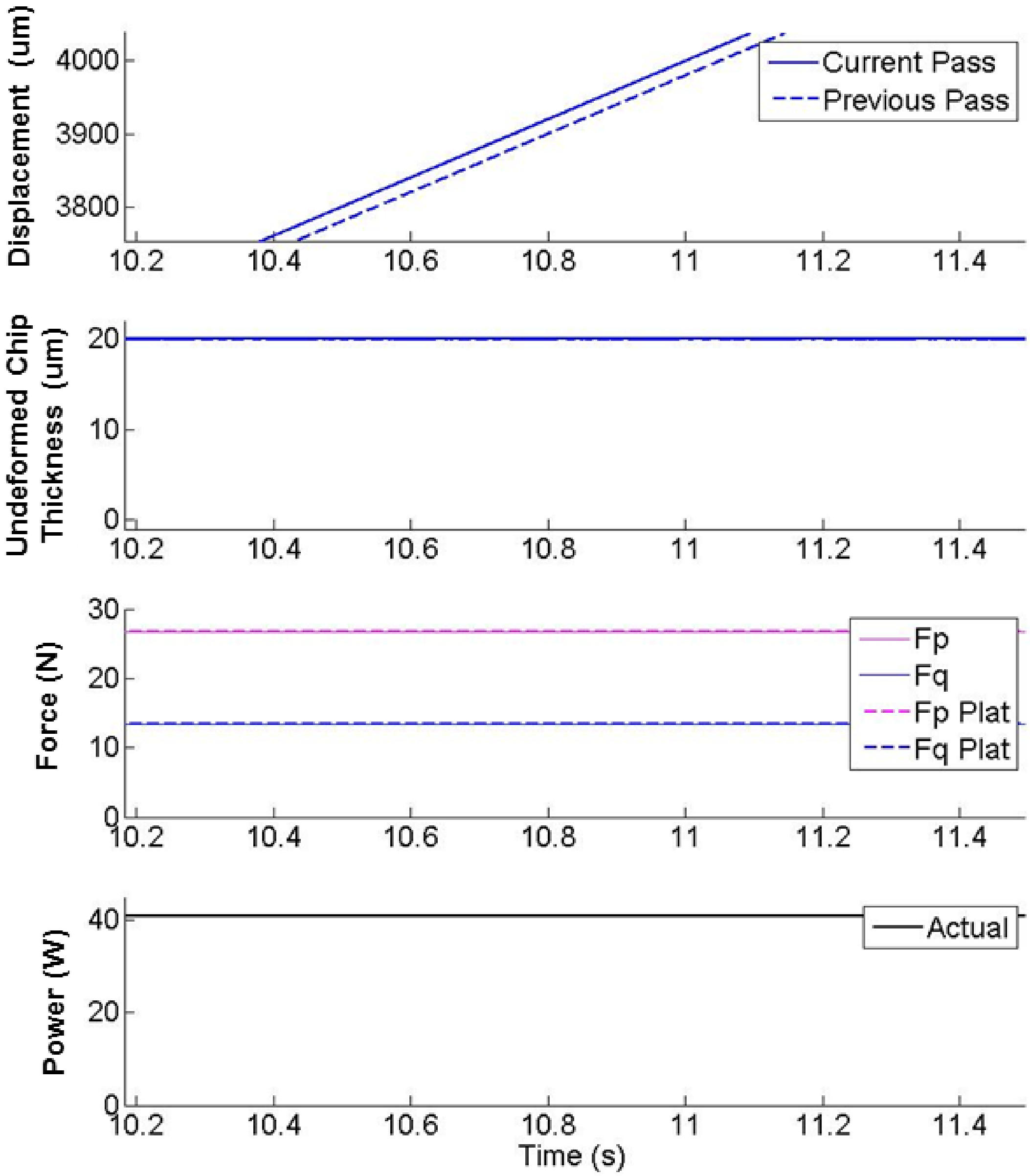


Figure 5.9: Results from numerical analysis of artificial data simulating a conventional cut at s=0.020mm/rev, w=1200rpm using lathe.

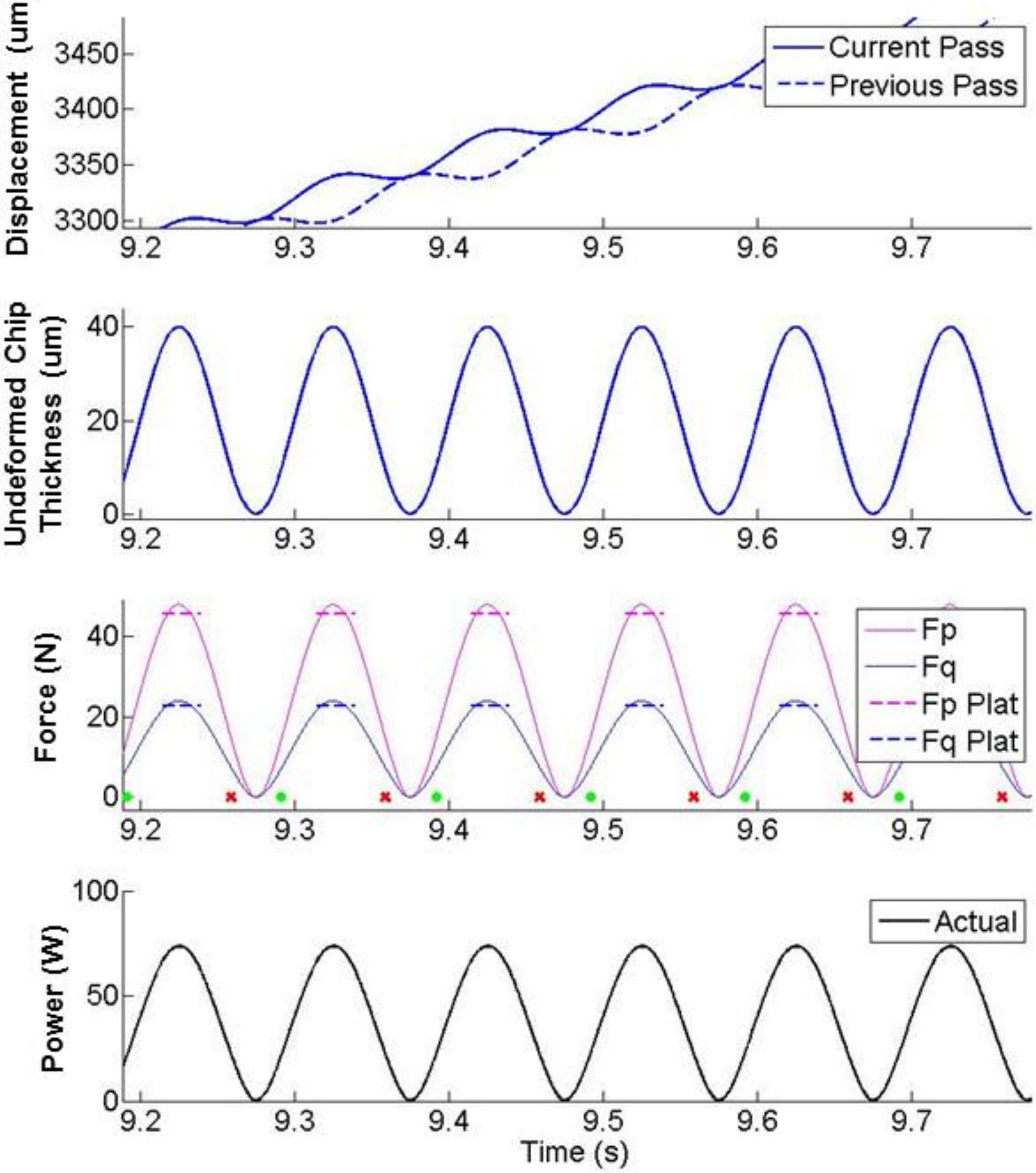


Figure 5.10: Results from numerical analysis of artificial data simulating feed-direction MAM at s=0.020mm/rev, w=1200rpm, fm=10Hz, K=0.010mm, 100% MAM using lathe.

Table 5.1: Comparison of theoretical values with those found using velocity-direction Matlab analysis routine

Velocity-Direction MAM	**Theor.**	**Act.**	**Pred.**	**Theor.**	**Act.**	**Theor.**	**Act.**
	e	**e**	**e**	**fm**	**fm**	**K**	**K**
	(J/mm^3)	**(J/mm^3)**	**(J/mm^3)**	**(Hz)**	**(Hz)**	**(um)**	**(um)**
Conventional	0.2000	0.2000	0.2000	-	-	-	-
100% MAM	0.2061	0.2061	0.2061	10.000	10.000	15.916	15.916

Table 5.2: Comparison of theoretical values with those found using feed-direction Matlab analysis routine

Feed-Direction MAM	**Theor.**	**Act.**	**Pred.**	**Theor.**	**Act.**	**Theor.**	**Act.**
	e	**e**	**e**	**fm**	**fm**	**K**	**K**
	(J/mm^3)	**(J/mm^3)**	**(J/mm^3)**	**(Hz)**	**(Hz)**	**(um)**	**(um)**
Conventional	1.3333	1.3333	1.3333	-	-	-	-
100% MAM	1.2764	1.2764	1.2765	10.000	9.999	10.000	9.996

Chapter 6: Results

6.1 Velocity-direction MAM

A. Forces and displacements Fig. 6.1 shows the measurement of force dissipated in conventional machining of AA6061-T6 at 0.05 mm/s with an undeformed chip thickness of 20um as well as the displacement (and velocity) of the linear stage over time. From the figure, it is clear that the cutting force, Fp, and the tangential force, Fq, rapidly rise and fall at the start and end of cutting, respectively. In the steady-state region, Fp and Fq are relatively constant over the length of the measurement at Fp,plat=19.86±1.15 N and Fq,plat=6.28±0.82 N, respectively. Figs. 6.2 and 6.3 show similar measurements at higher cutting speeds of 0.50 mm/s and 1.00 mm/s, respectively. While the magnitudes of the cutting and tangential forces appear quite similar of that of the low speed case, the length of the traces is noticeably smaller due to the higher velocities used. Figs. 6.4-6.6 show the force dissipated in conventional machining of OFE Cu under equivalent machining conditions. From these figures it is clear that the nature of the traces is fundamentally similar to that of the AA6061-T6 measurements. However, the force dissipated is not constant across the range of velocity; at the lower end of the velocity range, Fp is noticeably lower at 14.70±0.71 N for 0.05 mm/s compared to 20.04±0.99 N for 1.00 mm/s. A summary of these measurements is provided in Table 6.1.

The force response in modulation-assisted machining is quite different than that which occurs in conventional machining. Fig. 6.7a shows the force dissipated and displacement measurement in modulation-assisted machining the aluminum alloy at 0.50 mm/s, 10 Hz and 19 um amplitude. This same trace is shown at higher magnification over a smaller time range in Fig. 6.7b. The displacement of the workpiece with respect to time, which was linear in the conventional machining case, now also exhibits sinusoidal character due to the application of the modulation. The effective cutting velocity, as a result, exhibits sinusoidal character. These effects cause the force signature to contain clear periodicity, with a plateau value that occurs between trace segments where the force rises and falls to local minimum values (Fp,min>0 N and Fq,min≈0 N). The beginning and end of these valleys in the force signature correspond to periodic separation of the tool from the workpiece. The load observed at the plateau in the force trace is concurrent with maximal advance of the tool into the workpiece. From the figure, the forces have plateau values of Fp,plat=20.37±0.81 N and Fq,plat=4.88±0.29 N, similar to that seen in the conventional machining case for a speed of 0.50 mm/s. A similar trend as that seen for AA6061-T6 is seen in the modulation-assisted machining of OFE copper (Fig. 6.8a and 6.8b), where modulation changed the nature of the force trace and resulted in similar plateau values as those observed in conventional machining at a similar initial cutting speed.

B. Power dissipated The instantaneous power dissipated in modulation-assisted machining also exhibited periodicity due to the periodicity in instantaneous velocity and in instantaneous force Fp. The peak power values occur simultaneously with the peak effective cutting velocities. Unlike the cutting force, the peak power dissipated during modulation-assisted cutting ~.025W is significantly larger than that which was measured for conventional machining (~.010W). This was attributed to the fact that the instantaneous velocity at these peak power values was higher than in conventional machining. Figs. 6.9 and 6.10 show the effect of cutting velocity on power dissipated in conventional machining of both materials. From the figures, it is clear that power dissipated increases monotonically with cutting speed. To derive an empirical relationship between power and cutting velocity, P(X), where X is the modulated variable (cutting velocity), conventional machining data was fitted according to the function $\mathrm{Fp} = \mathrm{A}\ln[\mathrm{X}] + \mathrm{B}$ where Fp has units of N and X of mm/s, as described in Chapter 5. A least squares fit of the AA6061-T6 conventional machining data yielded constants of A=0.543 and B=21.5 with a resultant $R^2=0.972$. Fitting the OFE copper conventional machining data in a similar fashion yielded A=1.72 and B=19.4 with an $R^2=0.926$. These same coefficients were incorporated to fit the P(X) relationship for the aluminum alloy and the copper, yielding values of $R^2=1.000$ and $R^2=0.998$, respectively. The fit curves are also plotted in Figs. 6.9 and 6.10. From the figures, a greater degree of convexity in P(X) is evident for the copper ($\partial^2 P/\partial X^2=1.1e\text{-}3$ for X=0.5 mm/s) than for the aluminum alloy ($\partial^2 P/\partial X^2=3.4e\text{-}3$ at X=0.5 mm/s). For both materials, it is clear that the P(X) relationship is a good indicator of instantaneous power in modulation-assisted machining.

C. Energy consumption For conventional machining, cutting energy (E) was obtained by numerical integration of the cutting force (Fp) measurement and the instantaneous velocity data from the encoder. This quantity was then normalized by volume to obtain specific energy consumption (e) (see Chapter 5). The specific cutting energies for conventional and modulation-assisted machining of AA6061-T6 at $V_{DC}=0.50$ mm/s were 1.056±0.021 J/mm^3 and 1.047±0.013 J/mm^3, respectively (Fig. 6.11). In order to qualify these results statistically, a hypothesis test was performed with a null hypothesis of h_0: $\bar{e}_{conv}=\bar{e}_{mam}$ and alternative of h_1: $\bar{e}_{conv}<\bar{e}_{mam}$, where $\bar{e}$ represents the sample's average specific energy value. The null hypothesis was not rejected for an $\alpha=0.05$ level of significance. A plot of the probability density functions for conventional machining and modulation-assisted machining can be seen in Fig. 6.12. From this figure, one can see the relative degree of similarity between the statistically expected energy consumption values for these cases. The specific cutting energy for OFE copper was also evaluated. For conventional machining, the specific energy was 0.921±0.020 J/mm^3 while that of modulation-assisted machining was 0.947±0.019 J/mm^3, (see Fig. 6.13). The same hypothesis test was performed for copper and it was determined that the null hypothesis should be rejected at the α=0.05 level of significance. Thus, it can be said that modulation-assisted machining requires more energy than conventional cutting in OFE copper at a 95% confidence level. A plot of the probability density functions for conventional and modulated cutting can be seen in Fig. 6.14. From this, it is easy to see the difference between the statistically expected energy values for these two types of cutting. To summarize, the specific cutting energy for modulation-assisted machining is seen to be higher for OFE copper and approximately the same for AA6061-T6 when compared with that of conventional machining. These general trends are not surprising considering the P(X) relationship for both materials, which exhibited more convexity in the case of OFE copper.

6.2 Feed-direction MAM

A. Forces and displacements Figs. 6.15-6.17 show measurements of force dissipated in conventional machining of AA6061-T6 over a range of feed rates from 0.005 mm/rev to 0.050 mm/rev. From the figures it is clear that the morphology of the force traces is quite similar to that of the earlier conventional machining experiments, but the range over which it increases is significantly higher. For example, the cutting force at a feed rate of 0.005 mm/rev was Fp,plat=21.29±1.31 N while that at 0.050 mm/rev was Fp,plat=75.72±4.04 N. Figs. 6.18-6.20 show similar measurement in Ti3Al2.5V at equivalent feed rates as in the aluminum alloy. From these figures it is clear that the nature of the traces is fundamentally similar to that of the AA6061-T6 measurements. However, the magnitudes of the forces are clearly higher due to the titanium alloy's intrinsically higher strength. The cutting force increased from Fp,plat=32.38±0.72 N for 0.005 mm/rev to Fp,plat=120.38±2.26 N for 0.050 mm/rev. The measurements for conventional machining of both alloys are summarized in Table 6.2.

As was the case in velocity-direction modulation, the force response in feed-direction modulation is quite different than that which occurs in conventional machining. Fig. 6.21a shows the force dissipated and displacement measurement in feed-direction modulation-assisted machining for the aluminum alloy at 0.020 mm/rev, 10 Hz and 14.5 um amplitude. This same trace is shown at higher magnification over a smaller time range in Fig. 6.21b. The displacement of the tool with respect to time, which was linear in the conventional machining case, now exhibits sinusoidal character due to the application of the modulation. The sinusoidal displacement of the tool, as described in Chapter 3, causes the effective undeformed chip thickness to have sinusoidal character when fm/fw is an odd integer multiple of 1/2. This characteristic is also seen for modulation frequencies of fm=[50, 110 Hz], where the modulated amplitude is K=[15.0, 13.4 um] (see Figs 6.22a,b and 6.24a,b). The modulation of chip thickness causes the force signature to exhibit periodicity as well, with a plateau value that occurs between trace segments where the force rises and falls to zero. The load observed at the plateau in the force trace is concurrent with the maximum undeformed chip thickness. From Figs. 6.21a,b, the forces at 10 Hz have plateau values of Fp,plat=58.17±0.65 N and Fq,plat=50.25±1.49 N, both significantly higher than those seen in conventional machining with a constant undeformed chip thickness of 0.020 mm where Fp,plat=43.53±2.49 N and Fq,plat=37.95±3.67 N.

When the ratio fm/fw is an even integer multiple of 1/2, the undeformed chip thickness is expected to be constant as described in Chapter 3. This is shown in Figs. 6.23a,b for the aluminum alloy when fm=100 Hz. In this case, the force levels are constant despite the sinusoidal motion of the tool due to the constant undeformed chip thickness. Further, the magnitudes of the plateau forces, Fp,plat=43.19±1.60 N and Fq,plat=37.73±2.60 N, are seen to be similar to those of conventional machining with a constant undeformed chip thickness of 0.020 mm. Similar observations to those above can be made for the Ti3Al2.5V material, albeit at a higher force range. The force traces for the Ti3Al2.5V material are provided in Figs. 18-20 and 25a-28b and a summary of the measurements for both materials is provided in Table 6.2.

B. Power dissipated Modulation of the undeformed chip thickness caused periodic variation of power dissipation in both the aluminum and titanium alloys. Figs. 6.29 and 6.30 show the effect of undeformed chip thickness on power dissipation in conventional machining. From the figures, it is clear that power dissipated increases monotonically with increases in undeformed chip thickness. To derive an empirical relationship between power and undeformed chip thickness, P(X), where X is the modulated variable (undeformed chip thickness), conventional machining data was fitted according to the function $Fp = AX + B - \frac{B}{CX+1}$. For AA6061-T6, this yielded the fitting parameter values of A=0.869, B=32.3, and C=0.258, yielding R^2=0.992 when Fp has units of N and undeformed chip thickness units of um. For Ti3Al2.5V, these constants were determined to be A=1.94, B=23.4, and C=6.44, yielding R^2=1.000. The same fitting parameters were used to characterize the power dissipation data in conventional machining and the R^2 values were equivalent to those of the Fp(X) relationship as Fp(X) and P(X) have essentially identical shapes. Recall that for feed-direction modulation, P=F*V where V is constant. This can be compared with velocity-direction modulation where V is a function of time (see Figs. 6.1-6.8). Recall, from the discussion in Chapter 5, that the degree of concavity for the P(X) fit is related to the term $(-2*V*b*c^2) / (c*X+1)^3$. For X=20um, this term equates to -2.8×10^{-2} and -1.3×10^{-3} for AA6061-T6 and Ti3Al2.5V, respectively, indicating a greater degree of concavity for the aluminum alloy.

These empirically-derived P(X) relationships were then used to predict instantaneous power dissipation in modulation-assisted machining in the feed direction by using the measured instantaneous undeformed chip thickness derived from the displacement data; this is plotted in Figs. 21a-28b in red on the power traces. From the figures, it is clear that the P(X) relationship is generally a good indicator of instantaneous power in modulation-assisted machining, but noticeably over-predicts the peak in power dissipation for both materials.

C. Energy consumption For conventional machining, cutting energy (E) was obtained by numerical integration of the cutting force (Fp) measurement and the instantaneous velocity data from the encoder. This quantity could then be normalized by volume to obtain specific energy consumption (e) (see Chapter 5). Note the difference between this method and that used in velocity-direction modulation: here cutting velocity is constant and an effective velocity due to modulation is not necessary to determine power dissipation. As a result of this integration, the specific energy for conventional cutting of AA6061-T6 was found to be e=1.76 J/mm^3. This value compares to those values found under modulation conditions at fm=[10, 50, 100, 110 Hz] where e=[1.60, 1.58, 1.74, 1.54 J/mm^3]. This is summarized in Fig. 6.31 and Table 6.2. Note that in cases wherein fm/fw is an odd integer multiple of 1/2 (e.g., 10, 50, 110 Hz) that the average energy dissipation is noticeably lower than that of the conventional machining case. This can be compared with the specific energy for a situation wherein fm/fw is an even integer multiple of 1/2 (e.g., 100Hz), which was much closer to that of the conventional machining case. Indeed, the predicted values of specific cutting energy are quite close for all frequencies as the relative error for each of the modulation frequencies of fm=[10, 50, 100, 110 Hz] were [4.4, 2.6, 2.8, 0.1%], respectively. This indicates that the parameter modulation effects captured by the P(X) curve in conventional machining of the AA6061-T6 material is able to explain the lower energy consumption for modulation-assisted machining of the same material.

For the titanium alloy, the specific cutting energy was 2.38 J/mm^3, again higher than the aluminum alloy due to the higher flow stress of the titanium. This corresponding specific cutting energy values for modulation conditions at fm=[10, 50, 100, and 110 Hz] were e=[2.15, 2.04, 2.44, and 2.08 J/mm^3], as is shown in Fig. 6.32 and Table 6.2. As in the case of the aluminum alloy, the energy dissipated in modulated machining of the titanium was lower only for modulation conditions wherein fm/fw was an odd integer multiple of 1/2. However, this behavior is at odds with the predicted energy values using the P(X) curve, which were not substantially different than in the conventional machining case. The relative error in the predicted specific cutting energies for modulation frequencies of fm=[10, 50, 100, 110 Hz] were [9.5, 19.2, 1.7, 11.4%]. As only one trial was performed for each cutting condition, a statistical analysis like that performed for the velocity-direction modulation data would not be practical in this situation.

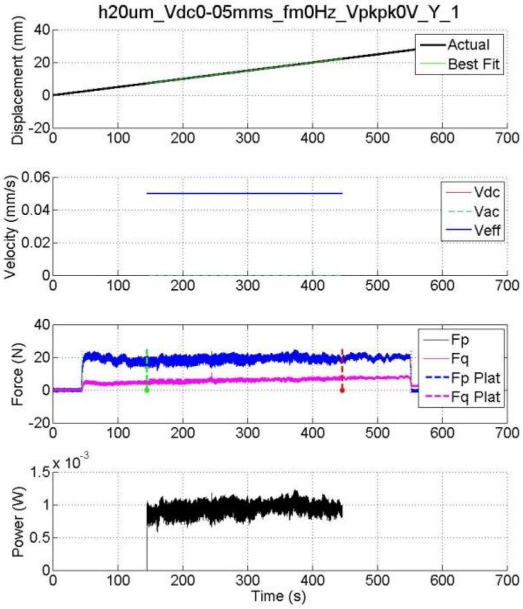


Figure 6.1: Numerical analysis results for AA6061-T6 cut at h0=20um and VDC=0.05mm/s using planer.

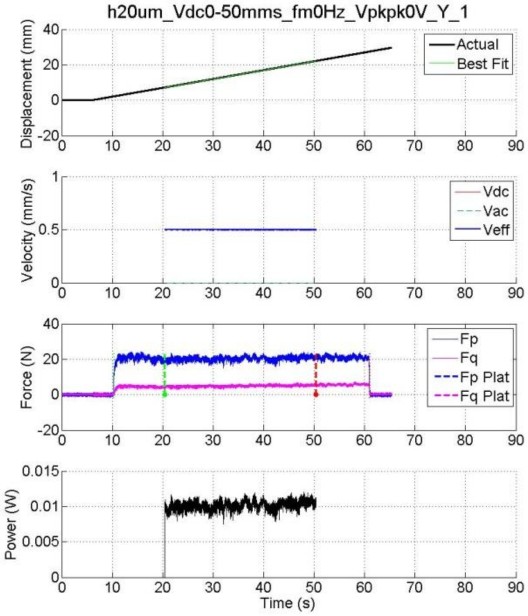


Figure 6.2: Numerical analysis results for AA6061-T6 cut at h0=20um and VDC=0.50mm/s using planer.

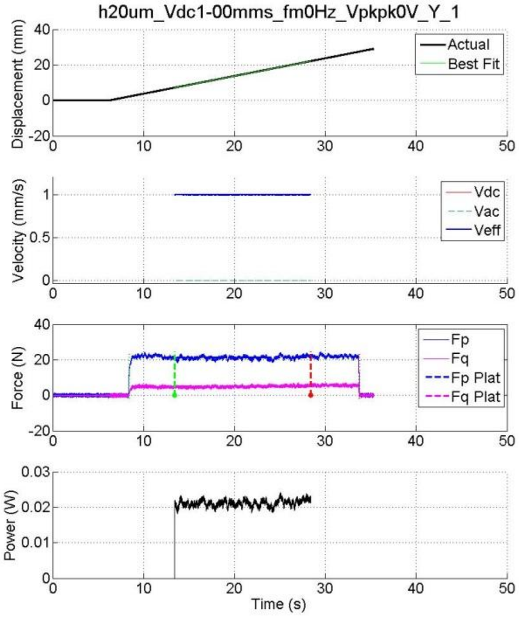


Figure 6.3: Numerical analysis results for AA6061-T6 cut at h0=20um and VDC=1.00mm/s using planer.

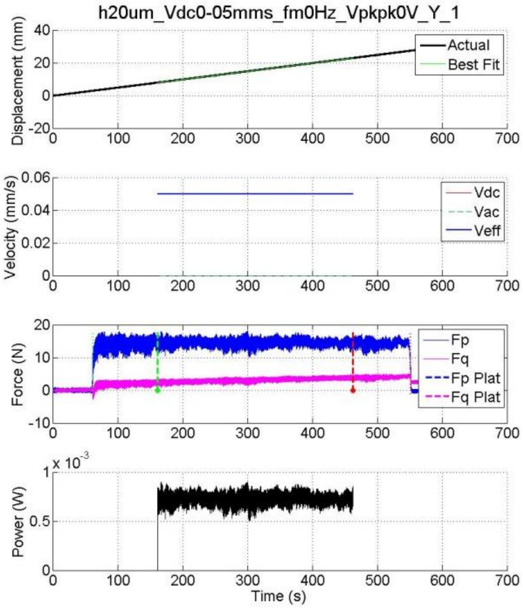


Figure 6.4: Numerical analysis results for OFE Cu cut at h0=20um and VDC=0.05mm/s using planer.

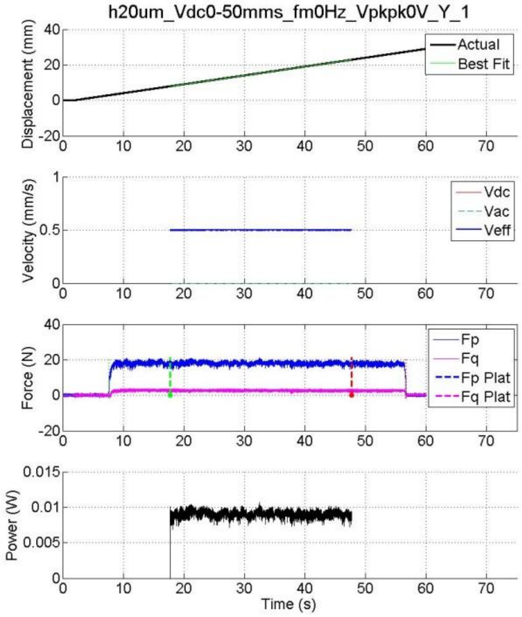


Figure 6.5: Numerical analysis results for OFE Cu cut at h0=20um and VDC=0.50mm/s using planer.

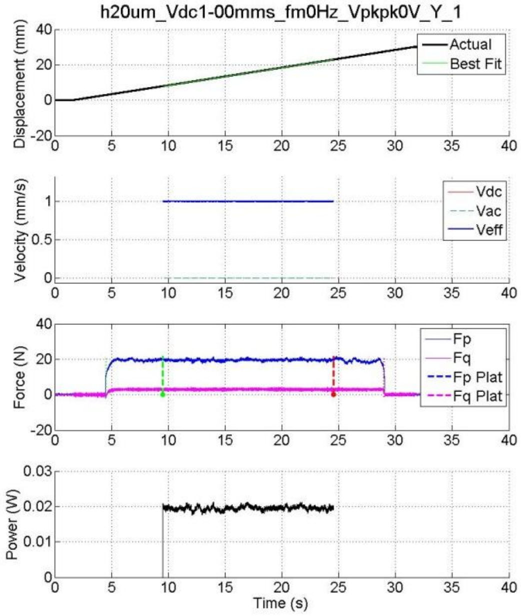


Figure 6.6: Numerical analysis results for OFE Cu cut at h0=20um and VDC=1.00mm/s using planer.

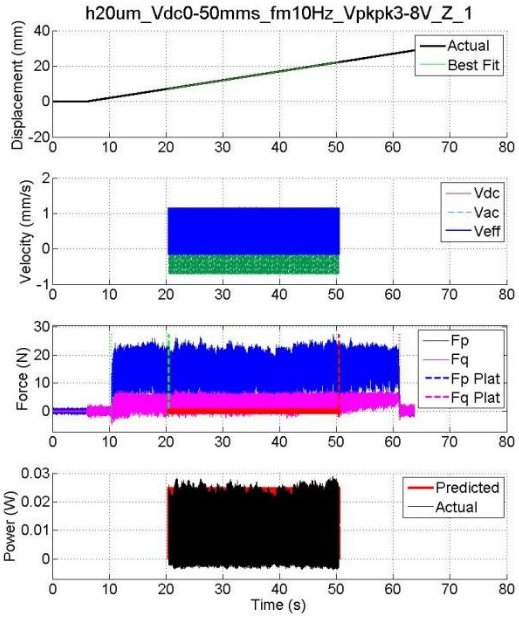


Figure 6.7a: Numerical analysis results for AA6061-T6 cut using planer at h0=20um, VDC=0.50mm/s, fm=10Hz, K=19V.

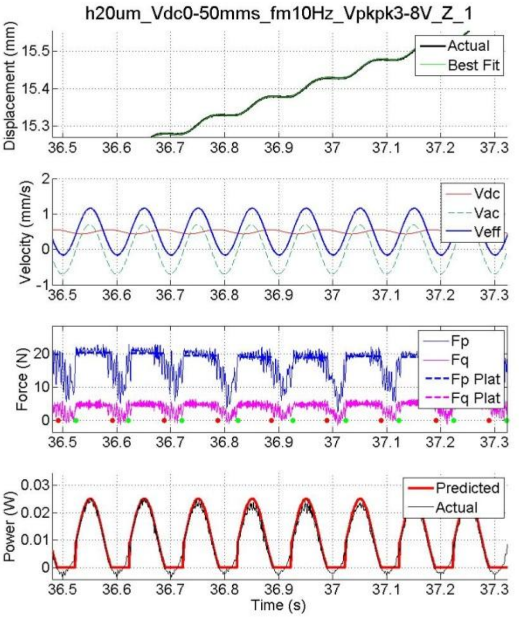


Figure 6.7b: Close-up of the numerical analysis results for AA6061-T6 cut using planer at h0=20um, VDC=0.50mm/s, fm=10Hz, K=19V.

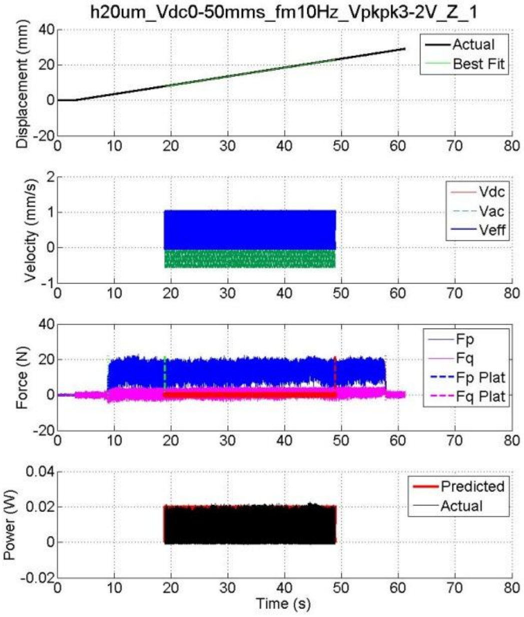


Figure 6.8a: Numerical analysis results for OFE Cu cut using planer at h0=20um, VDC=0.50mm/s, fm=10Hz, K=16V.

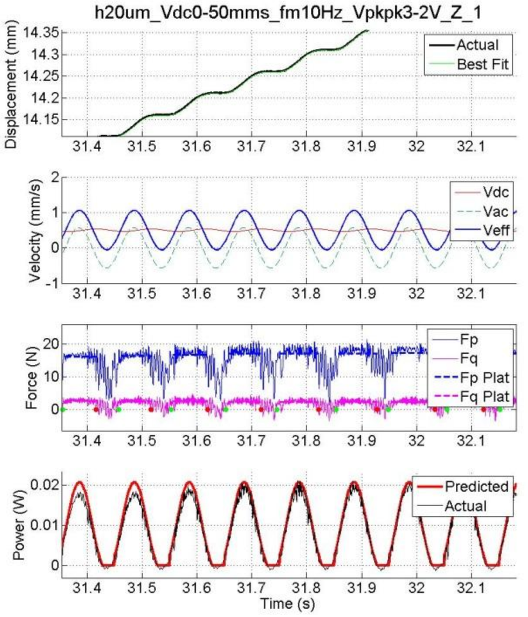


Figure 6.8b: Close-up of the numerical analysis results for OFE Cu cut using planer at h0=20um, VDC=0.50mm/s, fm=10Hz, K=16V.

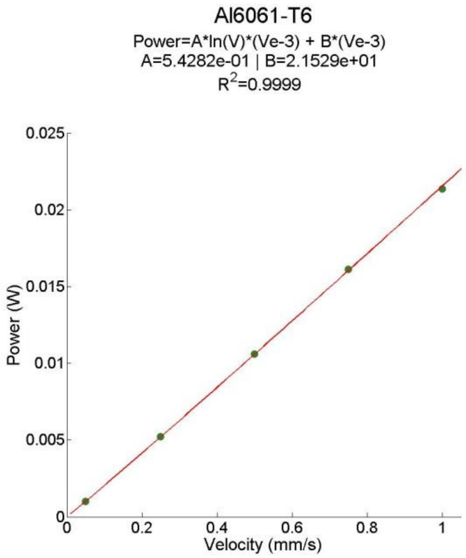


Figure 6.9: P(X) trace for AA6061-T6 where X denotes cutting speed.

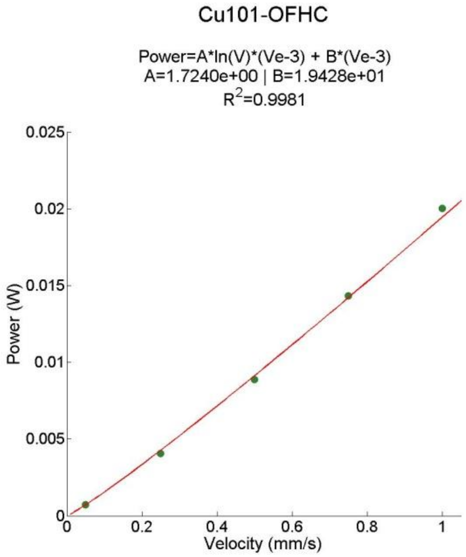


Figure 6.10: P(X) trace for OFE Cu where X denotes cutting speed.

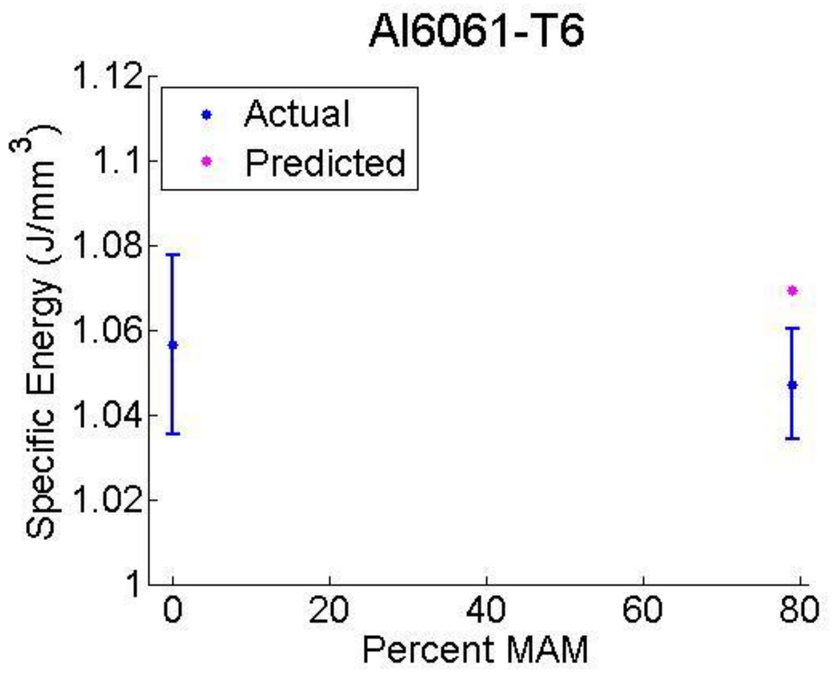


Figure 6.11: Specific energy consumption for AA6061-T6 cut with conventional and MAM conditions using planer.

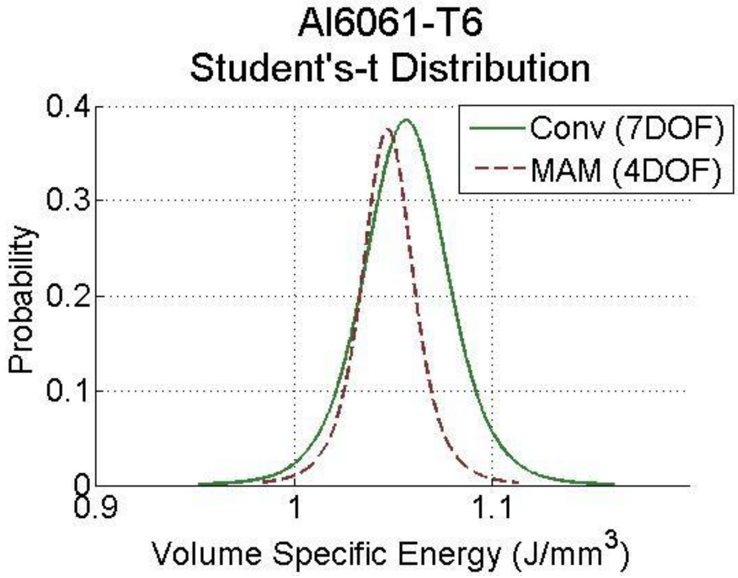


Figure 6.12: Probability density of t-distribution for AA6061-T6 cut with conventional and MAM conditions using planer.

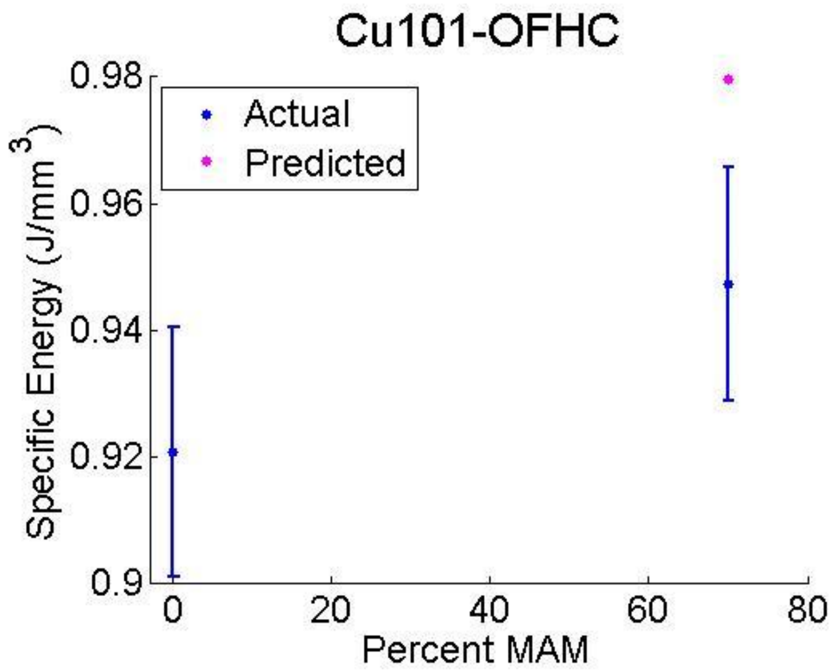


Figure 6.13: Specific energy consumption for OFE Cu cut with conventional and MAM conditions using planer.

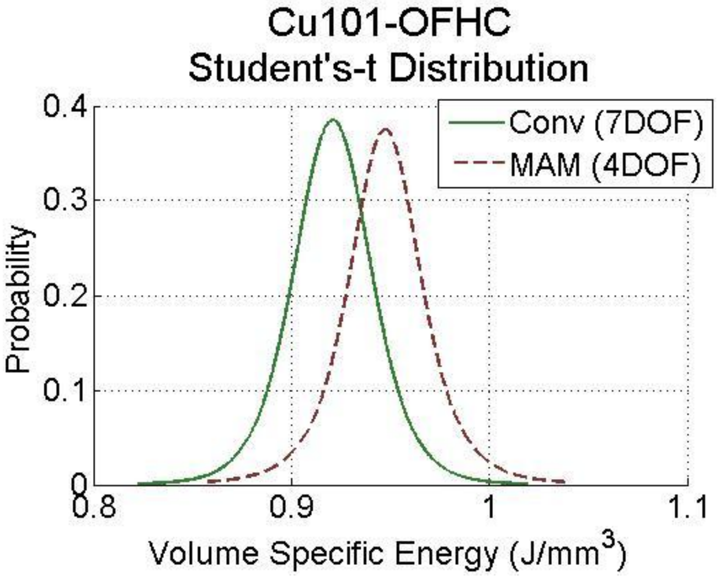


Figure 6.14: Probability density of t-distribution for OFE Cu cut with conventional and MAM conditions using planer.

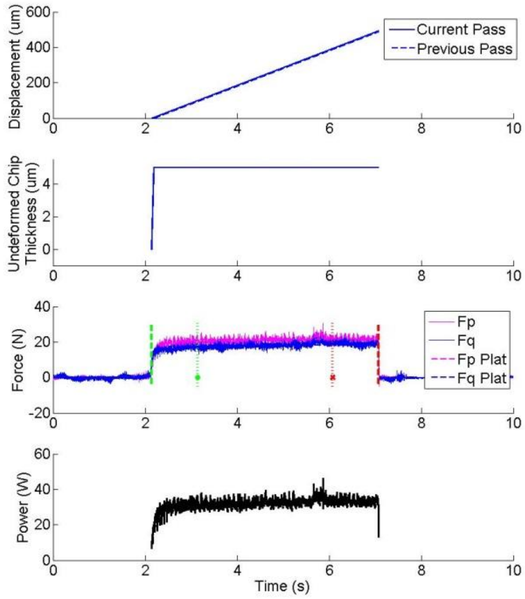


Figure 6.15: Numerical analysis results for AA6061-T6 cut at s=0.005 mm/rev using lathe.

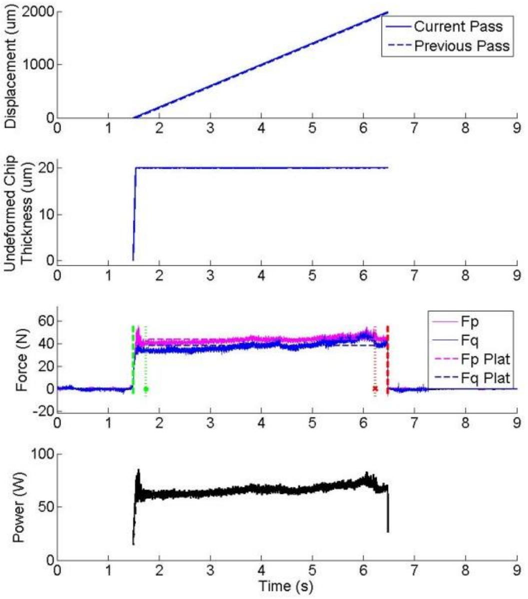


Figure 6.16: Numerical analysis results for AA6061-T6 cut at s=0.020 mm/rev using lathe.

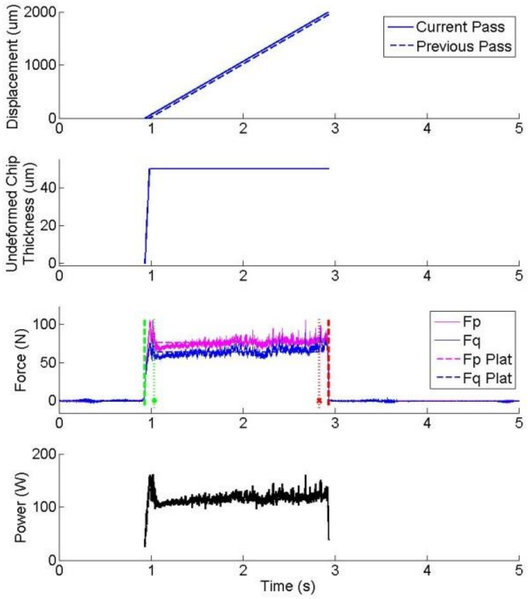


Figure 6.17: Numerical analysis results for AA6061-T6 cut at s=0.050 mm/rev using lathe.

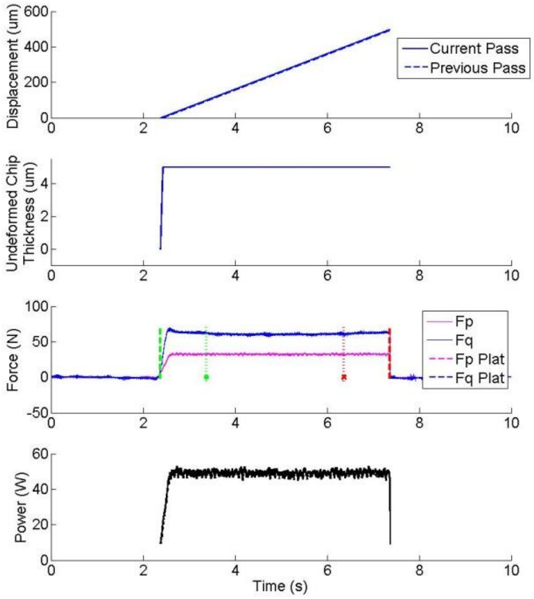


Figure 6.18: Numerical analysis results for Ti3Al2.5V cut at s=0.005 mm/rev using lathe.

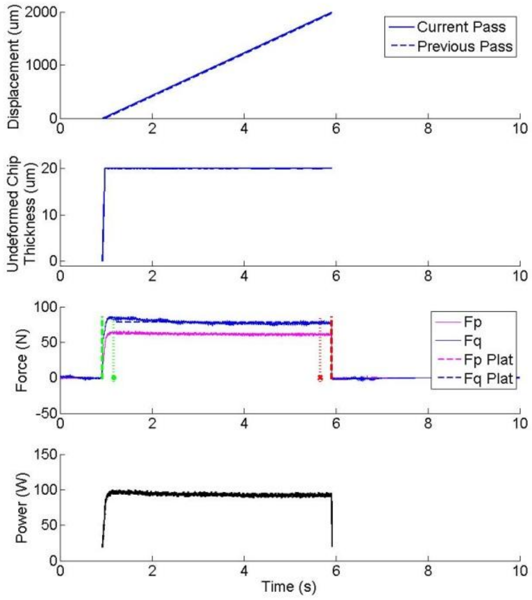


Figure 6.19: Numerical analysis results for Ti3Al2.5V cut at s=0.020 mm/rev using lathe.

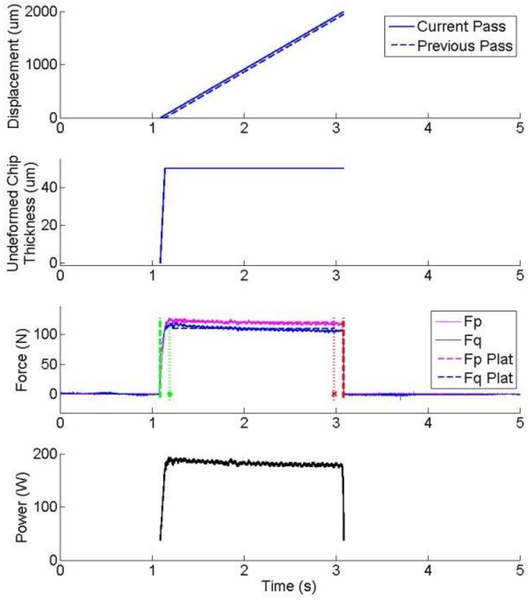


Figure 6.20: Numerical analysis results for Ti3Al2.5V cut at s=0.050 mm/rev using lathe.

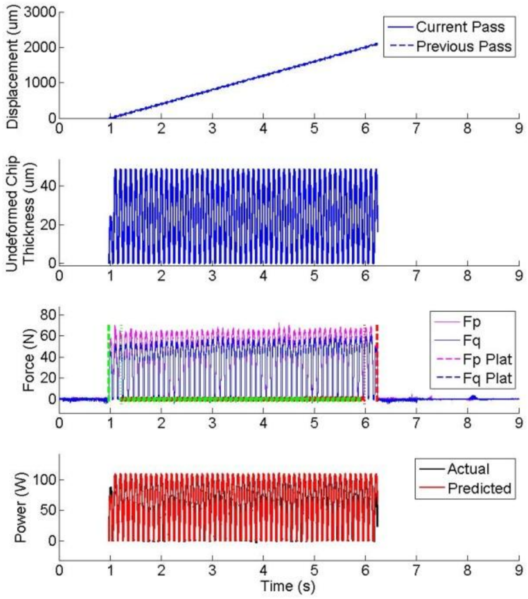


Figure 6.21a: Numerical analysis results for AA6061-T6 cut at s=0.020 mm/rev, fm=10Hz, K=60V using lathe.

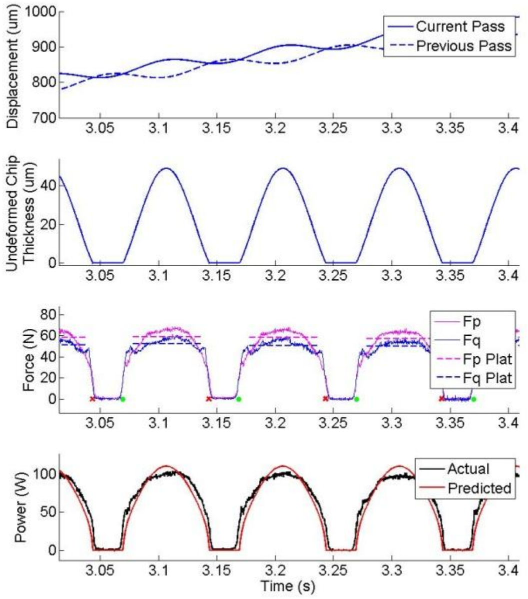


Figure 6.21b: Close-up of numerical results for AA6061-T6 cut at s=0.020 mm/rev, fm=10Hz, K=60V using lathe.

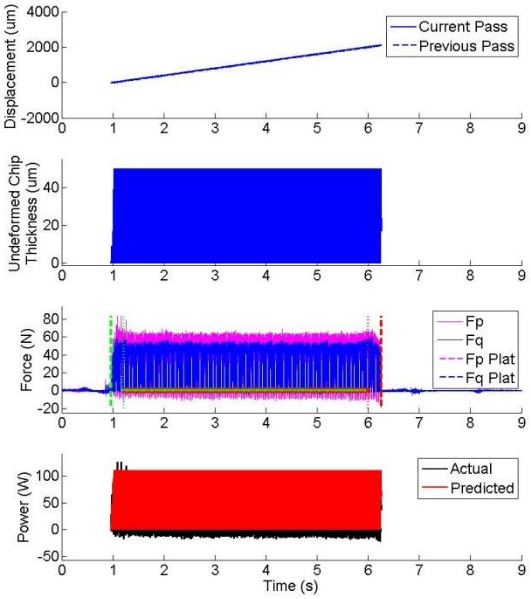


Figure 6.22a: Numerical analysis results for AA6061-T6 cut at s=0.020 mm/rev, fm=50Hz, K=60V using lathe.

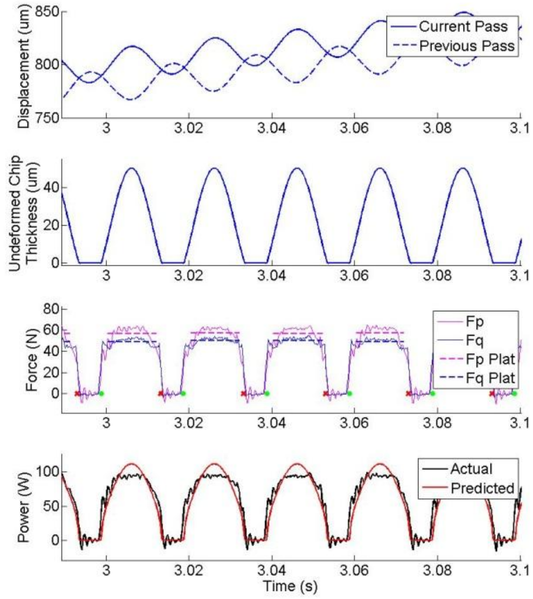


Figure 6.22b: Close-up of numerical results for AA6061-T6 cut at s=0.020 mm/rev, fm=50Hz, K=60V using lathe.

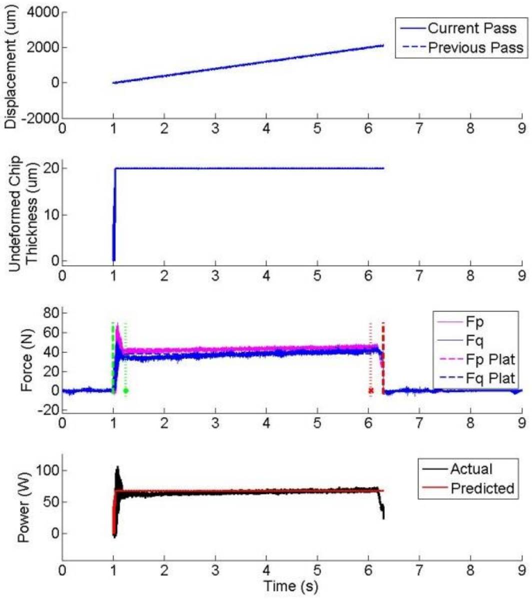


Figure 6.23a: Numerical analysis results for AA6061-T6 cut at s=0.020 mm/rev, fm=100Hz, K=60V using lathe.

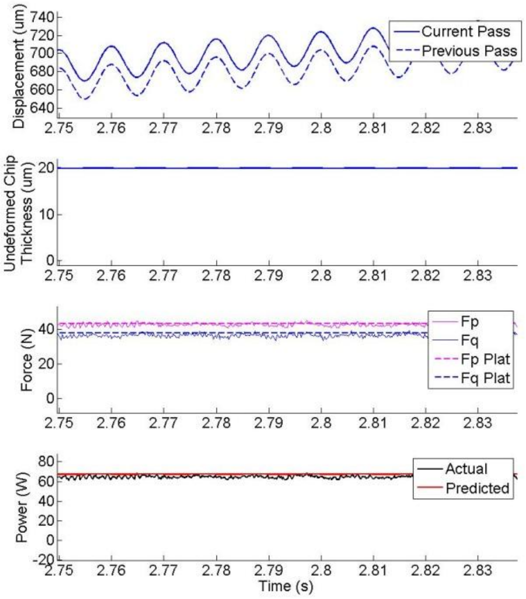


Figure 6.23b: Close-up of numerical results for AA6061-T6 cut at s=0.020 mm/rev, fm=100Hz, K=60V using lathe.

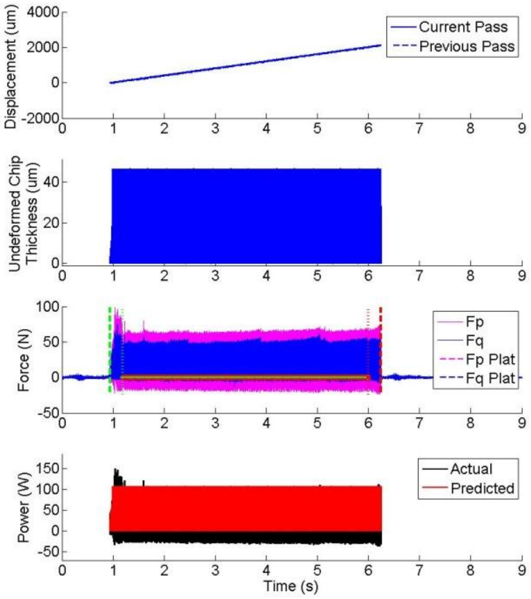


Figure 6.24a: Numerical analysis results for AA6061-T6 cut at s=0.020 mm/rev, fm=110Hz, K=60V using lathe.

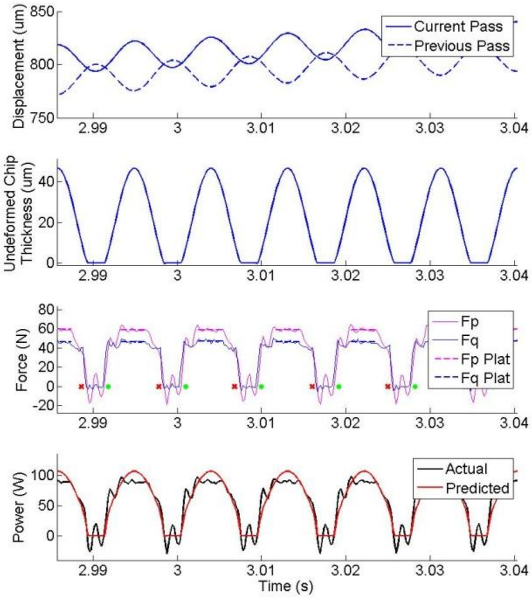


Figure 6.24b: Close-up of numerical results for AA6061-T6 cut at s=0.020 mm/rev, fm=110Hz, K=60V using lathe.

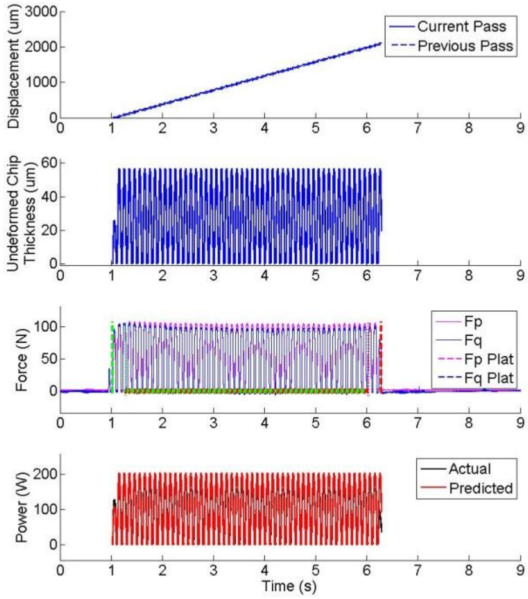


Figure 6.25a: Numerical analysis results for Ti3Al2.5V cut at s=0.020 mm/rev, fm=10Hz, K=60V using lathe.

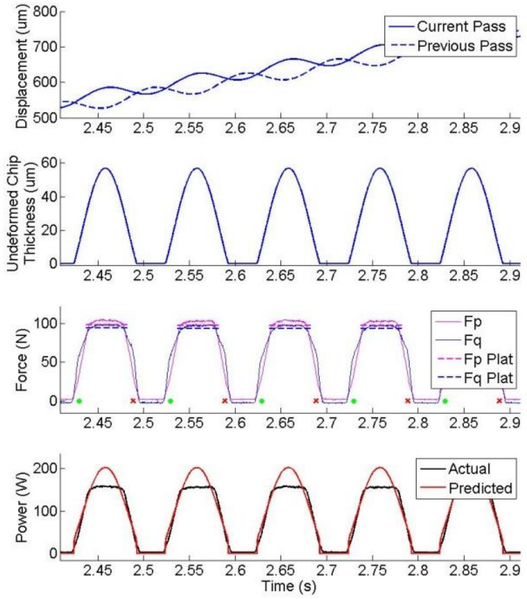


Figure 6.25b: Close-up of numerical results for Ti3Al2.5V cut at s=0.020 mm/rev, fm=10Hz, K=60V using lathe.

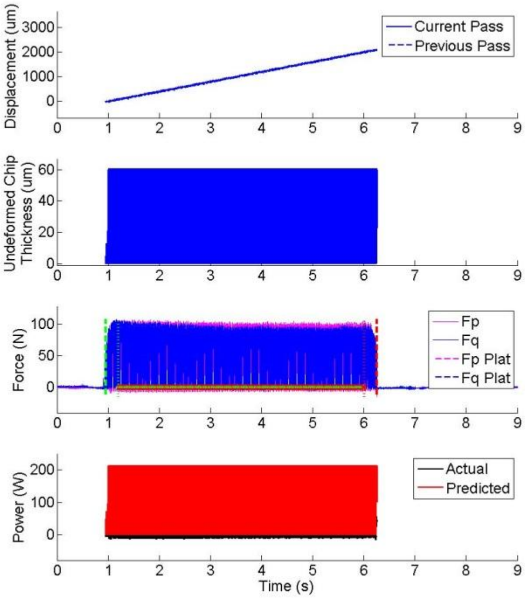


Figure 6.26a: Numerical analysis results for Ti3Al2.5V cut at s=0.020 mm/rev, fm=50Hz, K=60V using lathe.

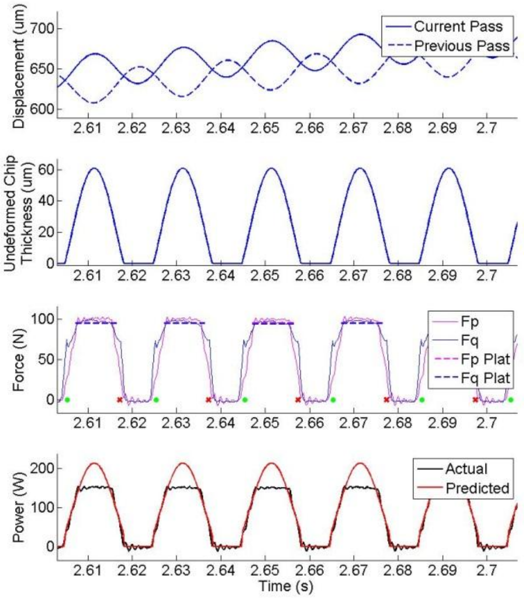


Figure 6.26b: Close-up of numerical results for Ti3Al2.5V cut at s=0.020 mm/rev, fm=50Hz, K=60V using lathe.

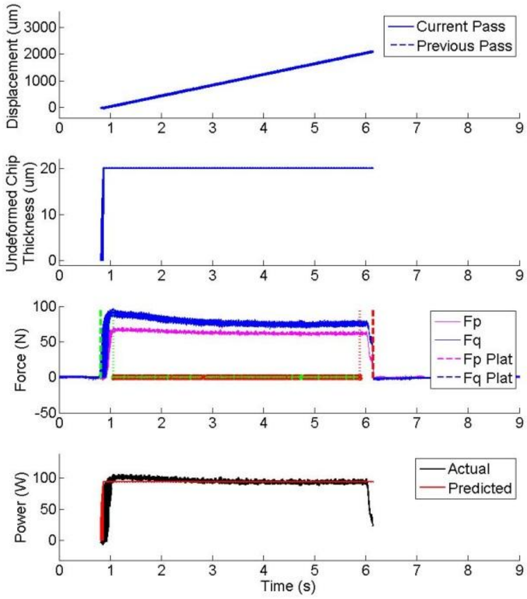


Figure 6.27a: Numerical analysis results for Ti3Al2.5V cut at s=0.020 mm/rev, fm=100Hz, K=60V using lathe.

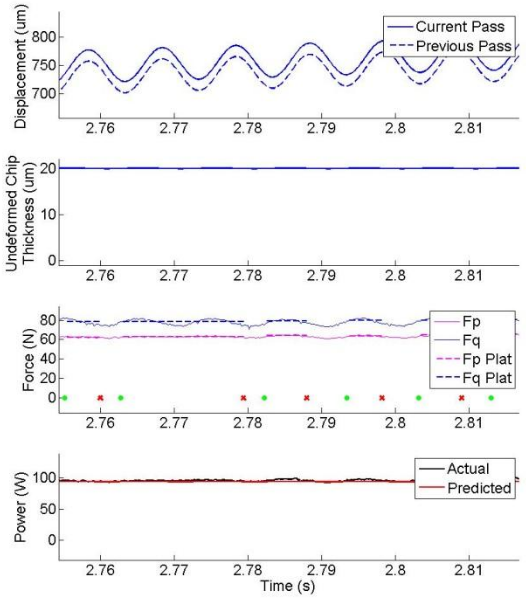


Figure 6.27b: Close-up of numerical results for Ti3Al2.5V cut at s=0.020 mm/rev, fm=100Hz, K=60V using lathe.

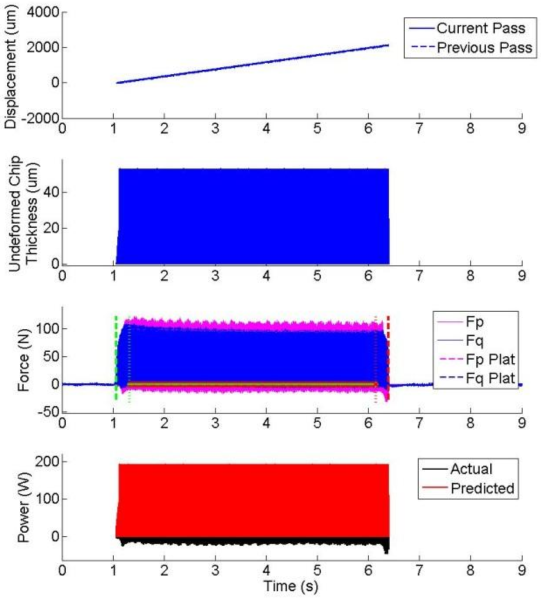


Figure 6.28a: Numerical analysis results for Ti3Al2.5V cut at s=0.020 mm/rev, fm=110Hz, K=60V using lathe.

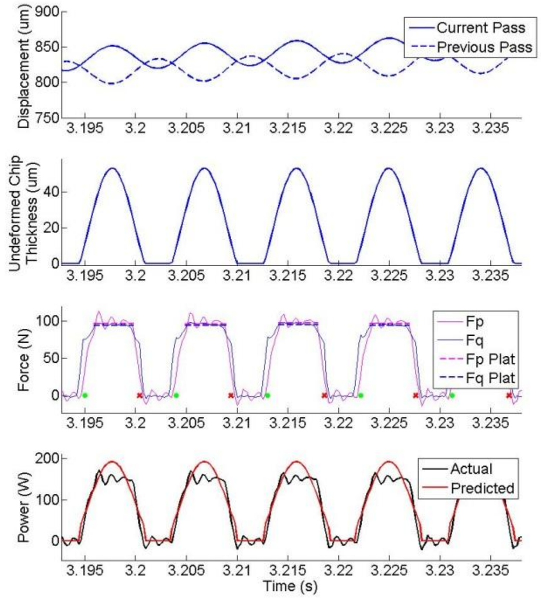


Figure 6.28b: Close-up of numerical results for Ti3Al2.5V cut at s=0.020 mm/rev, fm=110Hz, K=60V using lathe.

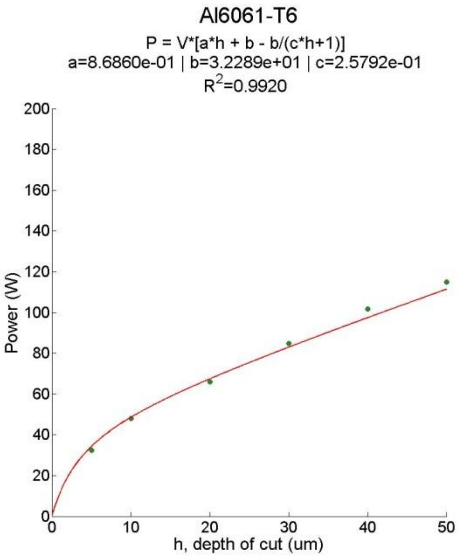


Figure 6.29: P(X) trace for AA6061-T6 where X denotes depth of cut.

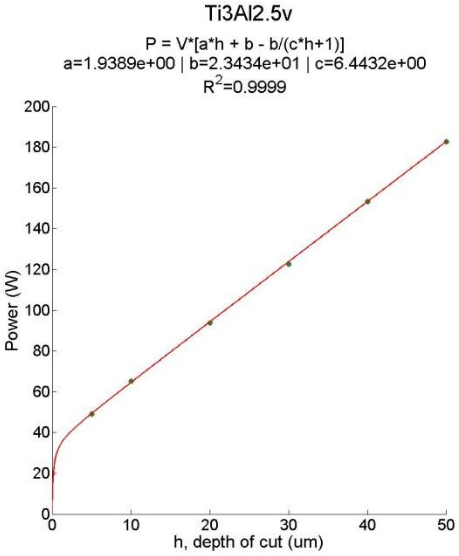


Figure 6.30: P(X) trace for Ti3Al2.5V where X denotes depth of cut.

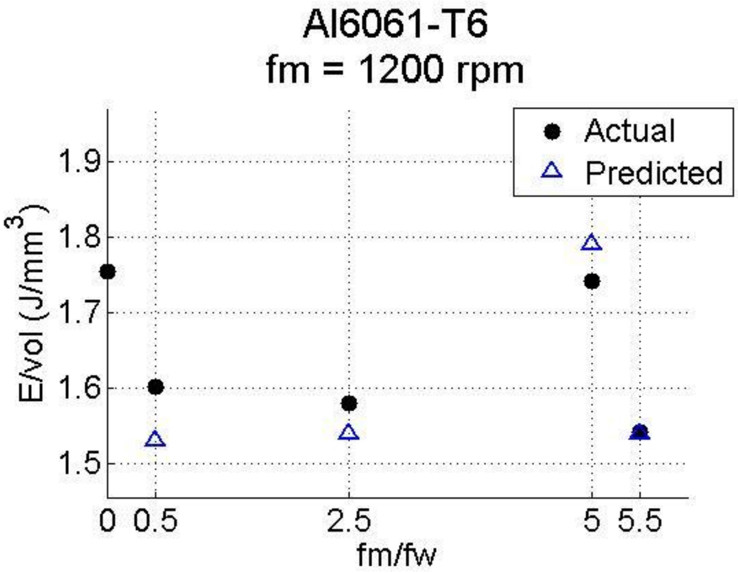


Figure 6.31: Actual and predicted value of specific energy for AA6061-T6 at various fm/fw ratios.

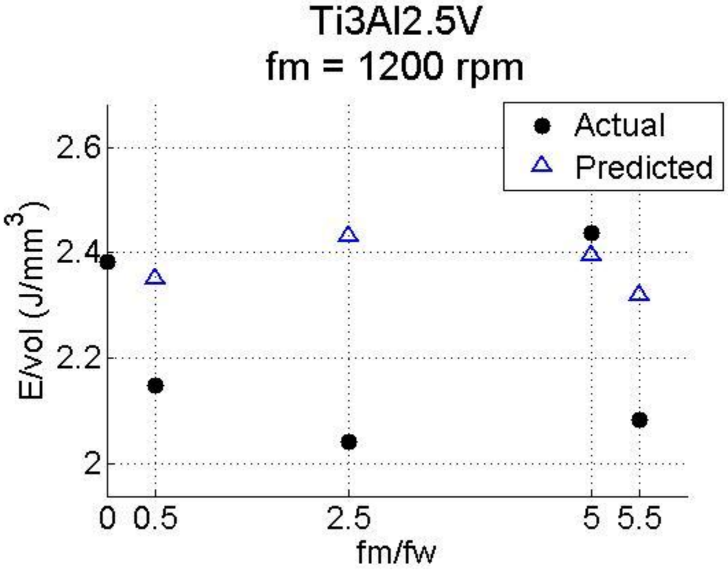
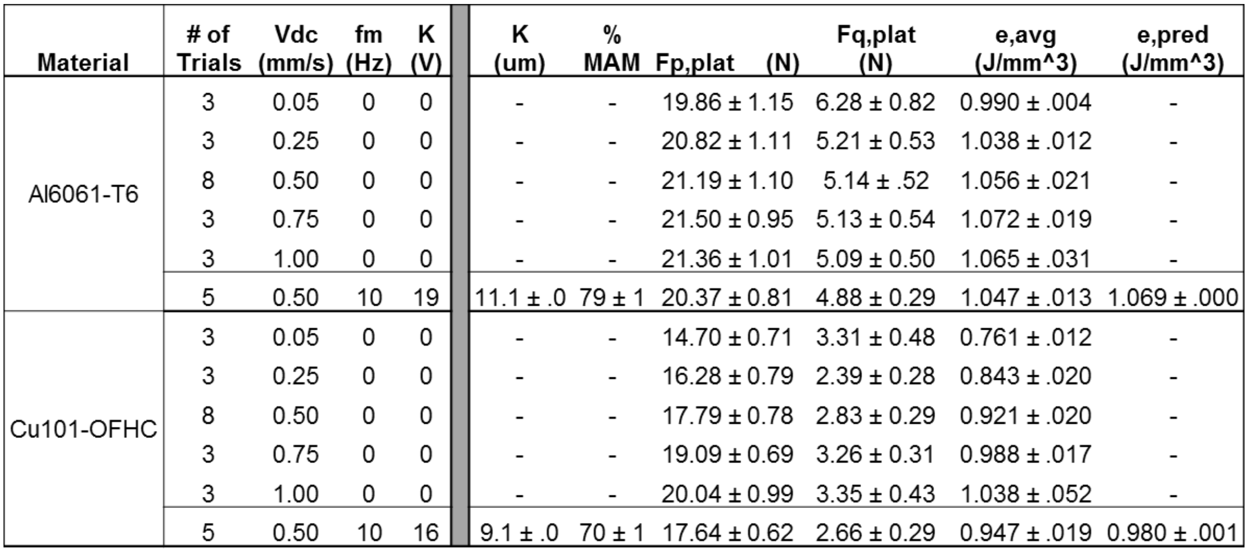


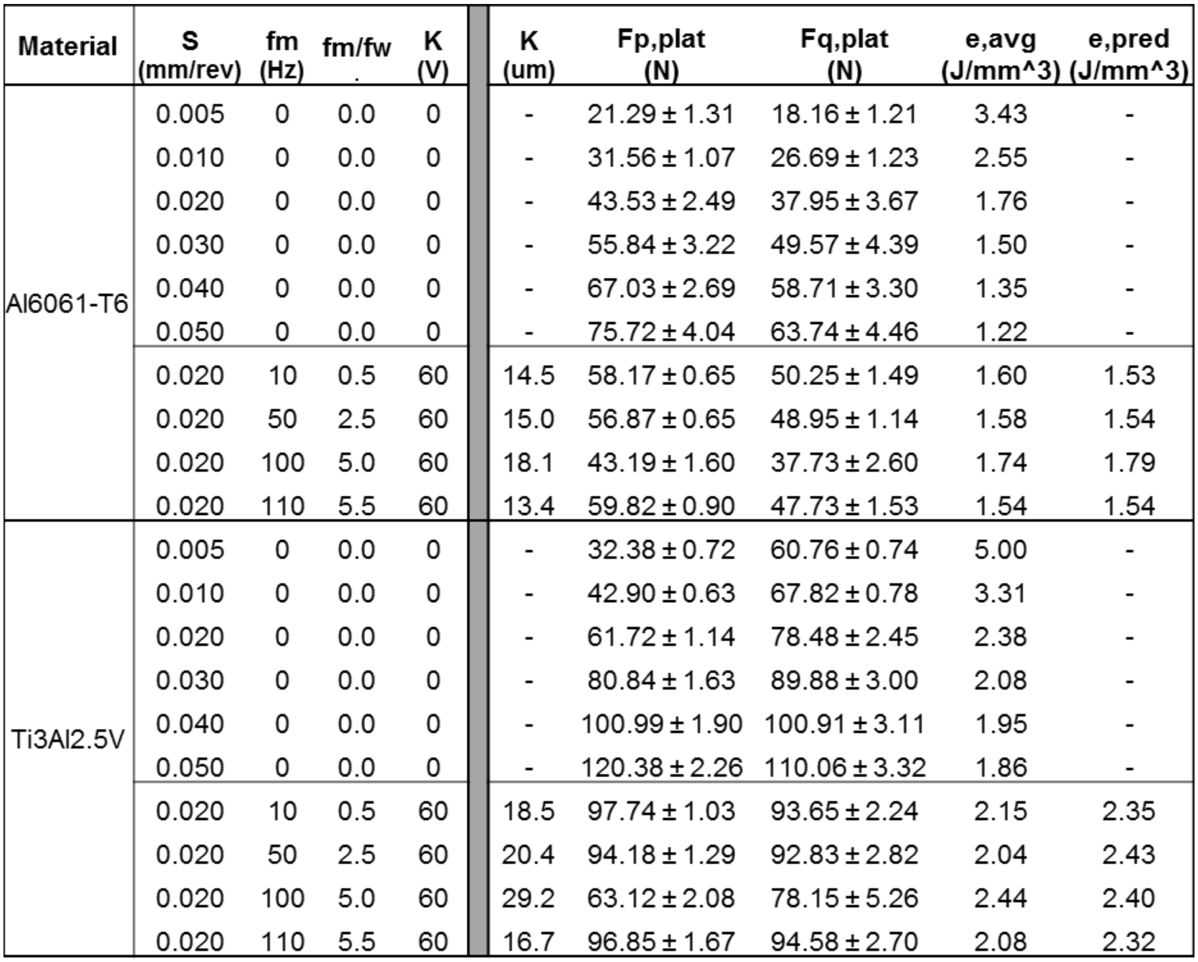
Figure 6.32: Actual and predicted value of specific energy for Ti3Al2.5V at various fm/fw ratios.

Table 6.1: List of trials performed using planer (velocity-direction MAM) and their results. All cuts were made at h0=20um. Note that e,avg denotes the average specific energy for the empirical data and e,pred that resulting from integrating the predicted instantaneous power values.



Material	# of Trials	Vdc (mm/s)	fm (Hz)	K (V)	K (um)	% MAM	Fp,plat (N)	Fq,plat (N)	e,avg (J/mm^3)	e,pred (J/mm^3)
Al6061-T6	3	0.05	0	0	-	-	19.86 ± 1.15	6.28 ± 0.82	0.990 ± .004	-
	3	0.25	0	0	-	-	20.82 ± 1.11	5.21 ± 0.53	1.038 ± .012	-
	8	0.50	0	0	-	-	21.19 ± 1.10	5.14 ± .52	1.056 ± .021	-
	3	0.75	0	0	-	-	21.50 ± 0.95	5.13 ± 0.54	1.072 ± .019	-
	3	1.00	0	0	-	-	21.36 ± 1.01	5.09 ± 0.50	1.065 ± .031	-
	5	0.50	10	19	11.1 ± .0	79 ± 1	20.37 ± 0.81	4.88 ± 0.29	1.047 ± .013	1.069 ± .000
Cu101-OFHC	3	0.05	0	0	-	-	14.70 ± 0.71	3.31 ± 0.48	0.761 ± .012	-
	3	0.25	0	0	-	-	16.28 ± 0.79	2.39 ± 0.28	0.843 ± .020	-
	8	0.50	0	0	-	-	17.79 ± 0.78	2.83 ± 0.29	0.921 ± .020	-
	3	0.75	0	0	-	-	19.09 ± 0.69	3.26 ± 0.31	0.988 ± .017	-
	3	1.00	0	0	-	-	20.04 ± 0.99	3.35 ± 0.43	1.038 ± .052	-
	5	0.50	10	16	9.1 ± .0	70 ± 1	17.64 ± 0.62	2.66 ± 0.29	0.947 ± .019	0.980 ± .001

Table 6.2: List of trials performed using lathe (feed-direction MAM) and their results. All cuts were made at fw=1200 rpm.



Material	S	fm	fm/fw	K	K	Fp,plat	Fq,plat	e,avg	e,pred
	(mm/rev)	(Hz)	.	(V)	(um)	(N)	(N)	(J/mm^3)	(J/mm^3)
Al6061-T6	0.005	0	0.0	0	-	21.29 ± 1.31	18.16 ± 1.21	3.43	-
	0.010	0	0.0	0	-	31.56 ± 1.07	26.69 ± 1.23	2.55	-
	0.020	0	0.0	0	-	43.53 ± 2.49	37.95 ± 3.67	1.76	-
	0.030	0	0.0	0	-	55.84 ± 3.22	49.57 ± 4.39	1.50	-
	0.040	0	0.0	0	-	67.03 ± 2.69	58.71 ± 3.30	1.35	-
	0.050	0	0.0	0	-	75.72 ± 4.04	63.74 ± 4.46	1.22	-
	0.020	10	0.5	60	14.5	58.17 ± 0.65	50.25 ± 1.49	1.60	1.53
	0.020	50	2.5	60	15.0	56.87 ± 0.65	48.95 ± 1.14	1.58	1.54
	0.020	100	5.0	60	18.1	43.19 ± 1.60	37.73 ± 2.60	1.74	1.79
	0.020	110	5.5	60	13.4	59.82 ± 0.90	47.73 ± 1.53	1.54	1.54
Ti3Al2.5V	0.005	0	0.0	0	-	32.38 ± 0.72	60.76 ± 0.74	5.00	-
	0.010	0	0.0	0	-	42.90 ± 0.63	67.82 ± 0.78	3.31	-
	0.020	0	0.0	0	-	61.72 ± 1.14	78.48 ± 2.45	2.38	-
	0.030	0	0.0	0	-	80.84 ± 1.63	89.88 ± 3.00	2.08	-
	0.040	0	0.0	0	-	100.99 ± 1.90	100.91 ± 3.11	1.95	-
	0.050	0	0.0	0	-	120.38 ± 2.26	110.06 ± 3.32	1.86	-
	0.020	10	0.5	60	18.5	97.74 ± 1.03	93.65 ± 2.24	2.15	2.35
	0.020	50	2.5	60	20.4	94.18 ± 1.29	92.83 ± 2.82	2.04	2.43
	0.020	100	5.0	60	29.2	63.12 ± 2.08	78.15 ± 5.26	2.44	2.40
	0.020	110	5.5	60	16.7	96.85 ± 1.67	94.58 ± 2.70	2.08	2.32

Chapter 7: Discussion

The present work involved characterization of load response, power dissipation and energy expended as a function of controllable process parameters in conventional machining as well as in two configurations of MAM: that is, velocity-direction and feed-direction. A fundamental characteristic of these modulation configurations is that instantaneous machining input variables of cutting velocity and undeformed chip thickness are modulated over a controllable range of values. From the results, it was clear that parameter modulation was an important factor in determining the nature of the measured response for most of the modulation-assisted machining cases investigated. In this regard, the effects of these parameters on load response in conventional machining were found to explain well the nature of the loading behavior in modulation-assisted machining. This section of the thesis serves as a discussion of these results and introduces potential explanations of this behavior.

7.1 Velocity-direction MAM

Conventional machining experiments at different velocities were used to model the correlation between cutting velocity and load response (e.g., force, power). The general form of the fit used to model the force trace was $\mathrm{Fp} = \mathrm{A}\ln[\mathrm{X}] + \mathrm{B}$ and that used to model power dissipation was $P = A(X * 10^{-3})\ln[X] + B(X * 10^{-3})$, where X is cutting velocity in mm/s and A and B are constants. Recall that these generalized expressions were elucidated from cutting experiments in soft steels at low speeds [11], and were used here to describe the materials of the present study. From the results, it was clear from the high R^2 values (e.g., R^2>0.998 for both materials' P(X) relationships) of the fits that these general relationships were adequate in modeling the effects of machining variables on conventional machining behavior. These fitted power dissipation relationships in conventional machining were then used to predict instantaneous power dissipation in modulation-assisted machining, wherein values of the independent variable X (where X denotes cutting velocity here) varied over time. Figures 6.7b and 6.8b show that the predicted power values track the actual power data quite well with small over-prediction (less than 3.5%) at peak values. The ability to predict loading behavior in modulation-assisted machining in this manner indicates that modulation within the parameter space investigated (e.g., low frequency) likely does not modify the underlying mechanics of deformation that occurs in conventional machining.

With regard to energy consumption, it was found that no statistically significant difference existed between the specific cutting energy of conventional machining and that of modulation-assisted machining of AA6061-T6. On the other hand, conventional machining of OFE Cu was found to require less specific energy than velocity-direction MAM. (It should be noted that although this difference is statistically significant, it is still marginal in context of the overall amount consumed.) These results can be explained by consideration of the nature of the P(X) relationships that describe effects in conventional machining of these materials. It was shown earlier that the shape of a material's P(X) curve may have a direct influence on the energy expended in machining when X is time-varying. Specifically, for materials with convex forms of the P(X) relationship, modulation-assisted machining is expected to require more energy than its conventional counterpart. The converse is true for concave forms and the terms are mathematically equivalent for linear forms of P(X). Figs. 6.9 and 6.10 show the P(X) relationships for both materials; the curvature of the copper P(X) trace is more than three times greater that of the aluminum alloy, though both appear fairly linear overall. Thus, it is not surprising that the energy differences between conventional and modulated cutting are relatively small (<3%) for both materials.

The physical basis of the slightly more pronounced curvature in the OFE copper P(X) behavior can come from an understanding of strain rate effects in deformation. For many strain-hardening materials, increases in strain rate result in increased flow stress [2] and the degree to which a material is affected by strain rate is material-dependent [35-38]. This phenomenon is manifested in a measure commonly referred to as strain rate sensitivity (m), which is defined as $m=\ln(\sigma_2/\sigma_1)/\ln(\dot{\varepsilon}_2/\dot{\varepsilon}_1)$, where σ is the true stress, EDOT is the strain rate, and subscripts 1 and 2 refer to the specimens deformed at the lower and higher of the two strain rates, respectively [35,37]. The parameter m is usually evaluated at constant strain and temperature. Two common methods exist for evaluating strain rate sensitivity. In the first, two samples of the same material are deformed under tensile load at different strain rates ($\dot{\varepsilon}_1$ and $\dot{\varepsilon}_2$) thus generating two curves. In the second, a single sample is deformed at a specified strain rate ($\dot{\varepsilon}_1$) to some limit and then the strain rate is rapidly changed to the higher rate ($\dot{\varepsilon}_2$). This is often referred to as a jump test [37].

The first of these methods was performed for both AA6061-T6 and OFE copper at room temperature to better understand rate-dependent response of both materials. Cylindrical tensile testing specimens were made using a CNC lathe with consideration to ASTM E8. A drawing of these specimens is provided in Figure 7.1, modified so as to effect a 1% (.003") increase in diameter at either end of the narrow section—a common allowance made in order to provide some control over the necking/fracture location. Two specimens for each material were tested in tension to failure using a uniaxial testing machine (Instron 4206) at strain rates of $3.3\text{x}10^{-4}$ s^{-1} and $3.3\text{x}10^{-3}$ s^{-1} and elongation was measured with an extensometer (Instron 2630-037, 2" gauge length). It should be noted that these strain rates are expected to be several orders of magnitude lower than those in the present study. Loading methods capable of assessing strain rate sensitivity at higher strain rates are possible using the Hopkinson bar technique [36,38,39].

Engineering stress and strain measured during testing were converted to true stress – true strain up to necking according to $\sigma_T = \sigma_E*(1+\varepsilon_E)$ and $\varepsilon_T = \ln(1+\varepsilon_E)$ [40] and results from these experiments are provided in Figs. 7.2-7.3. Strain rate sensitivity for each material was calculated as a function of strain in the plastic regime. From Table 7.1, OFE Cu exhibited strain rate sensitivity approximately an order of magnitude higher (m = 1.07e-2) than that of AA6061-T6 (m = 9.17e-4) at their respective yield points. The values for m determined for copper are consistent with those found in literature ($m=1.4\text{x}10^{-2}$ at $\varepsilon=.25$, $\dot{\varepsilon}_1=.00014s^{-1}$, $\dot{\varepsilon}_2=.015s^{-1}$) [36]. Although no value for m could be found in literature for the AA6061-T6, the order of magnitude in Table 7.1 is similar to that of AA2024 at room temperature [37]. The relatively higher strain rate sensitivity indicates that the copper is more susceptible to non-linear $P(\dot{\varepsilon})$ response than is the aluminum alloy. This is consistent with the greater degree of convexity exhibited in the P(X) trace for copper, where X is the cutting velocity. Regardless of the material or degree of P(X) convexity, the dependent relationship between cutting speed and power in conventional machining vis-à-vis P(X) characterization was found to be an adequate model to describe the loading response in modulation-assisted machining. In this regard, the underlying deformation response in velocity-direction modulation under dry conditions is not significantly different than that of conventional machining.

Finally, it should be noted that the effect of system compliance can be seen in the slight discrepancies between the force and displacement data. However, the effects of compliance on the analysis method used in the present study are insignificant since they merely act to narrow the effective performance range of the system.

7.2 Feed-direction MAM

Similar to the method used in the velocity-direction modulation case, conventional machining experiments at different undeformed chip thicknesses were used to model the correlation between undeformed chip thickness and load response (e.g., power, force). The general form of the fit used to model the force trace was $Fp = AX + B - \frac{B}{CX+1}$ and that used to model power dissipation was $P = V_{DC}\left[AX + B - \frac{B}{CX+1}\right]$ where X denotes undeformed chip thickness (units of um), P has units of Watts, V_{DC} units of m/s, and A, B, C are constants. These generalized expressions were elucidated from a study performed on plain carbon steel [10], and were used here to describe the materials of the present study. As was the case in velocity-direction modulation, high R^2 values (e.g., $R^2 > 0.99$ for the P(X) relationship in both materials) indicated that these general relationships were adequate in modeling the effects of undeformed chip thickness on conventional machining behavior. These relationships were then applied to fit the power dissipated in modulation-assisted machining. For both materials, there is a good correlation between the actual and predicted power traces except at the peaks in the power trace when fm/fw was an odd integer multiple of 1/2 (e.g., fm = 10, 50, 110 Hz), as is seen in Figs. 6.21,6.22,6.24 and 6.25,6.26,6.28. This can be compared to the relatively good prediction when fm/fw was an integer multiple of 1/2 (e.g., fm = 100 Hz), as is seen in Figs. 6.23 and 6.27. The primary difference in both scenarios is that the undeformed chip thickness is time varying in the former while it is constant in the latter. In this regard, feed modulation with a constant chip thickness more closely resembles a conventional machining configuration. The unexpected behavior at the peaks in the power trace for cases of dynamic undeformed chip thickness suggests that an additional mechanism is present in these materials when they are cut using feed-direction modulation. The effects of such a mechanism on loading response can be characterized by the generation of a dynamic P(X) trace, examples of which can be seen in Figures 7.4-7.5. It is suggested that such a phenomenon could be due to the assumption that undeformed chip thickness is the only significant dynamic processing parameter in feed-direction MAM. Such is not entirely true however, as rake angle and cutting speed are also technically dynamic here. Thus, further studies on this subject may make good use of a more robust definition for the parameter X that would somehow combine the respective dynamic natures of undeformed chip thickness, rake angle, and cutting speed.

With regards to specific energy consumption, the amount consumed in conventional cutting for each material dropped significantly as undeformed chip thickness increased from low to moderate values before leveling off somewhat (see Fig. 7.6). This trend is in agreement with literature already published regarding the size effect [4]. Furthermore, AA6061-T6 and Ti3Al2.5V showed similar trends in that the specific energy in modulation-assisted machining when the fm/fw ratio is an odd integer multiple of 1/2 (e.g., fm = 10, 50, 110 Hz) was lower than when the ratio is either zero (e.g., fm = 0) or when it is an even integer multiple of 1/2 (e.g., fm = 100 Hz). As was true for the velocity-direction MAM cases, the energy expenditure in feed-direction MAM can also be explained by consideration of the nature of the P(X) relationships that describe effects in conventional machining of these materials. Figs. 6.29 and 6.30 show the P(X) relationships for both materials: both are seen to exhibit some degree of concavity in their traces. An unexpected outcome in this regard is that both materials showed similar energy reductions with the use of modulation, despite the aluminum alloy exhibiting significantly more concavity than the titanium alloy ($\partial^2P/\partial X^2 = -2.8\text{x}10^{-2}$ in aluminum versus $-1.3\text{x}10^{-3}$ in titanium). Regardless, it was shown earlier that in situations wherein P(X) is concave, modulation-assisted machining is expected to have lower energy expenditure than conventional machining due to the shape of the input power distribution.

The physical basis of curvature in the P(X) trace for both materials can come from an understanding of the effects of undeformed chip thickness and size scale in deformation. In this regard, the non-linear response of materials with decreasing sample size has been well documented for various types of loading configurations and is commonly referred to as the “size effect” when applied to the field of metal cutting [1,4,10,12]. The size effect phenomenon in machining has been explained by Shaw through consideration of intrinsic defect density [15]. The premise of his theory rests in the notion that the distribution of defects in the material affects flow stress in a non-linear manner with changes in depth of cut. Specifically, for smaller depths of cut, a fewer number of defects pass through the PDZ that would ultimately give rise to slip plane formation. A schematic of this method as applied to orthogonal cutting can be seen in Figure 7.7 [15]. This schematic is also used in Shaw’s later publication which serves as a literature review on the size effect [12].

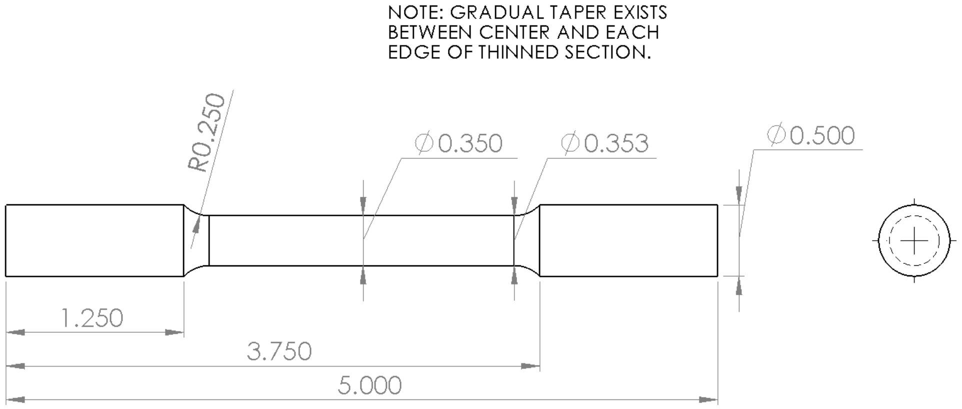


Figure 7.1: Drawing of round tensile test specimens fabricated using a CNC lathe. A gradual taper is enforced on the narrow section such that either end has a diameter of 0.353" in order to control the location of necking/fracture.

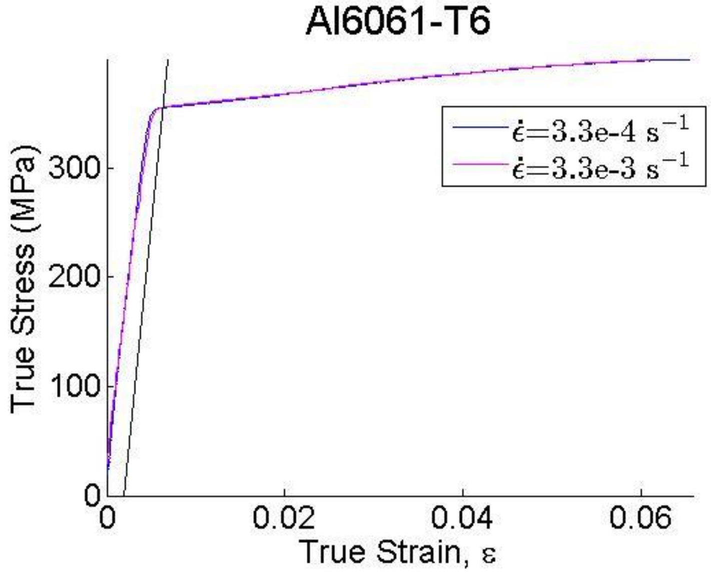


Figure 7.2: True stress – true strain curves for AA6061-T6 deformed at different strain rates, used to determine the strain rate sensitivity parameter, m.

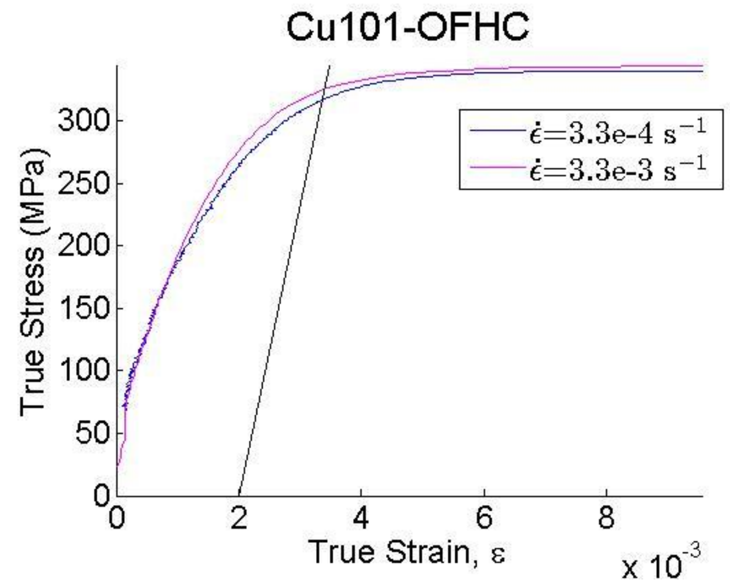


Figure 7.3: True stress – true strain curves for OFE Copper deformed at different strain rates, used to determine the strain rate sensitivity parameter, m.

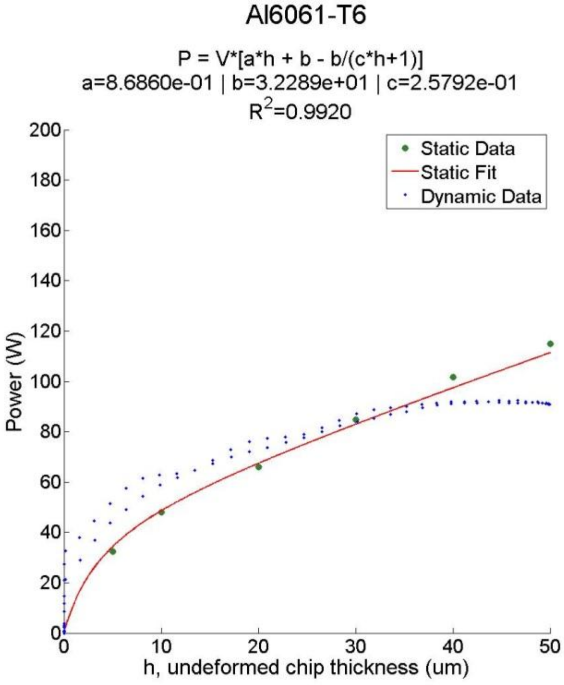


Figure 7.4: P(X) trace for AA6061-T6 demonstrating difference between the static data and dynamic data collected at fm=50Hz.

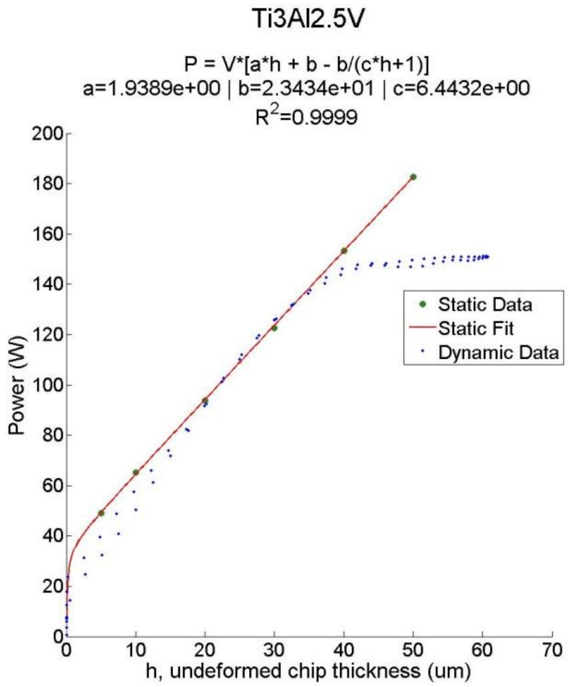


Figure 7.5: P(X) trace for Ti3Al2.5V demonstrating difference between the static data and dynamic data collected at fm=50Hz.

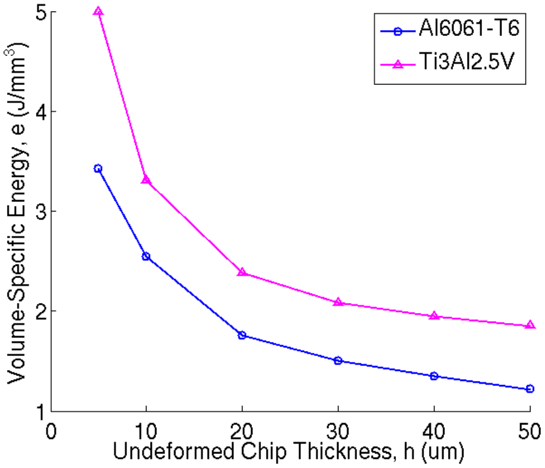


Figure 7.6: Specific energy versus undeformed chip thickness for AA6061-T6 and Ti3Al2.5V.

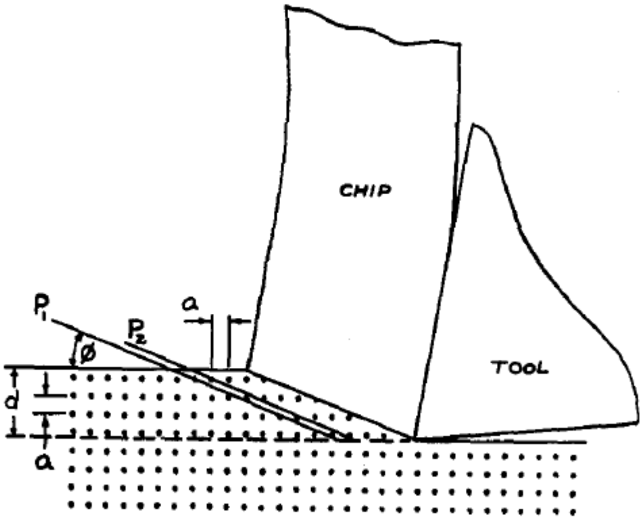
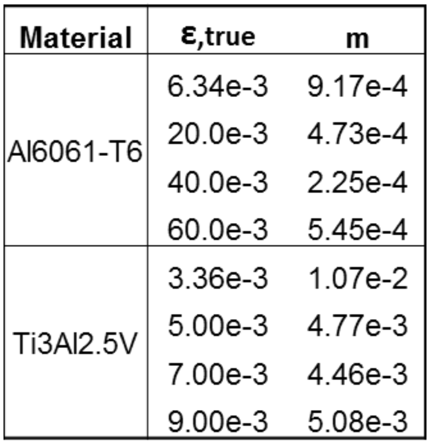


Figure 7.7: Figure taken from Shaw [15] used to conceptualize the presence of defects within a material and their effect on slip-plane formation. Note here that a denotes defect spacing, d the undeformed chip thickness, P an individual slip plane, and Φ the shear angle.

Table 7.1: Values for strain rate sensitivity determined empirically for AA6061-T6 and OFE copper as a function of true strain.



Material	ε,true	m
Al6061-T6	6.34e-3	9.17e-4
	20.0e-3	4.73e-4
	40.0e-3	2.25e-4
	60.0e-3	5.45e-4
Ti3Al2.5V	3.36e-3	1.07e-2
	5.00e-3	4.77e-3
	7.00e-3	4.46e-3
	9.00e-3	5.08e-3

Chapter 8: Conclusions and Future Work

The present study included a detailed characterization of loading response (e.g., force, power, energy) in conventional machining and modulation-assisted machining in the velocity and feed directions. It was shown that material response in conventional machining provided the basis for an adequate model to describe the loading response in modulation-assisted machining. In this regard, it was found that the relationship between power dissipation and a modulated machining parameter (e.g., velocity or undeformed chip thickness), P(X), could be determined from conventional machining experiments and applied to predict power dissipation in modulation-assisted machining. This was true for every material tested in velocity-direction modulation (AA6061-T6 and OFE Cu) and in feed-direction modulation (AA6061-T6 and Ti3Al2.5V). The shape of the P(X) relationship was shown to be important, as convexity in the trace indicated that modulated cutting would require more energy than conventional machining. The opposite was shown to be true for concave P(X) relationships and the terms were found to be equal in the case of linear P(X) relationships. This comparative reasoning is, of course, only valid if the underlying thermo-mechanical response of the system with regard to static and dynamic variations in processing parameters is constant. The analysis of power consumption and specific energy performance parameters indicated this assumption was valid for velocity-direction modulation of both materials and for feed-direction modulation of the aluminum alloy.

The predictive analysis correctly indicated that specific cutting energy in velocity-direction modulation of AA6061-T6 was approximately the same for conventional cutting. Further, it also was able to properly determine that, for OFE Cu, velocity-direction modulation required slightly more energy than conventional machining. The convexity of the P(X) relationship for the aluminum alloy was approximately three times greater than that of copper and was linked to this behavior. Tensile testing experiments indicated that copper has a strain rate sensitivity an order of magnitude greater than that of the aluminum alloy at their respective yield points, which is consistent with the above findings regarding energy consumption and P(X) convexity. With regard to feed-direction modulation, the specific energies required to cut both AA6061-T6 and Ti3Al2.5V using modulation were found to be significantly less than those required by conventional machining. Although this was an expected result due to the concavity of each material’s respective static P(X) trace, the predicted power at peak values of instantaneous undeformed chip thickness was found to overestimate the actual power requirement, particularly for the titanium alloy. This was evidenced through a dynamic P(X) trace which deviated from the static P(X) trace at large values of undeformed chip thickness (h_0>40 um). Thus, although it appears that the size effect phenomenon is present in feed-direction MAM, the mechanical response of the system to dynamic changes in this processing parameter is different than that observed for static changes in undeformed chip thickness. It is suggested that a further study be carried out which incorporates the combined effects of undeformed chip thickness, cutting speed, and rake angle in the variable X, since each of these is dynamic in nature for feed-direction MAM.

Finally, it is suggested that a future study make use of PDZ thickness as an intermediate variable relating the processing parameters of cutting speed and undeformed chip thickness to specific energy. This would build upon the work performed by Kececioglu who believed that both an increase in cutting speed and a decrease in undeformed chip thickness reduced the size of the PDZ and thus increased the flow stress of the material [2,14]. Such an approach could lead to a better understanding of the physics underlying how processing parameters affect energy consumption and whether their effects can be explained in terms of some common variable.

References

[1] Shaw, Milton, 1984, *Metal Cutting Principles*, Oxford University Press, New York, NY.

[2] Kececioglu, D., 1958, “Shear-Strain Rate in Metal Cutting and its Effects on Shear-Flow Stress,” Trans. ASME, **80**(1), pp. 158-167.

[3] Moscoso, W., Olgun, E., Compton, W. D., and Chandrasekar, S., 2005, “Effect of Low-Frequency Modulation on Lubrication of Chip-Tool Interface in Machining,” Trans. ASME, J. Tribol., **127**(1), pp. 238-244.

[4] Joshi, S. S., and Melkote, S. N., 2004, “An Explanation for the Size-Effect in Machining Using Strain Gradient Plasticity,” Trans. ASME, J. Manuf. Sci. Eng., **126**(4), pp. 679-684.

[5] Merchant, M. E., 1945, “Mechanics of the Metal Cutting Process,” J. Appl. Phys. **16**, pp. 167-275, 318-324.

[6] Piispanen, V., 1948, “Theory of Formation of Metal Chips,” J. Appl. Phys., **19**(10), pp. 876-881.

[7] Mann, J. B., Guo, Y., Saldana, C., Compton, W. D., and Chandrasekar, S., 2011, “Enhancing Material Removal Processes Using Modulation-Assisted Machining,” Tribol. Int., **44**(10), pp. 1225-1235.

[8] Doyle, E. D., Horne, J. G., and Tabor, D., 1979, “Frictional Interactions Between Chip and Rake Face in Continuous Chip Formation,” Proc. R. Soc. Lond. A, Math. Phys. Sci., **366**(1725), pp. 173-183.

[9] Deyuan, Z., and Lijiang, W., 1998, “Investigation of Chip in Vibration Drilling,” Int. J. Mach. Tools. Manuf., **38**(3), pp. 165-176.

[10] Arsecularatne, J. A., 1997, “On Tool-Chip Interface Stress Distributions, Ploughing Force and Size Effect in Machining,” Int. J. Mach. Tools Manuf., **37**(7), pp. 885-899.

[11] Maan, N., and Broese Van Groenou, A., 1977, “Low Speed Scratch Experiments on Steels,” Wear, **42**(2), pp. 365-390.

[12] Shaw, M. C., 2003, “The Size Effect in Metal Cutting,” Sadhana, **28**, pp. 875-896.

[13] Dinesh, D., Swaminathan, S., Chandrasekar, S., and Farris, T. N., 2001, “An Intrinsic Size-Effect in Machining due to the Strain Gradient,” Am. Soc. Mech. Eng. Manuf. Eng. Div., **12**, pp. 197-204.

[14] Kececioglu, D., 1960, “Shear-Zone Size, Compressive Stress, and Shear Strain in Metal-Cutting and their Effects on Mean Shear-Flow Stress,” Trans. of ASME J. Eng. Industry, **82**(1), pp. 79-86.

[15] Shaw, M. C., 1950, “A Quantized Theory of Strain Hardening as Applied to the Cutting of Metals,” J. Appl. Phys., **21**, pp. 599-606.

[16] Toews, H. G. III, Compton, W. D., and Chandrasekar, S., 1998, “A Study of the Influence of Superimposed Low-Frequency Modulation on the Drilling Process,” Precis. Eng., **22**(1), pp. 1-9.

[17] Chhabra, P. N., Ackroyd, B., Compton, W. D., and Chandrasekar, S., 2002, “Low-Frequency Modulation-Assisted Drilling Using Linear Drives,” Proc. Inst. Mech. Eng. Part B J. Eng. Manuf., **216**(3), pp. 321-330.

[18] Mann, J. B., Saldana, C., Chandrasekar, S., Compton, W. D., and Trumble, K. P., 2007, “Metal Particulate Production by Modulation-Assisted Machining,” Scr. Mater., **57**(10), pp. 909-912.

[19] Mann, J. B., Saldana, C., Moscoso, W., Compton, W. D., and Chandrasekar, S., 2009, “Effects of Controlled Modulation on Interface Tribology and Deformation in Machining,” Tribol. Lett., **35**(3), pp. 211-217.

[20] Saldana, C., Swaminathan, S., Brown, T. L., Moscoso, W., Mann, J. B., Compton, W. D., and Chandrasekar, S., 2010, “Unusual Applications of Machining: Controlled Nanostructuring of Materials and Surfaces,” Trans. ASME J. Manuf. Sci. Eng., **132**(3), pp. 0309081-03090812.

[21] Mann, J. B., Guo, Y., Saldana, C., Yeung, H., Compton, W. D., and Chandrasekar, S., 2011, “Modulation-Assisted Machining: A New Paradigm in Material Removal Processes,” Adv. Mater. Res., **223**, pp. 514-522.

[22] Brehl, D. E., and Dow, T. A., 2007, “Review of Vibration-Assisted Machining,” Precis. Eng., **32**(3), pp. 153-172.

[23] Joshi, R. S., and Singh, H., 2011, “Piezoelectric Transducer Based Devices for Development of a Sustainable Machining System – A Review,” *Proc. 2011 Joint IEEE International Symposium on Applications of Ferroelectrics/International Symposium on Piezoresponse Force Microscopy & Nanoscale Phenomena in Polar Materials*, IEEE, Piscataway, NJ.

[24] Shamoto, E., and Moriwaki, T., 1994, “Study on Elliptical Vibration Cutting,” CIRP Ann, **43**(1), pp. 35-38.

[25] Dasch, J. M., Ang, C. C., Wong, C. A., Cheng, Y. T., Weiner, A. M., Lev, L. C., and Konca, E., 2004, “A Comparison of Five Categories of Carbon-Based Tool Coatings for Dry Drilling of Aluminum,” Surf. Coat. Technol., **200**(9), pp. 2970-2977.

[26] Litvinov, L. P., 1990, “Vibration-Assisted Drilling of Deep Holes,” Vestnik Mashinostroeniya, **70**(5), pp. 22-24.

[27] Maan, J. B., 2010, “Modulation-Assisted Machining,” Ph.D. thesis, Purdue University.

[28] Takeyama, H, and Kato, S., 1991, “Burrless Drilling by Means of Ultrasonic Vibration,” CIRP Ann., **40**(1), pp. 83-86.

[29] Astashev, V. K., 1992, “Effect of Ultrasonic Vibration of a Single-Point Tool on the Process of Cutting,” J. Mach. Manuf. Reliab., **5**(3), pp. 65-70.

[30] Astashev, V. K., and Babitsky, V. I., 1998, “Ultrasonic Cutting as a Nonlinear (Vibro-Impact) Process,” Ultrasonics, **36**(1-5), pp. 89-96.

[31] Langenecker, B., 1966, “Effects of Ultrasound on Deformation Characteristics of Metals,” IEEE Trans. Sonics. Ultrason. **SU-13**(1), pp. 1-8.

[32] Rosa, P. A. R., Kolednik, O., Martins, P. A. F., and Atkins, A. G., 2007, “The Transient Beginning to Machining and the Transition to Steady-State Cutting,” Int. J. Mach. Tools Manuf., **47**(12-13), pp. 1904-1915.

[33] Jensen, J. L. V. W., 1905, “Sur les Fonctions Convexes et les Inégalités Entre les Valeurs Moyennes,” Acta Mathematica, **90**.

[34] Marusich, T. D., 2001, “Effects of Friction and Cutting Speed on Cutting Force,” Am. Soc. Mech. Eng. Manuf. Eng. Div., **12**, pp. 115-123.

[35] Nieh, T. G., and Wadsworth, J., 1990, “Superplastic Ceramics,” Annual review of materials science, **32**, pp.117-140.

[36] Follansbee, P. S., and Kocks, U. F., 1988, “A Constitutive Description of the Deformation of Copper Based on the Use of the Mechanical Threshold Stress as an Internal State Variable,” Acta Matall., **36**(1), pp. 81-93.

[37] Hosford, W. F., 2011, *Metal Forming Mechanics and Metallurgy*, 4th ed., Cambridge University Press, Chap. 5.

[38] Glenn, T., and Bradley, W., 1973, “The Origin of Strain-Rate Sensitivity in OFHC Copper,” Metall. Trans. A, Phys. Metal. Mater. Sci., **4**(10), pp. 2343-2348.

[39] Nicholas, T., 1981, “Tensile Testing of Materials at High Rates of Strain,” Exp. Mech., **21**(5), pp. 177-185.

[40] Callister, W. D., 2007, *Materials Science and Engineering: An Introduction*, 7th ed., John Wiley & Sons, Inc., New York, NY, Chap. 6.

Appendix A: Photos of planer setup

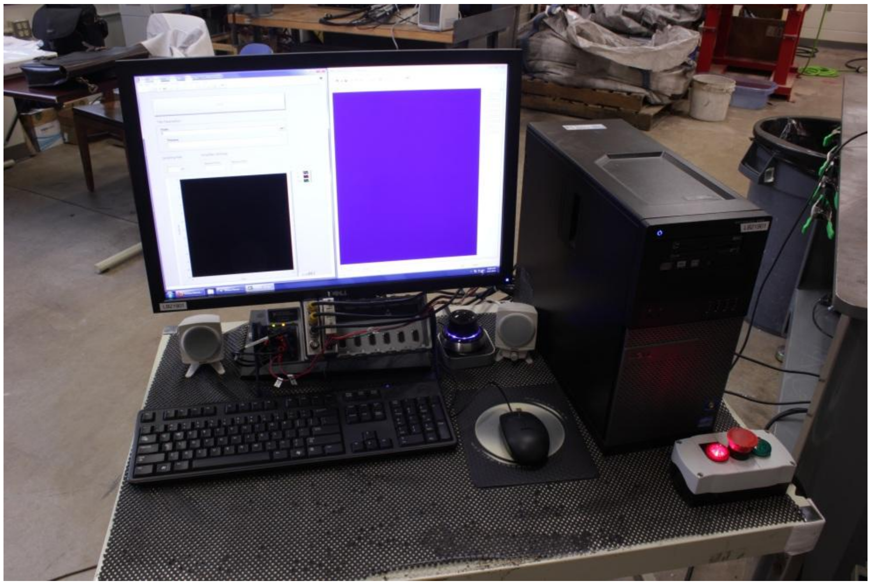


Figure A.1: Desktop PC unit and NI data acquisition assembly.

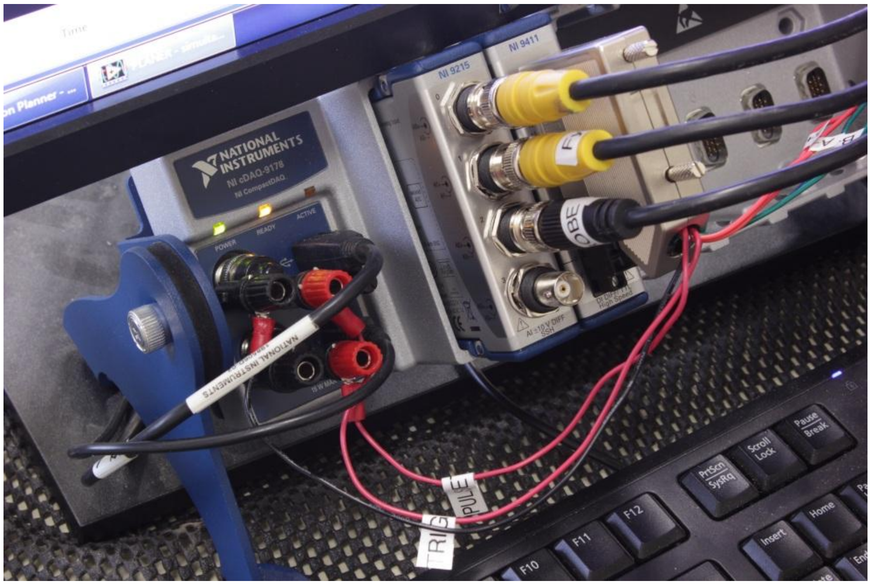


Figure A.2: Close-up of NI data acquisition assembly. Note: Chassis PFI0 outputs trigger and chassis PFI1 outputs pulsetrain (dictates sampling rate), both of which are collected by NI-9411. The NI-9411 also receives differentially encoded digital data from the rotary encoder (D_{DC}). Channels 0/1/2 on the NI-9215 receive -10 to +10V analog signals carrying data regarding Fp/Fq/D_{AC}, respectively.

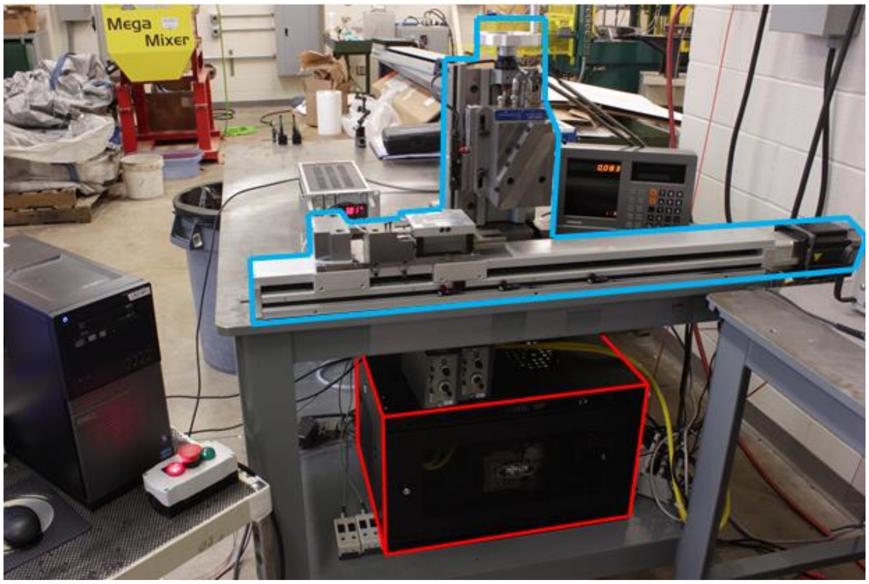


Figure A.3: Table holding planer assembly mechanical components (outlined in blue, top) and controller housing (outlined in red, bottom)

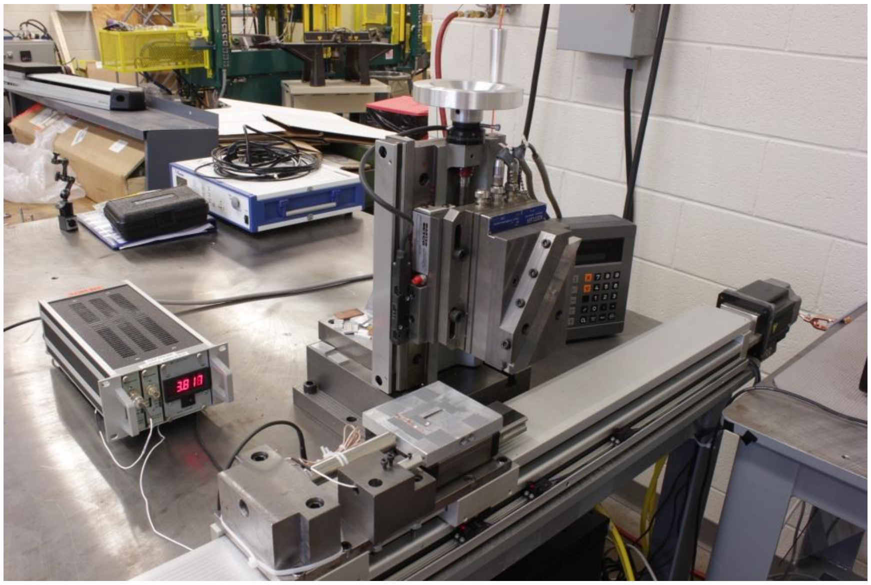


Figure A.4: Isometric view of planer assembly, analogous to Fig. 4.1.

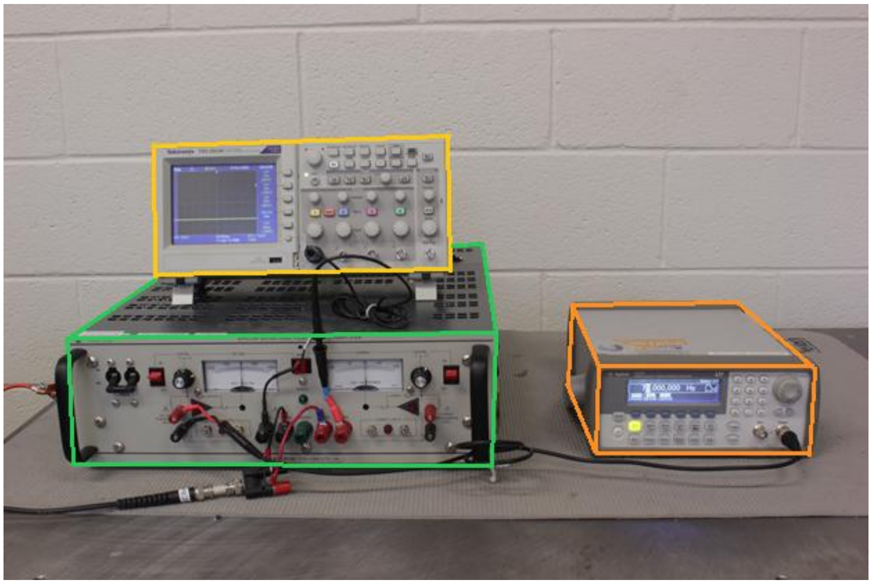


Figure A.5: Devices used to generate and monitor signal sent to piezo-ceramic actuator. Specifically, an Agilent 33220A waveform generator (outline in orange, right), a Kepco BOP 100-4M voltage amplifier / power supply (outlined in green, bottom left) and a Taktronix TDS2024C oscilloscope (outlined in yellow, top left) were used.

Appendix B: Program flowchart

Summary: The following diagram is a flowchart used to represent the interaction between the various data sources, acquisition routines, and analysis programs. These programs are provided in the following appendices, as indicated by the diagram.

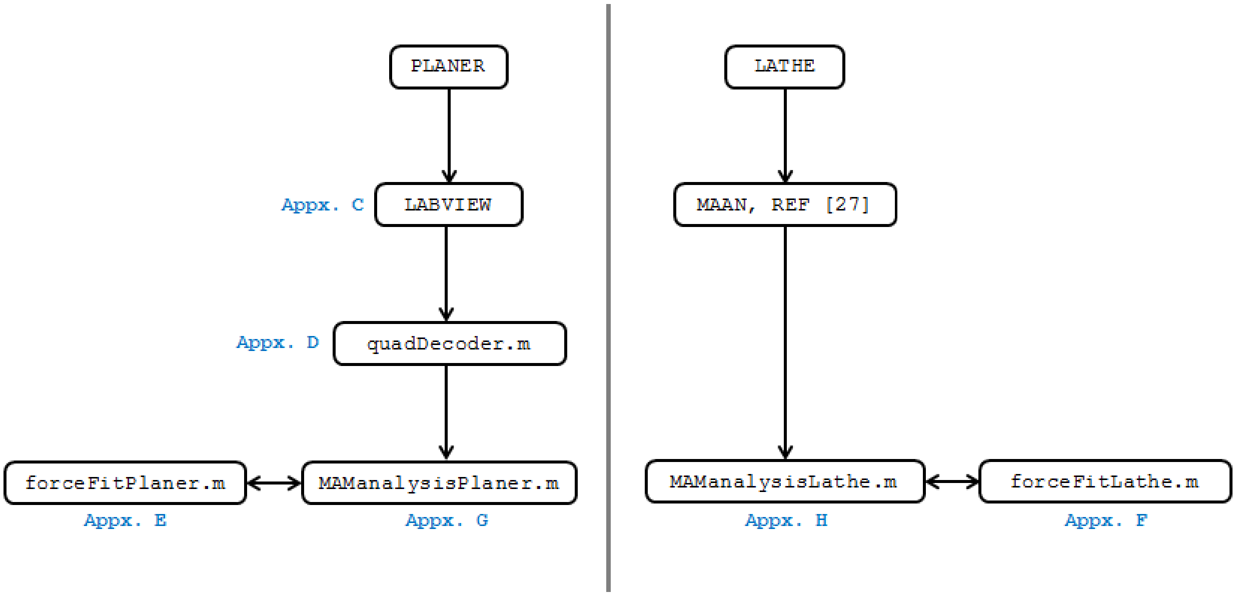
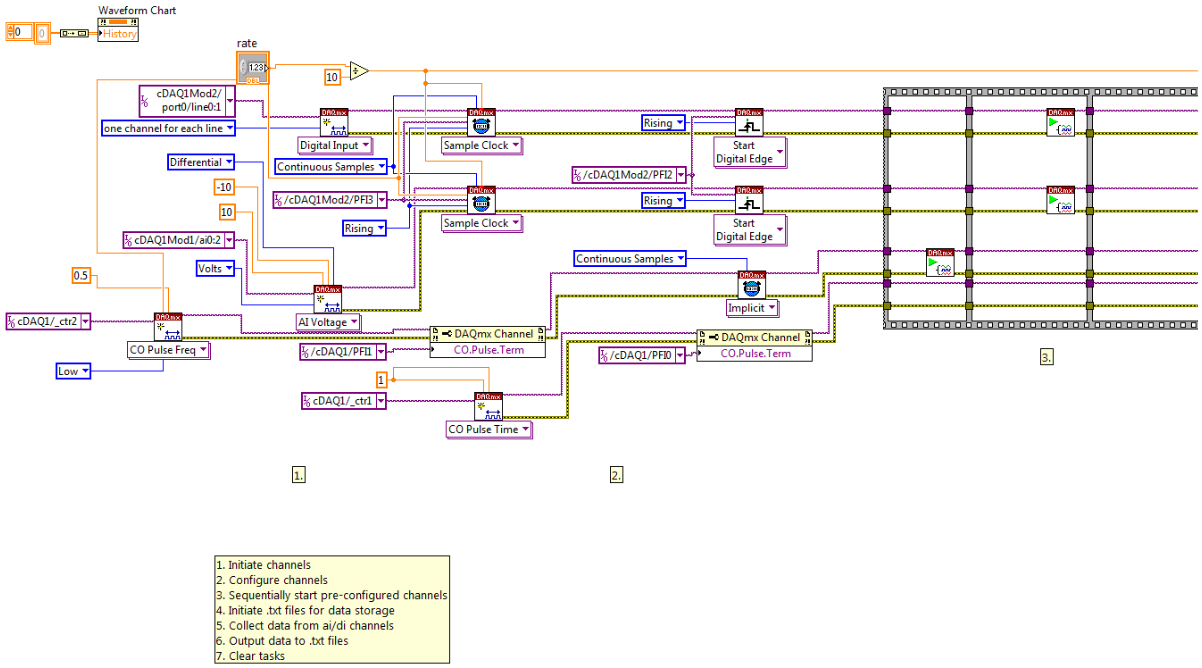


Fig. B.1: Flowchart of data sources → acquisition routines → analysis programs.

Appendix C: NI Labview data acquisition VI

Summary: The Labview VI seen below in Figs. B.1, B.2 was used to collect 3 channels of analog voltages and 2 channels of digital voltages simultaneously at a specified sampling frequency (10 kHz). The hardware which this program accessed can be seen in Fig. A.2. The analog data was stored in a user-named .txt file with the classifier "_A" appended to the end of the filename. Similarly, the digital data was stored in its own .txt file with the classifier "_D" appended to its filename. Each channel was stored as a separate column in these files. Furthermore, each file contained a single headerline detailing the respective gain settings of the Fp and Fq charge amplifiers.



1. Initiate channels
2. Configure channels
3. Sequentially start pre-configured channels
4. Initiate .txt files for data storage
5. Collect data from ai/di channels
6. Output data to .txt files
7. Clear tasks

Figure C.1: Left half of VI's block diagram.

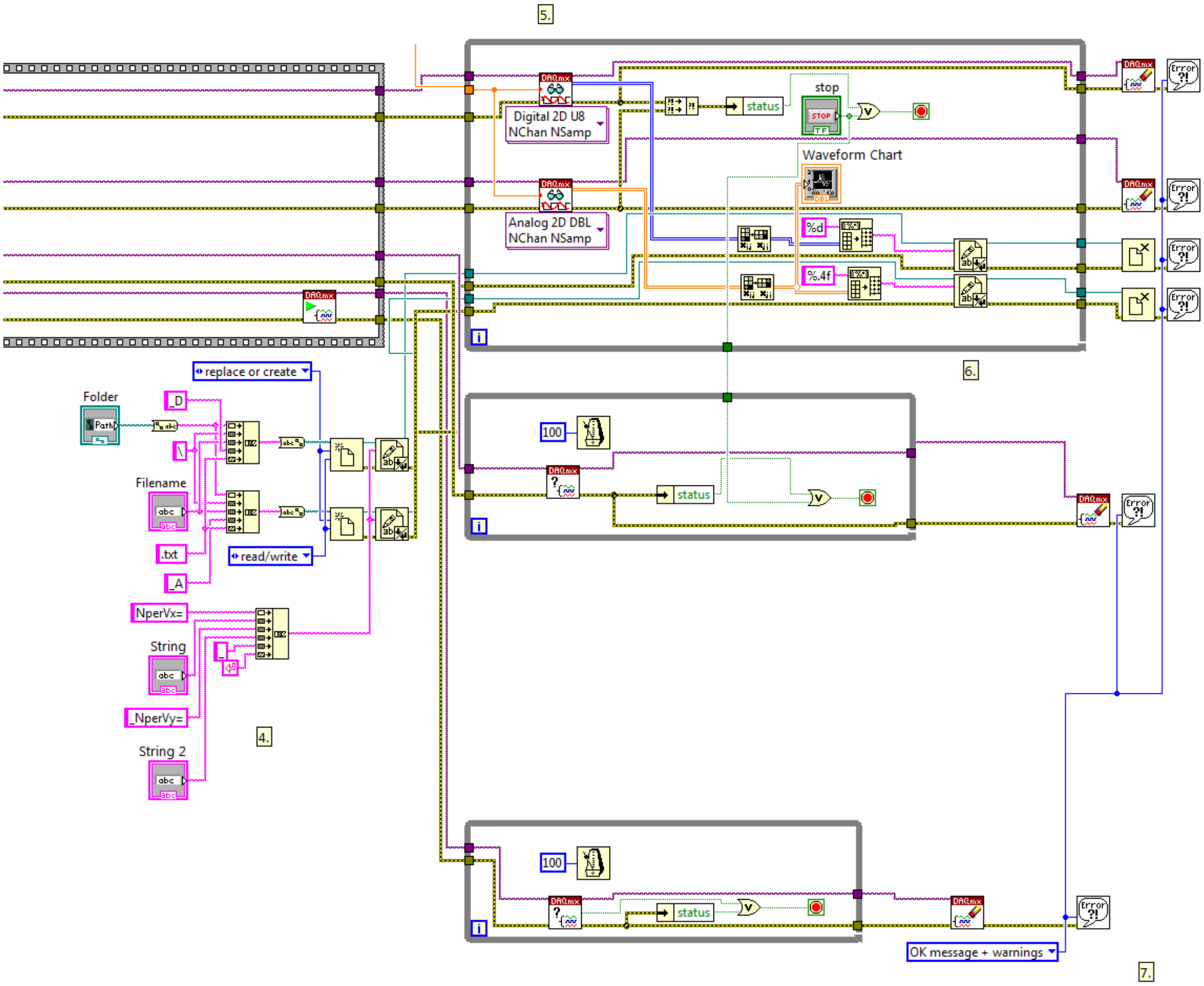
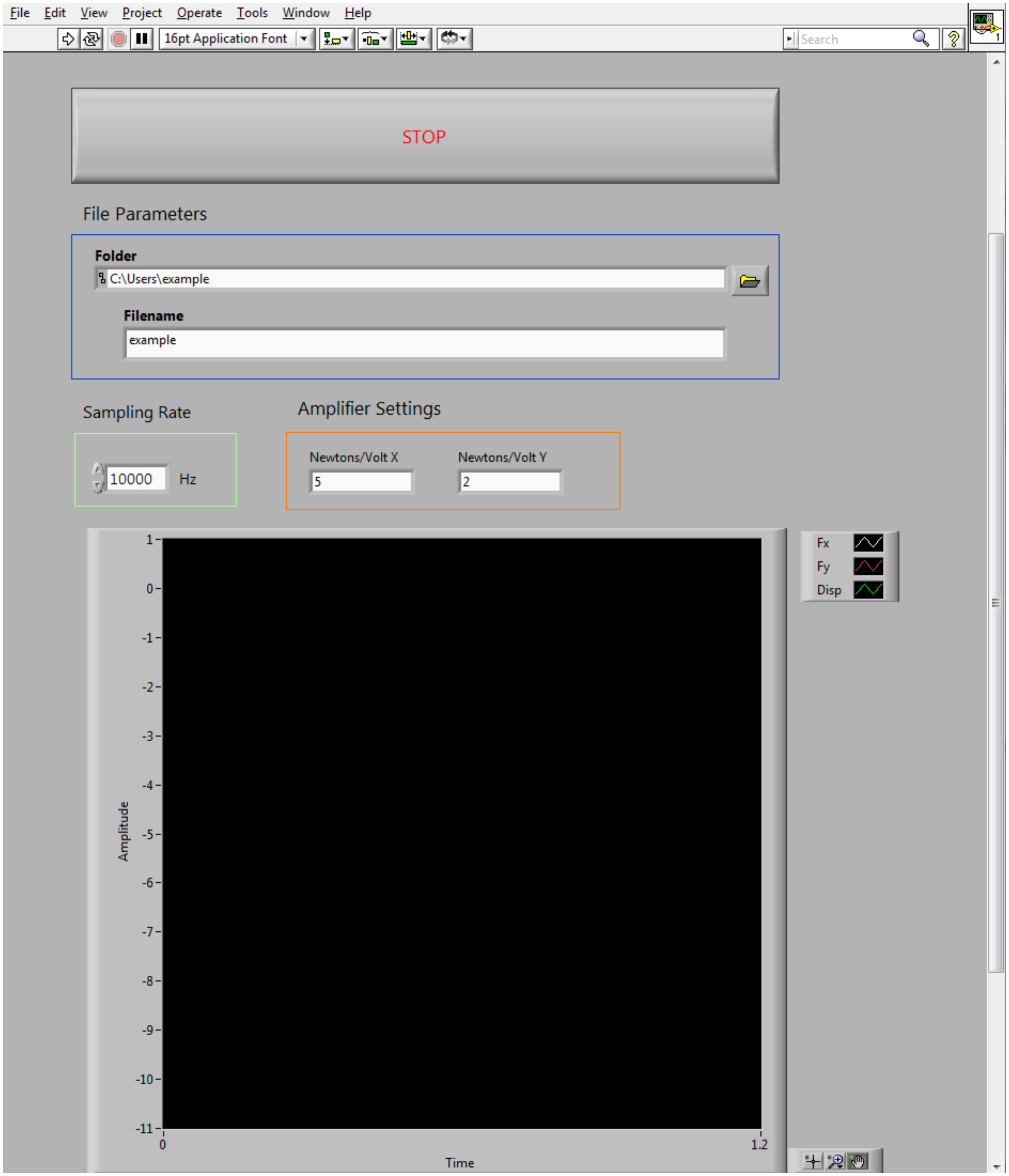


Figure C.2: Right half of VI's block diagram.



File Parameters

Folder
C:\Users\example

Filename
example

Sampling Rate
10000 Hz

Amplifier Settings

Newtons/Volt X
5

Newtons/Volt Y
2

Figure C.3: Front panel of VI.

Appendix D: Quadrature decoding routine (Matlab)

Summary: The purpose of the Matlab code presented here is to decode the digital data sent from the rotary encoder and collected via the NI-9411 and Labview into a series of "steps." Each post-quadrature count on the rotary encoder represents a step from which 3 conditions arise: either the encoder sees a forward rotation between samplings (step=+1), no rotation (step=0), or a backward rotation (step=-1). Once decoded, the routine imports the corresponding analog data from a separate text file and incorporates this along with the vector of steps into a new .txt file. The original data from the analog and digital files is stored in subfolder, in case future reference in required.

```
%% quadDecoder.m
%
% WRITTEN BY: JOSH NORMAN
% DATE: 10/24/12
% PROGRAM SUMMARY: DECODES QUADRATURE ENCODER PULSES INTO A FORM OF
% RELATIVE LINEAR POSITION. DATA IS IMPORTED FROM USER SELECTED "*_D.TXT"
% FILE WHICH IS IN A 2 COLUMN FORMAT.  COLUMN 1 IS TRACK (A) AND COLUMN 2
% IS TRACK (B). THE DECODED POSITION DATA IS THEN OUTPUTTED TO A NEW TEXT
% FILE ALONG WITH RAW DATA FROM THE ANALOG "*_A.TXT" FILE.  THE TWO
% ORIGINAL .TXT FILES ARE MOVED TO A "RAW" SUBFOLDER IN THE WORKING
% DIRECTORY IN CASE FUTURE REFERENCE IS REQUIRED.
%
% >>>>>>>>>>                MECHANICAL UNIT = mm                <<<<<<<<<<

clc; close all; clear all;

%% INPUTS

encRes = 8000;              %post quadrature encoder resolution [counts/rev]
lead = 20;                  %ball-screw lead [MU/rev]
MUperCOUNT = lead/encRes;   % [MU/count]

%% CHOOSE TXT FILE TO IMPORT

% Select "relevant" files
wildcard_txt = '*D.txt';
files_struct = dir(wildcard_txt);                   %list all encoded _D.txt files in structure
variable
files_cell = struct2cell(files_struct);             %list these files in cell
names_cell = files_cell(1,:);                       %select first row of cell b/c it contains
names
names_char = char(names_cell);                      %convert this cell row of names to characters
[m,n] = size(names_char);                           %use m to know how many files are available

if m==0
   error('NO ENCODED TXT FILES ARE LOCATED IN CURRENT DIRECTORY')

elseif m<10     %single digit number of files
   for i = 1:m
        FileOptions(i,:) = sprintf('%i) %s',i,names_char(i,:));     %create character array of
file options to choose from
   end

elseif m>=10    %double digit number of files
```

```
    for i = 1:9
        FileOptions(i,:) = sprintf('% i) %s',i,names_char(i,:));    %create character array of
file options to choose from
    end
    for i = 10:m
        FileOptions(i,:) = sprintf('%i) %s',i,names_char(i,:));     %create character array of
file options to choose from
    end
end

% Print filenames to workspace
disp(' ')
fprintf('\t\t\tANALYSIS OF EXPERIMENTAL DATA')
fprintf('\n\n')
disp(FileOptions)                                       %print list of numericized _D.txt files
to workspace
fprintf('\n\n')

% Choose file on which to run analysis
fileChoice = input('Enter File #   -->  ');             %user enters number corresponding to
desired file
filename_ext_D = deblank(names_char(fileChoice,:));
[PATHSTR, filename, EXT] = fileparts(filename_ext_D);

%% IMPORT DATA FROM TXT FILE

% Read .txt File
fidD = fopen(filename_ext_D);                           %open the file
header = fscanf(fidD,'%s',1);                           %store the first line of the file as the
header
DATAcell = textscan(fidD,'%f %f','headerlines',1);      %import data from file into cell variable
DATAmat = cell2mat(DATAcell);                           %convert cell variable to matrix
fclose(fidD);                                           %close the file

% Assign Data Channels
raw.A = DATAmat(:,1);      %[s]                         %first column of data
raw.B = DATAmat(:,2);      %[mm]                        %second column of data

% Find length of these data columns
N = length(raw.A);

%% CONVERT DATA TO BOOLEAN
% If data is not in boolean form yet (ie 0 or 1) this will convert it.
% Otherwise, it will be left as is.

bool.A = raw.A>0;
bool.B = raw.B>0;

%% CREATE DIFFERENTIAL VARIABLES
% Where diff.A(i) = bool.A(i)-bool.A(i-1)
% and   diff.B(i) = bool.B(i)-bool.B(i-1)

%Create variables of same data that's been shifted forward 1 row
shift.A(2:N+1,1) = bool.A;
shift.B(2:N+1,1) = bool.B;

%Eliminate last entry
shift.A(end) = '';
shift.B(end) = '';

%Create differential variables
diff.A = bool.A-shift.A;
diff.B = bool.B-shift.B;

%% CONVERT A AND B TRACKS TO STEP NUL/FWD/REV
% A "step" is defined as a single count on the rotary encoder wheel.  Thus,
% each step can be converted to relative linear displacement by multiplying
% by MUperCOUNT.  NUL/FWD/REV are to be represented as 0/1/-1 respectively.  Due
% to oversampling, a majority of these should be STEP NUL, or 0, (ie encoder
% remained quasi-stationary).

%Initialize step vector
stepVec = zeros(N,1);

%Prepare waitbar divider.  This simple indicates approximate amount of file
%that has been decoded at any given time.
divideWait = 10^(floor(log10(N))-1);
hWait = waitbar(0,'Decoding...');

%Start Main Loop
for i = 2:N                         %must start at 2nd index due to shifted variables
    %Create reusable variables
    diff.Ai = diff.A(i);
    diff.Bi = diff.B(i);
    bool.Ai = bool.A(i);
    bool.Bi = bool.B(i);

    %Determine NUL/FWD/REV
    switch diff.Ai
        case 0
            switch diff.Bi
                case 0
                    stepi = 0;      %NUL
                case 1
                    if bool.Ai==0
                        stepi = -1; %REV
                    else
                        stepi = 1;  %FWD
                    end
                case -1
                    if bool.Ai==0
                        stepi = 1;  %FWD
                    else
                        stepi = -1; %REV
                    end
            end
        case 1
            if bool.Bi==0
                stepi = 1;          %FWD
            else
                stepi = -1;         %REV
            end
        case -1
            if bool.Bi==0
                stepi = -1;         %REV
            else
                stepi = 1;          %FWD
            end
    end

    %Write stepi to ith position in stepVec
    stepVec(i) = stepi;             %vector of 0/1/-1 indicating steps
    if round(i/divideWait)==i/divideWait %if condition is met...
        waitbar(i/N,hWait)          %...update waitbar
    end
end
close(hWait);                       %close waitbar once decoding is finished

%% OUTPUT DECODED DATA TO FILE

% Import analog data
filename_A = filename;              %initialize variable
filename_A(end) = 'A';              %choose analog file instead of digital
filename_ext_A = sprintf('%s.txt',filename_A);
fidA = fopen(filename_ext_A);       %open analog data file
DATAcellA = textscan(fidA,'%f %f %f %f','headerlines',1); %read in analog data to cell variable
DATAmatA = cell2mat(DATAcellA);     %convert cell variable to matrix
```

```
fclose(fidA);                       %close analog data file

% Assign data channels
raw.Fx = DATAmatA(:,1);             %first column
raw.Fy = DATAmatA(:,2);             %second column
raw.D_AC = DATAmatA(:,3);           %third column

% Output header to new file
filename_NEW = filename(1:end-2);
filename_ext_NEW = sprintf('%s.txt',filename_NEW);  %new filename to which all relevant data is
written for use in numerical analysis routines
fidNEW = fopen(filename_ext_NEW,'w+t');
fprintf(fidNEW,'%s\n',header);

% Output column headers
fprintf(fidNEW,'Fx Fy VcapProbe StepVector\n');

% Output data to new file
fprintf(fidNEW,'%.4f %.4f %.4f %i\n',[raw.Fx,raw.Fy,raw.D_AC,stepVec]');
fclose(fidNEW);

% Create Directory for RAW (*_A and *_D) files
mkdir(pwd,'RAW');                   %create "RAW" folder if it doesn't already exist in working
directory

% Move RAW files into this folder
movefile(filename_ext_D,['RAW/' filename_ext_D]);
movefile(filename_ext_A,['RAW/' filename_ext_A]);
```

Appendix E: Planer force data fitting routine (Matlab)

Summary: This program receives cutting velocity and average Fp values from static cutting experiments. The number of trials represented by the average force values can be seen in Table 6.1. The force-velocity data is fit according to the equation Fp(V) = A*ln(V) + B which was elucidated from Ref. [11]. Since this equation is linear with respect to the fitting parameters A and B, the least squares solution can be obtained through standard linear algebra techniques. Specifically, if

$$M = \begin{bmatrix} V_{DC,1} & 1 \\ V_{DC,2} & 1 \\ \vdots & \vdots \\ V_{DC,5} & 1 \end{bmatrix}$$

$$n = \begin{bmatrix} A \\ B \end{bmatrix}$$

$$P = \begin{bmatrix} Fp_{avg,1} \\ Fp_{avg,2} \\ \vdots \\ Fp_{avg,5} \end{bmatrix}$$

then the vector n, which contains the fitting parameters, can be solved according to

$$n = (M^T M)^{-1} M^T P.$$

```
%% forceFitPlaner.m
%
% WRITTEN BY: JOSH NORMAN
% DATE: 12/17/12
% PROGRAM SUMMARY: RECEIVES CUTTING VELOCITY AND AVERAGE FORCE VALUES FROM
% STATIC CUTTING TRIALS AND USES LEAST SQUARES FITTING TECHNIQUE TO FIT
% FORCE DATA ACCORDING TO THE SHAPE F=A*ln(x)+B WHICH WAS DERIVED FROM
% Maan, N., and Broese Van Groenou, A., 1977, “Low Speed Scratch
% Experiments on Steels,” Wear, 42(2), pp. 365-390.

clc; close all; clear all;

%% INPUTS

%Static cutting speeds
Vdc = [
    .05
    .25
    .50
    .75
    1.00
    ];  %[mm/s]
```

```
%Average forces in cutting direction
Favg = [
    19.862
    20.821
    21.19
    21.498
    21.364
    ];   %[N]

%% LEAST SQUARES FITTING

x = Vdc;
y = Favg;

A = [log(x),ones(length(x),1)];     %fitting shape
B = y;                              %data
X = inv(A'*A)*A'*B;                 %linear algebra solution method for least squares fitting

Ffit = X(1)*log(x) + X(2);

%Create variables to be used to visualize best fit curve
xPlot = .01:.01:2;
yfitPlot = X(1)*log(xPlot) + X(2);

%% EVALUATE FIT

RSS = sum((y-Ffit).^2);             %residual sum of squares
TSS = sum((y-mean(y)).^2);          %total sum of squares
R2F = 1-(RSS/TSS);                  %R-squared value

%% PLOTTING

hfig1 = figure(1);
set(hfig1,'color','w')
set(hfig1,'name','Force')

hold on
plot(Vdc,Favg,'.b','markersize',25)
plot(xPlot,yfitPlot,'r','linewidth',2)
hold off
xlim([0,1.05])
ylim([17,24])
set(gca,'YTick',[17:1:24])

set(gca,'fontsize',22)
hx = xlabel('Velocity (mm/s)');
hy = ylabel('Force (N)');
set(hx,'fontsize',24)
set(hy,'fontsize',24)
strT = sprintf('Al6061-T6\n\n\n\n')
hT = title(strT);
set(hT,'fontsize',30)
strT2 = sprintf('Force=A*ln(V) + B\nA=%.4e | B=%.4e\nR^2=%.4f',X(1),X(2),R2F);
hT2 = gtext(strT2);
set(hT2,'fontsize',24,'horizontalalignment','center')

%% DISPLAY TO SCREEN

disp(' ')
disp('Coefficients')
disp(sprintf('%.4e %.4e',X(1),X(2)))

%%                      ^^^   F O R C E   ^^^
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       vvv   P O W E R   vvv

%% POWER
% Least-squares fitting technique IS NOT performed for power.  Rather the
% values for the fitting variables determined above for force are used here
% and a known equation for power as a function of force and velocity is
% used.

Pavg = Favg.*Vdc / 1000;                                        %power "data"
PfitPlot = X(1)*log(xPlot).*(xPlot/1000) + X(2)*(xPlot/1000);   %power "fit"

%% EVALUATE FIT
% Even though fitting technique was not re-performed, we can still
% determine how accurately the resulting equation "fits" the power values
% derived from data.

Pfit = X(1)*log(x).*(x/1000) + X(2).*(x/1000);
RSS = sum((Pavg-Pfit).^2);
TSS = sum((Pavg-mean(Pavg)).^2);
R2P = 1-(RSS/TSS);

%% PLOTTING

hfig2 = figure(2);
set(hfig2,'color','w')
set(hfig2,'name','Power')

hold on
plot(Vdc,Pavg,'.','color',[.2 .5 .2],'markersize',35)
plot(xPlot,PfitPlot,'r','linewidth',2)
hold off
xlim([0,1.05])
ylim('auto');
y=ylim;
ylim([0,y(2)])

set(gca,'fontsize',22)
hx = xlabel('Velocity (mm/s)');
hy = ylabel('Power (W)');
set(hx,'fontsize',24)
set(hy,'fontsize',24)
strT = sprintf('Al6061-T6\n\n\n\n');
hT = title(strT);
set(hT,'fontsize',30)
strT2 = sprintf('Power=A*ln(V)*(Ve-3) + B*(Ve-3)\nA=%.4e | B=%.4e\nR^2=%.4f',X(1),X(2),R2P);
hT2 = gtext(strT2);
set(hT2,'fontsize',24,'horizontalalignment','center')
```

Appendix F: Lathe force data fitting routine (Matlab)

Summary: This program receives undeformed chip thickness and average Fp values from static cutting experiments. The number of trials represented by the average force values can be seen in Table 6.2. The force-thickness data is fit according to the equation $Fp = AX + B - \frac{B}{CX+1}$ which was elucidated from Ref. [10]. Parameters A and B were solved by fitting a line through the last two data points (in order to best-approximate the oblique asymptote). The non-linear fitting parameter, C, was solved using the Newton-Raphson convergence algorithm.

```
%% forceFitLathe.m
%
% WRITTEN BY: JOSH NORMAN
% DATE: 12/17/12
% PROGRAM SUMMARY: RECEIVES UNDEFORMED CHIP THICKNESS AND AVERAGE FORCE
% VALUES FROM STATIC CUTTING TRIALS AND USES LEAST SQUARES FITTING
% TECHNIQUE TO FIT FORCE DATA ACCORDING TO THE SHAPE F=m*x+b-(b/(A*x+1))
% WHICH WAS DERIVED FROM Arsecularatne, J. A., 1997, “On Tool-Chip
% Interface Stress Distributions, Ploughing Force and Size Effect in
% Machining,” Int. J. Mach. Tools Manuf., 37(7), pp. 885-899.

clc; close all; clear all;

%% INPUTS

V = 1.5180;     %cutting velocity (m/s) taken from analysis program

%Static values of undeformed chip thickness
h = [
    5
    10
    20
    30
    40
    50
    ];  %[um]

%Average forces in cutting direction
Favg = [
    21.287
    31.559
    43.529
    55.845
    67.033
    75.719
    ];   %[N]

%% EQUATION OF ASYMPTOTE
% Best guess for asymptote is a straight line through the last two data
% points

P = polyfit(h(end-1:end),Favg(end-1:end),1);
m = P(1);   %slope of line
b = P(2);   %y-intercept of line

%% PERFORM NEWTON-RAPHSON
% This is performed since the fitting equation has a single non-linear
% fitting variable, referred to as "A" here

ADiffLim = .0001;
i=2;
A = [1 0];      %initial guess=0
```

```
while abs(A(i)-A(i-1))>ADiffLim     %determines whether variable has settled to a relatively
constant value between iterations

% EQUATION OF REMAINDER
% Remainder must cancel out y-intercept of asymptote such that curve passes
% through origin.  Also, a fitting coefficient must be involved which
% determines how quickly the curve deviates from the asymptote.  It has
% been observed that the final curve has a concave shape. Thus, the
% remainder will be of the form -b/(Ax+1)

% FIRST DERIVATIVE
D1 = sum((Favg - m*h - b + (b./(A(i)*h+1))) .* ((-2*b*h)./((A(i)*h + 1).^2)));

% SECOND DERIVATIVE
D2 = sum(((Favg - m*h - b + (b./(A(i)*h+1))) .* ((4*b*(h.^2))./((A(i)*h + 1).^3))) +
((2*b*h)./((A(i)*h+1).^2)).*((b*h)./((A(i)*h+1).^2)));

A(i+1) = A(i)-(D1/D2);              %calculate next guess for variable A

i = i+1;
end

Af = A(i);                          %choose last iteration as the best fit
Ffit = m*h + b + (-b./(Af*h + 1));  %fitted force equation

%Create variables to be used to visualize best fit curve
hPlot = 0:.1:max(h);
FfitPlot = m*hPlot + b + (-b./(Af*hPlot + 1));

%% EVALUATE FINAL FIT

RSS = sum((Favg-Ffit).^2);          %residual sum of squares
TSS = sum((Favg-mean(Favg)).^2);    %total sum of squares
R2 = 1-(RSS/TSS);                   %R-squared value

%% PLOTTING

hfig1 = figure(1);
set(hfig1,'color','w')
set(hfig1,'name','Force')

hold on
plot(h,Favg,'.b','markersize',25)
plot(hPlot,FfitPlot,'r','linewidth',2)
hold off
xlim([0,50])
ylim([0,125])

set(gca,'fontsize',22)
hx = xlabel('h, undeformed chip thickness (um)');
hy = ylabel('Force (N)');
set(hx,'fontsize',24)
set(hy,'fontsize',24)
strT = sprintf('Al6061-T6\n\n\n');
hT = title(strT);
set(hT,'fontsize',30)
strT2 = sprintf('F = a*h + b - b/(c*h+1)\na=%.4e | b=%.4e | c=%.4e\nR^2=%.4f',m,b,Af,R2);
hT2 = gtext(strT2);
set(hT2,'fontsize',24,'horizontalalignment','center')

%% DISPLAY TO SCREEN

disp(' ')
disp('Coefficients')
disp(sprintf('%.4e %.4e %.4e',m,b,Af))
disp(' ')
disp('Goodness of Fit')
disp(sprintf('R^2 = %.4f',R2))

%%                      ^^^   F O R C E   ^^^
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%                       vvv   P O W E R   vvv

%% POWER FIT
% Least-squares fitting technique IS NOT performed for power.  Rather the
% values for the fitting variables determined above for force are used here
% and a known equation for power as a function of force and velocity is
% used.

Pavg = Favg*V;              %power "data"
PfitPlot = FfitPlot*V;      %power "fit"

%% PLOTTING

hfig2 = figure(2);
set(hfig2,'color','w')
set(hfig2,'name','Power')

hold on
plot(h,Pavg,'.','color',[.2 .5 .2],'markersize',25)
plot(hPlot,PfitPlot,'r','linewidth',2)
hold off

ylim([0,200])

set(gca,'fontsize',22)
hx = xlabel('h, undeformed chip thickness (um)');
hy = ylabel('Power (W)');
set(hx,'fontsize',24)
set(hy,'fontsize',24)
strT = sprintf('Al6061-T6\n\n\n');
hT = title(strT);
set(hT,'fontsize',30)
strT2 = sprintf('P = V*[a*h + b - b/(c*h+1)]\na=%.4e | b=%.4e | c=%.4e\nR^2=%.4f',m,b,Af,R2);
hT2 = gtext(strT2);
set(hT2,'fontsize',24,'horizontalalignment','center')
```

Appendix G: Planer analysis routine (Matlab)

Summary: This routine imports planer-specific data (Fx,Fy,D_{AC},D_{DC}) from a user-specified .txt file and performs numerical analysis techniques. Primary code features include:

- determination of effective instantaneous displacement and velocity (via encoder data combined with that from capacitance probe)
- determination of best-fit equations characterizing said displacement
- determination of beginning and end of cut
- determination of beginning and end of individual periodic cutting instances
- force prediction routine based on Fp(Veff) found using forceFitPlaner.m
- specific energy calculations.

Note: This code refers to Fp/Fq as Fx/Fy, respectively.

```
%% MAManalysisPlaner.m
%
% WRITTEN BY: JOSH NORMAN
% PROGRAM SUMMARY: THIS ROUTINE IMPORTS PLANER-SPECIFIC DATA FROM A USER
% SPECIFIED .TXT FILE AND PERFORMS NUMERICAL ANALYSIS. PRIMARY FEATURES
% INCLUDE:
%       DETERMINATION OF EFFECTIVE INSTANTANEOUS DISPLACMENT AND VELOCITY
%           (ENCODER DATA COMBINED WITH THAT FROM CAPACITANCE PROBE) AS
%           WELL AS BEST FIT EQUATION CHARACTERIZING THIS DISPLACEMENT.
%       DETERMINATION OF BEGINNING AND END OF CUT.
%       DETERMINATION OF BEGINNING AND END OF PERIODIC CUTTING INSTANCES.
%       FORCE PREDICTION ROUTINE BASED ON Fp(Veff) FOUND USING
%           forceFitPlaner.m.
%       SPECIFIC ENERGY CALCULATIONS.
%       NOTE: Fp/Fq ARE REFERRED TO AS Fx/Fy RESPECTIVELY.

clc; clear all; close all;

%% INPUTS

specD = 1;                  %flag for determining which method will used to find "cutting"
portion of data (0/1 = total/specified distance)
Dwait = 5;                  %distance to wait between start of cut and start of analysis
Drel = 15;                  %relevant distance over which work will be determined [mm]
percentTrig1 = .50;         %percentage of Fx range for triggering
percentTrig2 = .50;         %percentage of Fx range for triggering

fs = 10000;                 %sampling rate [Hz]
encRes = 8000;              %post quadrature encoder resolution [counts/rev]
lead = 20;                  %ball-screw lead [MU/rev]
MUperCOUNT = lead/encRes;   % [MU/count]
mmPerV = .025;              %mm per Volt (capacitance probe)
w = 1.003;                  %workpiece width [mm]

%% CHOOSE TXT FILE TO IMPORT

% Select "Relevant" Files
wildcard_txt = '*.txt';
```

```
files_struct = dir(wildcard_txt);                   %list all .txt files in structure
files_cell = struct2cell(files_struct);             %list these files in cell
names_cell = files_cell(1,:);                       %select first row of cell b/c it contains
names
names_char = char(names_cell);                      %convert this cell row of names to characters
[m,n] = size(names_char);                           %use m to know how many files are available

if m==0
   error('NO TXT FILES ARE LOCATED IN CURRENT DIRECTORY')

elseif m<10     %single digits
   for i = 1:m
        FileOptions(i,:) = sprintf('%i) %s',i,names_char(i,:));     %create character array of
file options to choose from
   end

elseif m>=10    %double digits
    for i = 1:9
        FileOptions(i,:) = sprintf('% i) %s',i,names_char(i,:));    %create character array of
file options to choose from
    end
    for i = 10:m
        FileOptions(i,:) = sprintf('%i) %s',i,names_char(i,:));     %create character array of
file options to choose from
    end
end

% Print Filenames to Workspace
disp(' ')
fprintf('\t\t\tANALYSIS OF EXPERIMENTAL DATA')
fprintf('\n\n')
disp(FileOptions)                                                   %print list to workspace
fprintf('\n\n')

% Choose file on which to run analysis
fileChoice = input('Enter File #   -->  ');                         %user enters number
corresponding to file
filename_ext = deblank(names_char(fileChoice,:));                   %remove placeholding spaces
from end of selected filename
[PATHSTR, filename, EXT] = fileparts(filename_ext);

%% IMPORT DATA FROM TXT FILE

% Read .txt File
fid = fopen(filename_ext);
header = fscanf(fid,'%s',1);                                %store first line of file to variable
"header"
DATAcell = textscan(fid,'%f %f %f %f','headerlines',2);     %read in all four columns of data to
cell variable, skipping first two lines of file
DATAmat = cell2mat(DATAcell);                               %convert data from cell to matrix
fclose(fid);                                                %close file

%Automatically determine fm
fmInd = strfind(filename,'fm');                             %find start index
HzInd = strfind(filename,'Hz');                             %find end index
fm = str2num(filename(fmInd+2:HzInd-1));                    %use indices to locate and store
modulation frequency [Hz]

% Automatically determine NperVx and NperVy
NperVx_start = min(strfind(header,'='))+1;                  %find start index
NperVx_end   = min(strfind(header,'_'))-1;                  %find end index
NperVy_start = max(strfind(header,'='))+1;                  %find start index
NperVy_end   = max(strfind(header,'_'))-1;                  %find end index
NperVx = str2num(header(NperVx_start:NperVx_end));          %use indices to locate and store
amplifier gain value as number [N/V]
NperVy = str2num(header(NperVy_start:NperVy_end));          %use indices to locate and store
amplifier gain value as number [N/V]

% Automatically determine desired Vdc
Vdc_des_start = strfind(filename,'Vdc')+3;                  %find start index
Vdc_des_end   = strfind(filename,'mms')-1;                  %find end index
```

```
Vdc_des = str2num(strrep(filename(Vdc_des_start:Vdc_des_end),'-','.')); %use indices to locate
and store value as number

% Automatically determine depth of cut h0
h0_stop = strfind(filename,'um')-1;                         %find end index (note: this variable
must be the first item in the filename)
h0 = str2num(filename(2:h0_stop))/1000;                     %depth of cut [mm]

% Assign Data Channels
Fx = NperVx*(-DATAmat(:,1));                                %aka Fp [N]
Fy = NperVy*DATAmat(:,2);                                   %aka Fq [N]
ch2 = DATAmat(:,3);                                         %raw cap probe data [V]
steps = DATAmat(:,4);                                       %encoder counts (+1/0/-1 =
FWD/NUL/REV)

% Calculate cumulative "DC" displacement
D_DC  = MUperCOUNT*cumsum(steps);                           %use "steps" variable created by
quadDecoder.m to find linear displacement [mm]

%% LINEARLY INTERPOLATE
% The purpose of this section is to smooth out the innately choppy digital
% data by linearly interpolating between the midpoints of each 2.5um step

% Find Voffset of Cap Probe
switch fm
    case 0
        Voffset = ch2(1);
    otherwise
        NperPeriod = floor(fs/fm);              %No. of samples per period
        Voffset = mean(ch2(1:NperPeriod));      %cap probe voltage offset found by averaging 1
period of measurements
end

% Find length of data sets
N = length(Fx);

% Create time vector
dT = 1/fs;          %[s]
T  = dT*(1:1:N)';   %[s]

% Get indices of points where displacement has changed
dD_ind = (1:1:N)'.*abs(steps);

% Eliminate all entries that are 0
dD_ind(dD_ind==0) = [];

%Create two vectors of indices offset from each other by 1 row
dD_indShift = dD_ind;           %initialize vector
dD_indShift(2:end+1) = dD_ind;  %shift values down 1 row
dD_indShift(1) = 1;             %then make 1st entry == 1
dD_ind(end+1) = N;              %add entry at end of vector

% Find integer midpoints of these indices
dD_indMids = floor((dD_ind+dD_indShift)/2);

% Convert to time
T_Mids = dD_indMids*dT;

% Find displacement values at these times
D_Mids = D_DC(dD_indMids);

% Linearly interpolate between these values
D_lin = zeros(N,1);
for i = 1:length(dD_indMids)-1
    ind_curr = dD_indMids(i);
    ind_next = dD_indMids(i+1);
    D_curr = D_DC(ind_curr);
    D_next = D_DC(ind_next);
    D_lin(ind_curr:ind_next) = linspace(D_curr,D_next,ind_next-ind_curr+1);
end
D_lin(ind_next:end) = D_next;   %fill in remaining points with last distance midpoint value
```

```
%% ADD AC COMPONENT TO DC
% Note: AC refers to the dynamic portion of displacment which is induced by
% the piezo-ceramic actuator and monitored using a capacitance probe.  DC
% refers to the static portion of the displacement trace which is induced
% by the servo motor and monitored using a quadrature rotary encoder. The
% DC component is in fact not always constant throughout, as the motor has
% the tendency to backtrack on itself under oscillatory loads.  This will
% be taken into account when Deff is characterized using least-squares
% fitting.

% Let new D_DC be represented by D_lin
D_DC = -D_lin;              %negative is due to sign convention in that encoder sees a negative
step when workpiece is moving into the tool [mm]

% Find scaled and mean adjusted AC displacement data
D_AC = mmPerV*(ch2-Voffset);%dynamic component of displacement. Note that higher values of ch2
voltage indicate + displacement according to convention [mm]

% Add AC to DC
D_ACDC = D_DC+D_AC;         %effective displacement [mm]

%% DETERMINE START/STOP OF CUTTING
% These conditions are determined by monitoring the Fx (aka Fp) trace and
% noting when it rises above critical values.  These critical values are a
% percentage of the Fx signal's total range.

% Smooth out inconsistencies
FxSmooth = smooth(Fx,100);

xPercent = (FxSmooth-min(FxSmooth))/(max(FxSmooth)-min(FxSmooth));    %each Fx entry as percent
of Fx range
c1 = 1;                                     %initiate counter 1
while xPercent(c1) < percentTrig2           %looks for critical percent of > percentTrig
    c1 = c1+1;                              %count forward 1 index per loop iteration
end
cutStart = c1;                              %location in vector where cutting data starts

% Use switch to determine whether the entire length, specD=0, or just a
% specified distance, specD=1, will be used to represent the cut (ie used
% to calculate energy, etc). It is often advantageous to monitor only a
% portion of the total trace since the beginning and end may be
% misrepresentative of the crucial part of the cut.
switch specD

    case 0                                  %use entire cutting range
        c2 = length(FxSmooth);              %initiate counter 2 where at end of data set
        while xPercent(c2) < percentTrig2   %check if condition is met
            c2 = c2-1;                      %... if not, count backward 1 index per loop
iteration
        end
        cutStop = c2;                       %location in vector where cutting data stops

    case 1                                  %use specified cutting range
        Dstart = D_ACDC(cutStart);
        Dstart = Dstart+Dwait;
        while D_ACDC(cutStart)<Dstart       %check if condition is met
           cutStart=cutStart+1;             %... if not, add 1 to index
        end
        Dstop = Dstart + Drel;
        c2 = c1;
        while D_ACDC(c2)<Dstop              %check if condition is met
            c2 = c2+1;                      %... if not, count forward 1 index per loop iteration
        end
        cutStop = c2;                       %index at which cutting "stops"
end

FxRel = mean(Fx(cutStart:cutStop));         %newly found relevant range of Fx (aka Fp) data [N]

%% DETERMINE START/STOP OF PEAK PLATEAU
```

```
c1 = 1;                                     %initiate counter 1
while xPercent(c1) < percentTrig1           %check if condition is met
    c1 = c1+1;                              %... if not, count forward 1 index per loop iteration
end
peakStart = c1;                             %location in vector where cutting data starts

c2 = length(FxSmooth);                      %initiate counter 2  at end of data set
while xPercent(c2) < percentTrig1           %check if condition is met
    c2 = c2-1;                              %if not, count backward 1 index per loop iteration
end
peakStop = c2;                              %location in vector where cutting data stops

Tpeak = T(peakStart:peakStop);              %vector of times during which Fx has plateaued
Dpeak = D_ACDC(peakStop)-D_ACDC(peakStart); %vector of effective displacements during which Fx
has plateaued
FxPeak = mean(Fx(peakStart:peakStop));      %vector of Fx's during which Fx has plateaued

%% CALCULATE VOLUME
% This volume represents the amount of material removed and will be used to
% normalize energy

vol = (D_DC(cutStop)-D_DC(cutStart))*w*h0;  %[mm^3]

%% DEFINE "CUTTING" VECTORS

% Time
TCut = T(cutStart:cutStop);                 %time vector b/t when cutting starts and stops

% Displacement
D_DCcut = D_DC(cutStart:cutStop);
D_ACcut = D_AC(cutStart:cutStop);
D_ACDCcut = D_ACDC(cutStart:cutStop);

%% LEAST SQUARES FIT
% The purpose of this section is to provide a least-squares
% characterization for the effective displacement (and thus velocity). Such
% a characterization has the advantage of being completely noise free,
% which cannot be said of standard filtering techniques.  This noise-free
% criteria is important, since the force prediction scheme is sensitive to
% noise in the velocity trace.

switch fm


%             > > >   C O N V E N T I O N A L   < < <
    case 0
        f = 0;
            % LEAST SQUARES ON ACDC DISPLACEMENT
            cACDC = [TCut, ones(length(TCut),1)];
            dACDC = D_ACDCcut;
            xACDC = inv(cACDC'*cACDC)*cACDC'*dACDC;
            Aacdc = xACDC(1); Bacdc = xACDC(2);
            % Fitted displacement curve2
            D_ACDCfitBest = Aacdc*TCut + Bacdc*ones(length(TCut),1);
            D_DCfitBest = D_ACDCfitBest;        %same since there's no ac
            D_ACfitBest = zeros(length(TCut),1);


%               > > >   M O D U L A T I O N   < < <
    otherwise
        RSSbest = inf;                      %initialize with worst case residual sum of squares value
        for f = fm-.01:.001:fm+.01;         %scan a range of modulation frequencies near the expected
fm.  This often results in a better R^2 fit


            % LEAST SQUARES ON DC DISPLACEMENT - this is performed despite
            % "DC" classifier because motor has a tendency to periodically
            % backtrack on itself due to modulated loading
            cDC = [sin(2*pi*f*TCut), cos(2*pi*f*TCut), TCut, ones(length(TCut),1)];
            dDC = D_DCcut;
            xDC = inv(cDC'*cDC)*cDC'*dDC;
            Adc = xDC(1); Bdc = xDC(2); Cdc = xDC(3); Ddc = xDC(4);
            % Fitted displacement curve
            D_DCfit = Adc*sin(2*pi*f*TCut) + Bdc*cos(2*pi*f*TCut) + Cdc*TCut +
Ddc*ones(length(TCut),1);


            % LEAST SQUARES ON AC DISPLACEMENT
            cAC = [sin(2*pi*f*TCut), cos(2*pi*f*TCut)];
            dAC = D_ACcut;
            xAC = inv(cAC'*cAC)*cAC'*dAC;
            Aac = xAC(1); Bac = xAC(2);
            % Fitted displacement curve
            D_ACfit = Aac*sin(2*pi*f*TCut) + Bac*cos(2*pi*f*TCut);


            % DETERMINE FIT
            D_ACDCfit = D_DCfit+D_ACfit;
            % Evaluate closeness of fit
            RSS = sum((D_ACDCfit-D_ACDCcut).^2);    %RSS=residual sum of squares
                if RSS<RSSbest                      %if this iteration of loop resulted in lower
RSS, store the following values as "best"
                    RSSbest = RSS;
                    fbest = f;
                    D_DCfitBest = D_DCfit;
                    D_ACfitBest = D_ACfit;
                    D_ACDCfitBest = D_ACDCfit;
                    AacBest = Aac;
                    BacBest = Bac;
                end
        end
end

%% BEST FIT VARIABLES

D_DCfit = D_DCfitBest;
D_ACfit = D_ACfitBest;
D_ACDCfit = D_ACDCfitBest;

%% FIND VELOCITIES
% use standard numerical differentiation method

% DC Velocity
D_DCshift = D_DCfit;                            %intialize shifted displacement vector
D_DCshift(2:end+1) = D_DCshift;                 %shift entries down one row
delD_DC = D_DCfit - D_DCshift(1:end-1);         %D(i)-D(i-1)
delD_DC(1) = 0;                                 %sets first entry to zero
Vdc = delD_DC/dT;                               %velocity of stage wrt table [mm/s]
Vdc(1) = [];                                    %eliminate first entry

% AC Velocity
D_ACshift = D_ACfit;                            %initialize shifted displacement vector
D_ACshift(2:end+1) = D_ACshift;                 %shift entries down one row
delD_AC = D_ACfit - D_ACshift(1:end-1);         %D(i)-D(i-1)
delD_AC(1) = 0;                                 %sets first entry to zero
Vac = delD_AC/dT;                               %velocity of workpice wrt stage [mm/s]
Vac(1) = [];                                    %eliminate first entry

% Effective Velocity (ACDC)
D_ACDCshift = D_ACDCfit;                        %initialize shifted displacement vector
D_ACDCshift(2:end+1) = D_ACDCshift;             %shift entries down one row
delD_ACDC = D_ACDCfit-D_ACDCshift(1:end-1);     %D(i)-D(i-1)
delD_ACDC(1) = 0;                               %sets first entry to zero
Veff = delD_ACDC/dT;                            %velocity of workpice wrt table (or tool) [mm/s]
Veff(1) = [];                                   %eliminate first entry

%% FIND POINTS OF SEPARATION'S BEGINNING/MAXIMUM/ENDING
% Separation occurs when velocity becomes negative.  Max distance of
% separation occurs when velocity becomes positive.  Contact is
% reestablished when displacement is equal to that at which separation
% began.
% Use velocity derived from best fit, since this section is largely
% dependent on finding local mins and maxs, which is only effective in a
% noiseless environment.

% Find indices of local extrema
diffSignV = diff(sign(Veff));
diffSignV(2:end+1) = diffSignV;             %insert a zero as the first entry
locMin = find(diffSignV==2);                %indices of local mins
locMax = find(diffSignV==-2);               %indices of local maxs

% Determine if separation is occuring
if isempty(or(locMin,locMax))
    sepFlag = 0;                            %seperation does not occur
else
    sepFlag = 1;                            %separation occurs
end

% Create flags for each of 4 cases and rewrite D vector's such that
% locMin and locMax have same number of entries, locMax(1)/locMax(end) must come
% before locMin(1)/locMin(end)
if sepFlag == 1     %do this routine only if separation is actually occuring
    if and(locMax(1)<locMin(1),locMax(end)>locMin(end)) %starts on MAX, ends on MAX
        extFlag = 1;
        locMax(end) = '';
    elseif and(locMax(1)<locMin(1),locMax(end)<locMin(end)) %starts on MAX, ends on MIN
        extFlag = 2;
    elseif and(locMax(1)>locMin(1),locMax(end)>locMin(end)) %starts on MIN, ends on MAX
        extFlag = 3;
        locMin(1) = '';
        locMax(end) = '';
    elseif and(locMax(1)>locMin(1),locMax(end)<locMin(end)) %starts on MIN, ends on MIN
        extFlag = 4;
        locMin(1) = '';
    end

    % Displacements at local extrema
    D_fitMin = D_ACDCfit(locMin);
    D_fitMax = D_ACDCfit(locMax);

    % Points of separations' BEGINNING
    T_sepBeg = TCut(locMax);

    % Point of separations' MAXIMUM
    T_sepMax = TCut(locMin);

    % Values of maximum separation
    D_sepMax = D_fitMax - D_fitMin;

    % Points of separations' ENDING
    T_sepEnd = 0*T_sepBeg;      %initialize
    for k = 1:length(locMin)
        c3 = locMax(k)+1;      %initialize counter c3
        while D_ACDCfit(c3)<D_fitMax(k)
            c3 = c3+1;
        end
        locRecut(k) = c3;
        T_sepEnd(k) = TCut(c3);
    end
end

%% ELIMINATE VELOCITIES WHERE SEPARATION OCCURS
% Set these velocities equal to zero since they won't contribute to energy
% consumption

Veff2 = Veff;                                   %initialize new variable
if sepFlag==1
    for kk = length(locMax):-1:1
        Veff2(locMax(kk)+1:locRecut(kk)-1)=0;   %eliminate (use 1e-5 to approx 0) all noncutting
portions of Veff trace (must start at end of trace)
    end
end
```

```
%% ALGORITHM 2: DETERMINE RANGE OF DISCRETE "RELEVANT" DATA
% Algorithm 2: "Relevant" = when tool is engaged.  Thus this range will
% consist of multiple discrete instances of "relevant" data.
% Note: This range is determined using ch1 because ch1 responds
% "immdeiately" when the tool is engaged

percentTrigger3 = 0.7;
i = 1;                                  %initiate counter Nrel2
iMinus = 0;                             %after the outer while loop, we'll subtract this from i
c1 = cutStart;                          %initiate counter c1

switch f                                %this switch-case generates cPeriod, depending on the
piezo's input frequency
    case 0
        cPeriod = cutStop-cutStart;     %a zero freq case needs to exist to prevent Inf values
for cPeriod
    otherwise
        cPeriod = (1/fm)*(1/dT);        %number of steps in one period of piezo cycle based on
fft frequency
end

while c1<cutStop                        %stops loop when it reaches end of data set

        % Find Leading Edge > > > > > > > > > > >
        while and(xPercent(c1) < percentTrigger3,c1<cutStop)  %searches for leading edge by
looking for percent change of > percentTrigger2
            c1 = c1+1;                  %note: instance wont occur when loop is broken
        end
        relDataSTART2(i) = c1;          %location in vector where relevant data starts
        flag1 = 1;
        % > > > > > > > > >  > > > > > > > > > > >


    if c1<cutStop   %do this if we didn't just reach the end of the data set by looking for
another rising edge
        flag1 = 2;
        % Predict Falling Edge - - - - - - - - -
        switch f
            case 0      %if 0 frequency case, skip 3/4 of steps in between Alg 1's range
                cToSkip = floor((3/4)*(cutStop-cutStart));    %number of steps to skip before
searching for falling edge
                c1 = c1+cToSkip;                    %skip the cycles
            otherwise   %if not 0 frequency case, calculate and skip percentage of period, based
on user input as to approx how much of period has cutting ("percentHigh")
                cToSkip1 = floor(.3*cPeriod);       %number of steps to skip before searching for
falling edge
                c1 = c1+cToSkip1;                   %skip the cycles to predict falling edge
                if c1 >= cutStop                    %...but  if skipping these cycles puts us
past end of Rel1 set...
                    relDataSTART2(i) = '';          %...delete last instance of relDataSTART2
because Predict Falling Edge counted up to cutStop
                    iMinus = 1;                     %after the outer while loop, we'll subtract
this from i because we have a false positive in relDataSTART2
                end
        end
        % - - - - - - - - - - - - - - - - - - - -
    else %this "else" only occurs if we've reached the end of the data set while looking for the
next leading edge
        relDataSTART2(i) = '';                      %if Find Leading Edge counted up to cutStop
        iMinus = 1;                                 %after the outer while loop, we'll subtract
this from i
    end


    if c1<cutStop   %do this if we didn't just reach the end of the data set by predicting the
next falling edge
        % Find Falling Edge o o o o o o o o o o o
        while and(xPercent(c1) >= percentTrigger3 , c1<cutStop) %search for falling edge
            c1 = c1+1;                              %note: instance wont occur when loop is
broken
        end
        relDataSTOP2(i) = c1-1;                     %location in vector where relevant data stops
        flag1 = 3;
        % o o o o o o o o o o o o o o o o o o o o


        % Predict Leading Edge = = = = = = = = =
        switch f
            case 0  %do nothing since we're already at end of set

            otherwise
                if (relDataSTOP2(i)-relDataSTART2(i)) < cPeriod  %this if command prevents
cToSkip from becoming negative, which occurs if on/off triggers do not occur every period
                    cRemainInPeriod(i) = cPeriod - (relDataSTOP2(i)-relDataSTART2(i));
%number of cycles remaining in current period
                    cToSkip2 = floor((.5)*cRemainInPeriod(i));   %number of cycles to skip before
searching for next rising edge
                    c1 = c1+cToSkip2;               %skip the cycles
                end
        end
        % = = = = = = = = = = = = = = = = = = = =
        if c1<cutStop %if c1 is STILL less than relDataSTOP (ie we didn't just reach the end of
the relevant Alg 1 data by predicting next leading edge
        i = i+1;
        end
    end
end
Nrel2 = i-iMinus;                                   %subtract 1 (via iMinus) ONLY if there's a
false positive from while loop

if length(relDataSTART2)==length(relDataSTOP2)+1
   relDataSTART2(end) = '';
   Nrel2 = Nrel2-1;
end
% [] [] [] [] [] [] [] [] [] [] [] [] [] [] [] [] [] []

%% PLATEAU FORCES
% Plateau forces are designated as the average value over the discrete
% range where the Fx value has approximately saturated

switch f
    case 0      %conventional cutting
        for ii=1:Nrel2
           FxPlat(ii) = mean(Fx(relDataSTART2(ii):relDataSTOP2(ii)));
           FyPlat(ii) = mean(Fy(relDataSTART2(ii):relDataSTOP2(ii)));
        end
    otherwise   %modulated cutting
        for ii=2:Nrel2
           FxPlat(ii) = mean(Fx(relDataSTART2(ii):relDataSTOP2(ii)));
           FyPlat(ii) = mean(Fy(relDataSTART2(ii):relDataSTOP2(ii)));
        end

end

%% DETERMINE MAM CONDITIONS USING Fx
% Note: This should be done under conditions of high trigger start and high
% length of analysis

if fm~=0
   % Smooth Fx trace
   FxRelSmooth = smooth(Fx(cutStart:cutStop),NperPeriod/100);

   % Determine No. of periods
   Nperiods = floor(length(TCut)/NperPeriod);

   % Find max and min Fx for each period
   FxMaxs = zeros(Nperiods,1);  %initialize
   FxMins = zeros(Nperiods,1);  %initialize
   for k = 1:Nperiods
       FxMaxs(k) = max(FxRelSmooth(((k-1)*NperPeriod)+1:k*NperPeriod));
       FxMins(k) = min(FxRelSmooth(((k-1)*NperPeriod)+1:k*NperPeriod));
   end
   FxMinsPos = FxMins;
   FxMinsPos(FxMins<0)=0;          %set all negative values to zero

   % Determine percentage of critical MAM, where 0% is conventional cutting
   % and 100% represents a MAM condition in which Fx just touches 0N each
   % period
   critMAM = mean((FxMaxs-FxMinsPos)./FxMaxs)*100;

end

%% PREDICT ENERGY
% Use forceFitPlaner.m to determine function and its best fit linear
% coefficients. It was found that there exists a natural logarithmic
% relationship between Fx and V.  Since P = F*V, the best fit curve
% assumption is a combination of logarithmic and linear (since V-V is
% exactly linear).  Thus, P = A*ln(x)*x + B*x where A and B are constants
% found using the linear solver best-fit method

coeff = [5.4282e-01 2.1529e+01];                                    %coefficients from
forceFitPlaner.m

Pest = coeff(1)*log(Veff2).*(Veff2/1000) + coeff(2)*(Veff2/1000);   %[W]
Pest(Pest<0)=0;                                                     %effectively eliminates
singularity problem caused by log(Veff2)

if sepFlag==1
    Pest(Veff2==0)=0;                                               %power should = 0 when tool
has separated from workpiece
end

dWest = Pest*dT;
    dWest(1) = dWest(1)*.5;                                         %technicality of first
instance lasting only half a time step
    dWest(end) = dWest(end)*.5;                                     %technicality of last
instance lasting only half a time step
West = sum(dWest)/vol;

%% CALCULATE ENERGY

% Instantaneous work
dW = Fx(cutStart:cutStop).*delD_ACDC/1000;  %dW(i)=F(i)*[-(D(i)-D(i-1))] ONLY CUTTING PORTION
CONSIDERED [J]

% Power
P = dW/dT;                                  %[W]

% All work terms
Wall = sum(dW)/vol;                         %summation of all work terms [J/mm^3]

% Only positive work terms
dWpos = dW;
dWpos(dWpos<0)=0;
Wpos = sum(dWpos)/vol;                      %summation of only positive work terms [J/mm^3]

% Only work terms within separation boxes
if sepFlag == 1     %do this routine only if separation is actually occuring
    dWsep = 0*dW;       %initialize
    for k = 1:length(locMin)
       dWsep(locMax(k):locRecut(k)) = dW(locMax(k):locRecut(k));
    end
    Wsep = sum(dWsep);                      %[J]
end

%% PLOT DISPLACEMENT

% Initialize figure
hFig1 = figure(1);
fullscreen = get(0,'ScreenSize');
set(hFig1,'position',[10, 50, floor(fullscreen(3)/2)-30, fullscreen(4)-132])
set(hFig1,'color','w','Name','Analysis')
```

```
% Plot displacement
ax1 = subplot(4,1,1);
hold on
plot(T,D_ACDC,'k','linewidth',2.5)
plot(TCut,D_ACDCfit,'g','linewidth',1)
hold off

% Format figure
set(gca,'fontsize',18)
ylabel('Displacement (mm)')
legend('Actual','Best Fit')
hTtl = title(strrep(filename,'_','\_'));
set(hTtl,'fontsize',22)

grid on

%% PLOT SEPARATION BOXES
% Each separation box has width equal to amount of time tool was separated
% from workpiece and height equal to max distance of separation between
% tool and workpiece (NOT ACCOUNTING FOR COMPLIANCE)

if sepFlag == 1     %do this routine only if separation is actually occuring
    hold on
    for k = 1:length(locMin)
        rectangle('position',[T_sepBeg(k),D_fitMin(k),T_sepEnd(k)-
T_sepBeg(k),D_sepMax(k)],'edgecolor',[.25 .5 .7],'linewidth',2)
    end
    hold off
end

%% PLOT VELOCITIES

% Plot velocity
ax2 = subplot(4,1,2);
hold on
plot(TCut(2:end),Vdc,'-r')
plot(TCut(2:end),Vac,'--','color',[0 .6 .3])
plot(TCut(2:end),Veff,'b','linewidth',2)

% Format figure
set(gca,'fontSize',18)
ylabel('Velocity (mm/s)')
legend('Vdc','Vac','Veff')
grid on

%% PLOT FORCES

% Plot forces
ax3 = subplot(4,1,3);
hp1 = plot(T,Fx,'b');
hold on
hp2 = plot(T,Fy,'m');

plot([T(cutStart),T(cutStart)],[min(Fx),max(Fx)],':g','linewidth',2.5)
plot([T(cutStop),T(cutStop)],[min(Fx),max(Fx)],':r','linewidth',2.5)
plot([T(peakStart),T(peakStart)],[min(Fx),max(Fx)],'--g','linewidth',2.5)
plot([T(peakStop),T(peakStop)],[min(Fx),max(Fx)],'--r','linewidth',2.5)

plot([T(relDataSTART2)],0,'.g','markersize',18)
plot([T(relDataSTOP2)],0,'xr','markersize',9,'linewidth',2.5)

for jj = 1:Nrel2
   hp3 = plot([T(relDataSTART2(jj)),T(relDataSTOP2(jj))],[FxPlat(jj),FxPlat(jj)],'--
b','linewidth',2.5);
   hp4 = plot([T(relDataSTART2(jj)),T(relDataSTOP2(jj))],[FyPlat(jj),FyPlat(jj)],'--
m','linewidth',2.5);
end

% Format figure
hold off
```

```
set(gca,'fontSize',18)
legend([hp1 hp2 hp3 hp4],'Fp','Fq','Fp Plat','Fq Plat')
ylabel('Force (N)')
grid on

%% PLOT POWER

% Plot power
ax4 = subplot(4,1,4);
hold on
% plot(TCut(2:end),Pest,'r','linewidth',3)
plot(TCut,P,'k','linewidth',1)
hold off

% Format figure
set(gca,'fontSize',18)
xlabel('Time (s)')
ylabel('Power (W)')
% legend('Predicted','Actual')
grid on

%% LINK X-AXES

linkaxes([ax1,ax2,ax3,ax4],'x')
xlim([0,T(end)])

figure(1)

%% PRINT TO WORKSPACE

disp(' ')
disp(' ')
disp('C O N D I T I O N')
if fm==0
    disp('Conventional')
else
    disp(sprintf('  %3.0f%% of critical MAM',critMAM))
end
disp(' ')
disp(' ')

disp('F O R C E')
disp(sprintf('   Avg Fp             = %.2f N',FxRel))
if fm==0
    disp(sprintf('   Avg Plateau Fp     = %.2f +/- %.2e N',mean(FxPlat),std(FxPlat)))
    disp(sprintf('   Avg Plateau Fq     = %.2f +/- %.2e N',mean(FyPlat),std(FyPlat)))
else
    disp(sprintf('   Avg Plateau Fp     = %.2f +/- %.2e 
N',mean(FxPlat(2:end)),std(FxPlat(2:end))))
    disp(sprintf('   Avg Plateau Fq     = %.2f +/- %.2e 
N',mean(FyPlat(2:end)),std(FyPlat(2:end))))
end
disp(' ')
disp(' ')

disp('P O W E R')
disp(sprintf('   Avg Power          = %.2e J/s',mean(P)))
disp(' ')
disp(' ')

disp('V O L  S P E C  E N E R G Y ')

disp(sprintf('   Total summed       = %.4e J/mm^3',Wall))
disp(sprintf('   Predicted          = %.4e J/mm^3',West))

disp(' ')
disp(' ')

if fm~=0
disp('M A M  C O N D I T I O N S')
disp(sprintf('   Best fm            = %.4f Hz',fbest))
disp(sprintf('   Displacement Amp K = %.2f um',norm([AacBest,BacBest],2)*1000))
end

FxPlat1 = Fx(relDataSTART2:relDataSTOP2);
FyPlat1 = Fy(relDataSTART2:relDataSTOP2);
```

Appendix H: Lathe analysis routine (Matlab)

Summary: This routine imports lathe-specific data (Fx,Fy,Fz,D_{AC}) from a user-specified .txt file and performs numerical analysis techniques. Primary features include:

- determination of beginning and end of cut
- determination of beginning and end of individual periodic cutting instances
- determination of effective instantaneous displacement and velocity (assumed linear motion of turret is combined with data from capacitance probe)
- determination of best-fit equation characterizing said displacement
- determination of instantaneous undeformed chip thickness
- force prediction routine based on Fp(h0) found using forceFitLathe.m
- specific energy calculations

Note: This code refers to Fp/Fq as Fy/Fz, respectively.

```
%% MAManalysisLathe.m
%
% Written By: Josh Norman
% PROGRAM SUMMARY: THIS ROUTINE IMPORTS LATHE-SPECIFIC DATA FROM A USER
% SPECIFIED .TXT FILE AND PERFORMS NUMERICAL ANALYSIS. PRIMARY FEATURES
% INCLUDE:
%       DETERMINATION OF BEGINNING AND END OF CUT.
%       DETERMINATION OF BEGINNING AND END OF PERIODIC CUTTING INSTANCES.
%       DETERMINATION OF EFFECTIVE INSTANTANEOUS DISPLACMENT AND VELOCITY
%           (ASSUMED LINEAR MOTION OF TURRET IS COMBINED WITH DATA FROM
%           CAPACITANCE PROBE) AS WELL AS BEST FIT EQUATION CHARACTERIZING
%           THIS DISPLACEMENT.
%       DETERMINATION OF INSTANTANEOUS UNDEFORMED CHIP THICKNESS
%       FORCE PREDICTION ROUTINE BASED ON Fp(h0) FOUND USING
%           forceFitLathe.m.
%       SPECIFIC ENERGY CALCULATIONS.
%       NOTE: Fp/Fq ARE REFERRED TO AS Fy/Fz RESPECTIVELY.

clc; clear all; close all;

%% INPUTS

num_headerlines = 1;%number of headerlines
fs = 5000;          %sampling frequency [Hz]
percentHigh = 0.40; %approx % of cycle in which cutting occurs
w = 1200;           %angular velocity [rpm]
do = 25.4;          %workpiece outer diameter [mm]
wallThick = 1.24;   %workpiece wall thickness [mm]
NperVx = 10;        %Newtons per Volt in x dir [N/V]
NperVy = 20;        %Newtons per Volt in y dir [N/V]
NperVz = 20;        %Newtons per Volt in z dir [N/V]
uperV = 25;         %microns per Volt in displacement [um/V]
driftZComp = 1;     %compensate for drift? (y/n = 1/0)
showdriftZComp = 0; %show drift comp?      (y/n = 1/0)
SScase = 3;         %which freq to use for Single Sine...
                    %...  (1/2/3 = input/fft/modfft)
```

```
percentTrigger1 = 0.2; %trigger percentage for determining range of parent relevant data set

%% CHOOSE TXT FILE TO IMPORT

% Select "Relevant" Files
wildcard_txt = '*.txt';
files_struct = dir(wildcard_txt);                   %list all .txt files in structure
files_cell = struct2cell(files_struct);             %list these files in cell
names_cell = files_cell(1,:);                       %select first row of cell b/c it contains
names
names_char = char(names_cell);                      %convert this cell row of names to characters
[m,n] = size(names_char);                           %use m to know how many files are available

if m==0
   error('NO TXT FILES ARE LOCATED IN CURRENT DIRECTORY')

elseif m<10     %single digits
   for i = 1:m
        FileOptions(i,:) = sprintf('%i) %s',i,names_char(i,:));     %create character array of
file options to choose from
   end

elseif m>=10    %double digits
    for i = 1:9
        FileOptions(i,:) = sprintf('% i) %s',i,names_char(i,:));    %create character array of
file options to choose from
    end
    for i = 10:m
        FileOptions(i,:) = sprintf('%i) %s',i,names_char(i,:));     %create character array of
file options to choose from
    end
end

% Print Filenames to Workspace
disp(' ')
fprintf('\t\t\tANALYSIS OF EXPERIMENTAL DATA')
fprintf('\n\n')
disp(FileOptions)                                                   %print list to workspace
fprintf('\n\n')

% Choose file on which to run analysis
fileChoice = input('Enter File #   -->  ');                         %user enters number
corresponding to file
filename_ext = deblank(names_char(fileChoice,:));                   %remove placeholding spaces
from end of selected filename
[PATHSTR, filename, EXT] = fileparts(filename_ext);

%% IMPORT DATA FROM .txt FILE

fid = fopen(filename_ext);                                          %open file
Acell = textscan(fid,'%f %f %f %f','headerlines',num_headerlines);  %import data from all 4
columns of file to cell variable
Amat = cell2mat(Acell);                                             %convert cell variable to
matrix
fclose(fid);                                                        %close file

% Assigning Data Channels
ch0 = Amat(:,1);                                                    %first column
ch1 = Amat(:,2);                                                    %second column
ch2 = Amat(:,3);                                                    %third column
ch3 = Amat(:,4);                                                    %fourth column

%                    F O R C E   C A L C U L A T I O N

Fx = ch0*NperVx;                                                    %Force in x-direction [N]
Fy = ch1*NperVy;                                                    %Force in y-direction [N]
FzIni = ch2*NperVz;                                                 %Force in z-direction [N]

%              D I S P L A C E M E N T   C A L C U L A T I O N

%Displacement Calculation
```

```
D_AC = ch3*uperV;                                                   %Capacitive (ie dynamic)
displacement [um]

%% PRE-PROCESSING

%Determine Modulation Frequency - - - - -
locHz = strfind(filename_ext,'Hz');     %find index
loc_ = strfind(filename_ext,'_');       %find index
HzValStart = 1;
for i = 1:length(loc_)
    if loc_(i)<locHz;
        HzValStart = loc_(i)+1;         %by end of outer for loop, HzValStart will be the index
of the space where the piezo freq value begins
    end
end
HzValStop = locHz-1;                    %index of the space where the piezo freq value stops
fStr = filename_ext(HzValStart:HzValStop);
f = str2num(fStr);                      %piezo frequency [Hz]
% - - - - - - - - - - - - - - - - - - - - -

% Determine feedrate & & & & & & & & & &
locFR = strfind(filename_ext,'mmrev');
FR = (str2num(filename_ext(locFR-5)) + 1/1000*str2num(filename_ext(locFR-3:locFR-1)))*1000;
%turret feedrate (um/rev)
% & & & & & & & & & & & & & & & & & & & &

%Constants + + + + + + + + + + + + + +
N = length(ch0);
dT = 1/fs;                              %time step [s]
t = (0:1:N-1)'*dT;                      %time vector [s]
di = do - 2*wallThick;                  %inner diameter of tube [mm]
diaCut = mean([di,do]);                 %average diameter [mm]
wSI = w*2*pi/60;                        %angular velocity of workpiece in SI [rad/s]
V = (pi/(60*1000)) * w * diaCut;        %workpiece velocity at point of contact [m/s]
distCut = V.*t;                         %distance tool has cut as function of time [m]
Pspin = 60/w;                           %period for one revolution of spindle [s]
stepsPerRot = round(fs*Pspin);          %number of timesteps in one revolution of spindle
V_DC = FR*w/60;                         %turret velocity [um/s]

chCurr = ch1;
chCurrSmooth = smooth(chCurr,10);                   %smooth out chCurr data
yRange = max(chCurrSmooth)-min(chCurrSmooth);       %range of values on current channel
yPercent = (chCurrSmooth-min(chCurrSmooth))/yRange; %vector of channels values as percent of
range
% + + + + + + + + + + + + + + + + + + +

%% ALGORITHM 1: DETERMINE RANGE OF PARENT "RELEVANT" DATA
% Algorithm 1: "Relevant" = range between when tool is first engaged until
% it is last disengaged.

% START/STOP ALGORITHM 1 - - - - - - - - - - - -
c1 = 1;                                     %initiate counter 1
while yPercent(c1) < percentTrigger1        %looks for percent change of > percentTrigger1
    c1 = c1+1;                              %note: instance wont occur when loop is broken
end
relDataSTART1 = c1;                         %location in vector where relevant data starts
c2 = length(chCurr);                        %initiate counter 2 where at end of data set
while yPercent(c2) < percentTrigger1
    c2 = c2-1;
end
relDataSTOP1 = c2;                          %location in vector where relevant data stops

lengthRel1 = relDataSTOP1-relDataSTART1+1;  %length of parent data set
%  - - - - - - - - - - - - - - - - - - - - - - -

    % Drift Compensation in Fz * * * * * * * * * * *
    switch driftZComp       %Use a switch to easily turn on/off drift compensation
        case 0              %NO drift compensation on force in z-dir
            Fz = FzIni;     %leave Fz as it was initially
        case 1              %WITH drift compensation on force in z-dir
```

```
            distCutNonRel = distCut([1:relDataSTART1,relDataSTOP1:length(distCut)],1);  %vector
of the non-relevant distances cut
            FzNonRel = FzIni([1:relDataSTART1,relDataSTOP1:length(FzIni)],1);           %vector
of the non-relevant Fz's
            driftLineCoeffs = polyfit(distCutNonRel,FzNonRel,1);
            FzDriftLine = polyval(driftLineCoeffs,distCut);
            Fz = FzIni-FzDriftLine;
    end
    % * * * * * * * * * * * * * * * * * * * * * * * *

% Define vector of times during which cutting occurs
tCut = t(relDataSTART1:relDataSTOP1);

%% PERFORM FFT ON PARENT DISPLACEMENT DATA

displRelev1 = D_AC([relDataSTART1:relDataSTOP1],1);    %[um]
L = length(displRelev1);                                                %number of samples
fNyq = fs/2;                                                            %Nyquist frequency [Hz]
Nfft = 2^(nextpow2(L)-1);                                               %length of fft by taking
largest possible power of 2 from length of ch3
fft_freq = fNyq*linspace(0,1,Nfft/2)';                                  %fft frequencies taking
into account frequency resolution (single sided)
fft_displRelev = fft(displRelev1,Nfft);                                 %perform FFT
fft_displRelevAdj = fft_displRelev/L;                                   %scale FFT by dividing by
L (# of samples)
fft_displRelevAdjSingSided = 2*abs(fft_displRelevAdj(1:Nfft/2));        %single sided analysis of
fft results
fft_freqNoDCBinIndex = fft_freq > 0.5*f;                                %returns logical true for
all frequencies that are greater than 50% of input frequencies.  These frequencies will be
skipped when looking for max freq in order to eliminate DC component [Hz]
fft_freqNoDCIndex = find(fft_freq > 0.5*f);                             %returns index of all
frequencies that are greater than 50% of input frequencies
fft_displRelevAdjSingSidedNoDC = fft_freqNoDCBinIndex.*fft_displRelevAdjSingSided; %set power (y-
axis) of all frequencies lower than above criteria == 0 in order to filter out DC component

%% DETERMINE FREQUENCY

freqFFTIndex = find(fft_displRelevAdjSingSidedNoDC==max(fft_displRelevAdjSingSidedNoDC));
%location on freq domain where max occurs
if length(freqFFTIndex)>1               %if multiple frequencies appear as the primary
frequency...
    freqFFTIndex = min(freqFFTIndex);   %...select the lowest of these frequencies
    warning('MULTIPLE PRIMARY FREQUENCIES DETECTED')
end
freqFFT = fft_freq(freqFFTIndex);       %primary frequency of oscillation [Hz]
freqFFTdist = 1 / ((1/(2*pi*freqFFT))*wSI*(diaCut/1000/2));  %primary frequency of oscillation in
distance cut metric [rad/m]

%% DETERMINE WHICH CUTTING TYPE IS OCCURING
% Case 1: Used when either f = 0 or else when the cutting tool loses
% contact during machining
% Case 2: Used when the cutting tool oscillates, but maintains contact

xStartTest = round(0.45*(relDataSTOP1 - relDataSTART1) + relDataSTART1);    %start of range near
middle of data set
xStopTest  = round(0.55*(relDataSTOP1 - relDataSTART1) + relDataSTART1);    %end of range near
middle of data set
CutTypeCaseTestRange = Fy(xStartTest:xStopTest);                            %Fy values within
this range
stdTestRange = std(CutTypeCaseTestRange);                                   %calculates standard
deviation of the selected force data in order to determine if oscillation occurs, but at constant
so (constant thickness)

if or(or(f==0,min(yPercent(xStartTest:xStopTest)) <= 0.1),stdTestRange < 1)
    CutTypeCase = 1;
elseif min(yPercent(xStartTest:xStopTest)) > 0.1
    CutTypeCase = 2;
else
    error('Unrecognizable cutting type')
end
```

```
%% DETERMINE RANGE OF REPRESENTATIVE CONVENTIONAL DATA
% Eliminate loading/unloading portions of traces

Dskip = 100;                        %distance to eliminate on either side of data [um]
Nskip = round(Dskip/(V_DC*dT));     %number of points skipped
repStart = relDataSTART1+Nskip;     %start index
repStop  = relDataSTOP1-Nskip;      %stop index
FpRep = Fy(repStart:repStop);       %Fp data in this representative range [N]
FqRep = Fz(repStart:repStop);       %Fq data in this representative range [N]
FpAvgRep = mean(FpRep);             %average Fp [N]

%% ALGORITHM 2: DETERMINE RANGE OF DISCRETE "RELEVANT" DATA
% Algorithm 2: "Relevant" = when tool is engaged.  Thus this range will
% consist of multiple discrete instances of "relevant" data.
% Note: This range is determined using ch1 because ch1 responds
% "immediately" when the tool is engaged

switch CutTypeCase

% START/STOP ALGORITHM 2 FOR CASE 1 [] [] [] [] [] [] []
case 1
percentTrigger2 = 0.3;
i = 1;                                      %initiate counter Nrel2
iMinus = 0;                                 %after the outer while loop, we'll subtract this from
i
c1 = repStart;                              %initiate counter c1

switch f  %this switch-case generates cPeriod, depending on the piezo's input frequency
    case 0
        cPeriod = repStop-repStart;         %a zero freq case needs to exist to prevent Inf
values for cPeriod
    otherwise
        cPeriod = (1/freqFFT)*(1/dT);       %number of steps in one period of piezo cycle based
on fft frequency
end

while c1<repStop                            %stops loop when it reaches end of data set

        % Find Leading Edge > > > > > > > > > > >
        while and(yPercent(c1) < percentTrigger2,c1<repStop)  %searches for leading edge by
looking for percent change of > percentTrigger2
            c1 = c1+1;                      %note: instance wont occur when loop is broken
        end
        relDataSTART2(i) = c1;              %location in vector where relevant data starts
        flag1 = 1;                          %used for debugging
        % > > > > > > > > >  > > > > > > > > > > >


    if c1<repStop                           %do this if we didn't just reach the end of the data
set by looking for another rising edge
        flag1 = 2;                          %used for debugging
        % Predict Falling Edge - - - - - - - - -
        switch f
            case 0                          %if 0 frequency case, skip 3/4 of steps in between
Alg 1's range
                cToSkip = floor((3/4)*(repStop-repStart));    %number of steps to skip before
searching for falling edge
                c1 = c1+cToSkip;            %skip the cycles
            otherwise                       %if not 0 frequency case, calculate and skip
percentage of period, based on user input as to approx how much of period has cutting
("percentHigh")
                cToSkip1 = floor((percentHigh-.10)*cPeriod);    %number of steps to skip before
searching for falling edge
                c1 = c1+cToSkip1;           %skip the cycles to predict falling edge
                if c1 >= repStop            %...but  if skipping these cycles puts us past end of
Rel1 set...
                    relDataSTART2(i) = '';  %...delete last instance of relDataSTART2 because
Predict Falling Edge counted up to repStop
                    iMinus = 1;             %after the outer while loop, we'll subtract this from
i because we have a false positive in relDataSTART2
                end
        end
        % - - - - - - - - - - - - - - - - - - - -
    else                                    %this "else" only occurs if we've reached the end of
the data set while looking for the next leading edge
        relDataSTART2(i) = '';              %if Find Leading Edge counted up to repStop
        iMinus = 1;                         %after the outer while loop, we'll subtract this from
i
    end


    if c1<repStop                           %do this if we didn't just reach the end of the data
set by predicting the next falling edge
        % Find Falling Edge o o o o o o o o o o o
        while and( yPercent(c1) >= percentTrigger2 , c1<repStop ) %search for falling edge
            c1 = c1+1;                      %note: instance wont occur when loop is broken
        end

        relDataSTOP2(i) = c1-1;             %location in vector where relevant data stops
        flag1 = 3;                          %used for debugging
        % o o o o o o o o o o o o o o o o o o o o


        % Predict Leading Edge = = = = = = = = =
        switch f
            case 0                          %do nothing since we're already at end of set

            otherwise
                if (relDataSTOP2(i)-relDataSTART2(i)) < cPeriod     %this if command prevents
cToSkip from becoming negative, which occurs if on/off triggers do not occur every period
                    cRemainInPeriod(i) = cPeriod - (relDataSTOP2(i)-relDataSTART2(i));
%number of cycles remaining in current period
                    cToSkip2 = floor((1/2)*cRemainInPeriod(i));     %number of cycles to skip
before searching for next rising edge
                    c1 = c1+cToSkip2;       %skip the cycles
                end
        end
        % = = = = = = = = = = = = = = = = = = = =
        if c1<repStop                       %if c1 is STILL less than relDataSTOP (ie we didn't
just reach the end of the relevant Alg 1 data by predicting next leading edge
        i = i+1;
        end
    end
end
Nrel2 = i-iMinus;                           %subtract 1 (via iMinus) ONLY if there's a false
positive from while loop
% [] [] [] [] [] [] [] [] [] [] [] [] [] [] [] [] [] []

% START/STOP ALGORITHM 2 FOR CASE 2 <> <> <> <> <> <> <>
case 2
i = 1;                                      %initiate counter Nrel2
iMinus = 0;                                 %after the outer while loop, we'll subtract this from
i
c1 = repStart;                              %initiate counter c1.  This will count through each
of the points in the data set
percentTrigger2 = 0.3;                      %percent trigger for relevant data flagging for Alg2

%Find valueTrigger2: Use Fy data for triggering/flagging
switch f                                    %this switch-case generates cPeriod, depending on the
piezo's input frequency
    case 0
        cPeriod = repStop-repStart;         %a zero freq case needs to exist to prevent Inf
values for cPeriod
    otherwise
        cPeriod = (1/f)*(1/dT);             %number of steps in one period of piezo cycle
end

minRel2(i) = min(Fy(c1:c1+ceil(cPeriod)));
maxRel2(i) = max(Fy(c1:c1+ceil(cPeriod)));
rangeRel2(i) = maxRel2(i) - minRel2(i);
valueTrigger2(i) = minRel2(i) + percentTrigger2*rangeRel2(i); %value for relevant data flagging
for Alg2
```

```
while c1<repStop                            %stops loop when it reaches end of Alg1's relevant
data

        % Find Leading Edge > > > > > > > > > > >
        while and(Fy(c1) < valueTrigger2(i),c1<repStop)  %searches for leading edge by looking
for percent change of > percentTrigger
            c1 = c1+1;                      %note: instance wont occur when loop is broken
        end
        relDataSTART2(i) = c1;              %location in vector where relevant data starts
        flag1 = 1;                          %used for debugging
        % > > > > > > > > >  > > > > > > > > > > >


    if c1<repStop                           %do this if we didn't just reach the end of the
relevant data according to Alg1 by looking for another leading edge
        flag1 = 2;                          %used for debugging
        % Predict Falling Edge - - - - - - - - -
        cPeriod = (1/f)*(1/dT);             %number of steps in one period of piezo cycle
        switch f
            case 0                          %if 0 frequency case, skip 3/4 of steps in between
Alg 1's range
                cToSkip = floor((3/4)*(repStop-repStart));    %number of steps to skip before
searching for falling edge
                c1 = c1+cToSkip;            %skip the cycles
            otherwise                       %if not 0 frequency case, calculate and skip 3/4 of
steps for 1 piezo cycle...
                cToSkip1 = floor((1/2)*cPeriod);    %number of steps to skip before searching for
falling edge
                c1 = c1+cToSkip1;           %skip the cycles to predict falling edge
                if c1 >= repStop            %...but  if skipping these cycles puts us past end of
Rel1 set...
                    relDataSTART2(i) = '';  %...delete last instance of relDataSTART2 because
Predict Falling Edge counted up to repStop
                    iMinus = 1;             %after the outer while loop, we'll subtract this from
i because we have a false positive in relDataSTART2
                end
        end

        % - - - - - - - - - - - - - - - - - - - -
    else                                    %this "else" only occurs if we've reached the end of
the data set while looking for the next leading edge
        relDataSTART2(i) = '';              %if Find Leading Edge counted up to relDataSTOP2
        iMinus = 1;                         %after the outer while loop, we'll subtract this from
i because we have a false positive in relDataSTART2
    end


    if c1<repStop                           %do this if we didn't just reach the end of the
relevant data according to Alg1 by predicting the next falling edge
        % Find Falling Edge o o o o o o o o o o o
        while and( Fy(c1) >= valueTrigger2(i) , c1<repStop )   %search for falling edge
            c1 = c1+1;                      %note: instance wont occur when loop is broken
        end

        relDataSTOP2(i) = c1-1;             %location in vector where relevant data stops
        flag1 = 3;                          %used for debugging
        % o o o o o o o o o o o o o o o o o o o o


        % Predict Leading Edge = = = = = = = = =
        switch f
            case 0                          %do nothing since we're already at end of set

            otherwise
                if (relDataSTOP2(i)-relDataSTART2(i)) < cPeriod     %this "if" command prevents
cToSkip from becoming negative, which occurs if on/off triggers do not occur every period
                    cRemainInPeriod(i) = cPeriod - (relDataSTOP2(i)-relDataSTART2(i));
%number of cycles remaining in current period
                    cToSkip2 = floor((1/2)*cRemainInPeriod(i));     %number of cycles to skip
before searching for next rising edge
```

```
                    c1 = c1+cToSkip2;       %skip the cycles
                end
        end
        flag1 = 4;                          %used for debugging
        % = = = = = = = = = = = = = = = = = = = =
        minRel2(i+1) = min(Fy(relDataSTART2(i):relDataSTOP2(i)));
        maxRel2(i+1) = max(Fy(relDataSTART2(i):relDataSTOP2(i)));
        rangeRel2(i+1) = maxRel2(i+1) - minRel2(i+1);
        valueTrigger2(i+1) = minRel2(i+1) + percentTrigger2*rangeRel2(i+1); %value for relevant
data flagging for Alg2
        if c1<repStop                       %if c1 is STILL less than relDataSTOP (ie we didn't
just reach the end of the relevant Alg 1 data by predicting next leading edge
            i = i+1;
        end
    end
end
Nrel2 = i-iMinus;                           %subtract 1 (via iMinus) ONLY if there's a false
positive from while loop
% <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

end %of SWITCH on CutTypeCase

%% PLATEAU FINDING ALGORITHM

for c2 = 1:Nrel2    %do this for each chunk of relevant data according to Algorithm 2 (discrete)

    % Plateau Finding Algorithm 1 * * * * * * * * * * *
    percentTrigger3 = 0.6;
    % Fy Left  -  -  -  -  -  -
    LeftIndex1(c2) = relDataSTART2(c2);         %initiate position of left boundary of plateau
    while yPercent(LeftIndex1(c2)) < percentTrigger3
        LeftIndex1(c2) = LeftIndex1(c2) + 1;    %at completion of loop, this will be the index of
the plateau's trigger on
    end
    % Fy Right
    RightIndex1(c2) = relDataSTOP2(c2);         %initiate position of left boundary of plateau
    while yPercent(RightIndex1(c2)) < percentTrigger3
        RightIndex1(c2) = RightIndex1(c2) - 1;  %at completion of loop, this will be the index of
the plateau's trigger off
    end
    %  -  -  -  -  -  -  -  -  -
    % * * * * * * * * * * * * * * * * * * * * * * * * *

end


    % Plateau Processing / Short Plateau Correction = = = = = = = = = = = =

    % Correction of Forces o o o o o o o o o o o o o
    for c3 = 1:Nrel2      %counter c3 is essentially identical to c2

       % Fy Plateau
       FyPlateauDomain = Fy(LeftIndex1(c3):RightIndex1(c3));               %vector of forces on
current plateau [N]
       lengthPlatFy(c3) = length(FyPlateauDomain);                         %length of current
plateau
       if lengthPlatFy(c3)/(relDataSTOP2(c3)-relDataSTART2(c3)+1) < .01     %if current plateau
is short...
          lengthPlatFy(c3) = relDataSTOP2(c3)-relDataSTART2(c3)+1;          %...this accounts for
incorrect short plateau (short means <1% of rel2 range)...
          FyPlateauDomain = Fy(relDataSTART2(c3):relDataSTOP2(c3));         %...by making plateau
the entire rel2 range
       end
       FyPlateauAvgs(c3) = mean(FyPlateauDomain);                           %current plateau's
average force [N]
       FyPlateauStds(c3) = std(FyPlateauDomain);                            %current plateau's
standard deviation [N]

       % Fz Plateau
       FzPlateauDomain = Fz(LeftIndex1(c3):RightIndex1(c3));               %vector of forces on
current plateau [N]
```

```
       lengthPlatFz(c3) = length(FzPlateauDomain);                          %length of current
plateau
       if lengthPlatFz(c3)/(relDataSTOP2(c3)-relDataSTART2(c3)+1) < .01     %if current plateau
is short...
          lengthPlatFz(c3) = relDataSTOP2(c3)-relDataSTART2(c3)+1;          %...this accounts for
incorrect short plateau (short means <1% of rel2 range)...
          FzPlateauDomain = Fz(relDataSTART2(c3):relDataSTOP2(c3));         %...by making plateau
the entire rel2 range
       end
       FzPlateauAvgs(c3) = mean(FzPlateauDomain);                           %current plateau's
average force [N]
       FzPlateauStds(c3) = std(FzPlateauDomain);                            %current plateau's
standard deviation [N]

    end
    % o o o o o o o o o o o o o o o o o o o o o o o o


    % Correction of Start/Stop Indices / \ / \ / \ /
    %First Instance _ _ _ _ _ _ _ _
    FyPlateaus = Fy(LeftIndex1(1):RightIndex1(1));                          %initiate vector of
Plateau Forces [N]
    lengthToAddFy(1) = length(FyPlateaus);                                  %initiate vector
containing lengths of individual plateaus
    if lengthToAddFy(1)/(relDataSTOP2(1)-relDataSTART2(1)+1) < .01          %if current plateau
is short (short means <1% of rel2 range)...
       lengthToAddFy(1) = relDataSTOP2(1)-relDataSTART2(1)+1;               %...this accounts for
incorrect short plateau...
       FyLeftIndex2(1) = relDataSTART2(1);                                  %...by making left
index of plateau same as rel2 START index...
       FyRightIndex2(1) = relDataSTOP2(1);                                  %...and by making
right index of plateau same as rel2 STOP index
    else                                                                    %the left&right
indices stay the same, since it's not a short plateau
       FyLeftIndex2(1) = LeftIndex1(1);
       FyRightIndex2(1) = RightIndex1(1);
    end
    FyPlateaus = Fy(FyLeftIndex2(1):FyRightIndex2(1));                      %redefine plateaus
vector after short plateaus have been accounted for

    FzPlateaus = Fz(LeftIndex1(1):RightIndex1(1));                          %initiate vector of
Plateau Forces [N]
    lengthToAddFz(1) = length(FzPlateaus);                                  %initiate vector
containing lengths of individual plateaus
    if lengthToAddFz(1)/(relDataSTOP2(1)-relDataSTART2(1)+1) < .01          %if current plateau
is short (short means <1% of rel2 range)...
       lengthToAddFz(1) = relDataSTOP2(1)-relDataSTART2(1)+1;               %...this accounts for
incorrect short plateau...
       FzLeftIndex2(1) = relDataSTART2(1);                                  %...by making left
index of plateau same as rel2 START index...
       FzRightIndex2(1) = relDataSTOP2(1);                                  %...and by making
right index of plateau same as rel2 STOP index
    else                                                                    %the left&right
indices stay the same, since it's not a short plateau
       FzLeftIndex2(1) = LeftIndex1(1);
       FzRightIndex2(1) = RightIndex1(1);
    end
    FzPlateaus = Fz(FzLeftIndex2(1):FzRightIndex2(1));                      %redefine plateaus
vector after short plateaus have been accounted for

    %_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

    %Other Instances * * * * * * * * *
    for c4 = 2:Nrel2
       lengthFyPlat = length(FyPlateaus);                                   %length of force
vector for all plateaus up to present c4
       lengthToAddFy(c4) = RightIndex1(c4) - LeftIndex1(c4) + 1;            %length of current
force vector to be appended
       if lengthToAddFy(c4)/(relDataSTOP2(c4)-relDataSTART2(c4)+1) < .01    %if current plateau
is short (short means <1% of rel2 range)...
```

```
           lengthToAddFy(c4) = relDataSTOP2(c4)-relDataSTART2(c4)+1;       %...rewrite
lengthToAddFy to be the length of current rel2 range
           FyPlateaus(lengthFyPlat+1:lengthFyPlat+lengthToAddFy(c4),1) =
Fy(relDataSTART2(c4):relDataSTOP2(c4)); %make plateau the entire rel2 range
           FyLeftIndex2(c4) = relDataSTART2(c4);                            %left  index of
plateau AFTER short plateau correction
           FyRightIndex2(c4) = relDataSTOP2(c4);                            %right index of
plateau AFTER short plateau correction
       else                                                                 %plateau is ok
length, proceed with standard plateau finder
           FyLeftIndex2(c4) = LeftIndex1(c4);                               %the left index stays
the same, since it's not a short plateau
           FyRightIndex2(c4) = RightIndex1(c4);                             %the right index
stays the same, since it's not a short plateau
           FyPlateaus(lengthFyPlat+1:lengthFyPlat+lengthToAddFy(c4),1) =
Fy(FyLeftIndex2(c4):FyRightIndex2(c4));    %cumulative force vector [N]
       end

       lengthFzPlat = length(FzPlateaus);                                   %length of force
vector for all plateaus up to present c4
       lengthToAddFz(c4) = RightIndex1(c4) - LeftIndex1(c4) + 1;            %length of current
force vector to be appended
       if lengthToAddFz(c4)/(relDataSTOP2(c4)-relDataSTART2(c4)+1) < .01
           lengthToAddFz(c4) = relDataSTOP2(c4)-relDataSTART2(c4)+1;       %rewrite
lengthToAddFz if instance c4 is a "short" plateau (short means <1% of rel2 range)
           FzPlateaus(lengthFzPlat+1:lengthFzPlat+lengthToAddFz(c4),1) =
Fz(relDataSTART2(c4):relDataSTOP2(c4)); %make plateau the entire rel2 range
           FzLeftIndex2(c4) = relDataSTART2(c4);                            %left  index of
plateau AFTER short plateau correction
           FzRightIndex2(c4) = relDataSTOP2(c4);                            %right index of
plateau AFTER short plateau correction
       else                                                                 %plateau is ok
length, proceed with standard plateau finder
           FzLeftIndex2(c4) = LeftIndex1(c4);                               %the left index stays
the same, since it's not a short plateau
           FzRightIndex2(c4) = RightIndex1(c4);                             %the right index
stays the same, since it's not a short plateau
           FzPlateaus(lengthFzPlat+1:lengthFzPlat+lengthToAddFz(c4),1) =
Fz(FzLeftIndex2(c4):FzRightIndex2(c4));    %cumulative force vector [N]
       end
    end
    % * * * * * * * * * * * * * * * * *
    % / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ /


    FyPlatAvg = mean(FyPlateaus);                                           %average across all
data points included in plateau
    FzPlatAvg = mean(FzPlateaus);                                           %average across all
data points included in plateau

    tPlateausY = 0:dT:(length(FyPlateaus)-1)*dT;                            %create vector of
pseudotimes for Work integration
    tPlateausZ = 0:dT:(length(FzPlateaus)-1)*dT;                            %create vector of
pseudotimes for Work integration
    % = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

%% PROCESS RELEVANT DATA

%                           F   O   R   C   E   S

%Algorithm 1 - - - - - - - - -
FxRelev1 = Fx([relDataSTART1:relDataSTOP1],1);          %[N]
FyRelev1 = Fy([relDataSTART1:relDataSTOP1],1);          %aka Fp [N]
FzRelev1 = Fz([relDataSTART1:relDataSTOP1],1);          %aka Fq [N]
distCutRelev1 = distCut([relDataSTART1:relDataSTOP1],1);%to be used in least squares fitting [m]

tRelev1 = t([relDataSTART1:relDataSTOP1]);              %time vector of relevant data [s]
FRelev1Sq = [FxRelev1.^2, FyRelev1.^2, FzRelev1.^2];    %matrix with Fx^2, Fy^2, Fz^2 in columns
[N]
FRelev1SqSum = sum(FRelev1Sq,2);                        %sum across rows
```

```
FRelev1Res = sqrt(FRelev1SqSum);                        %resultant force considering Fx, Fy, Fz
[N]
% - - - - - - - - - - - - - - -


%Algorithm 2 < > < > < > < > < >
FxRelev2(:,1) = Fx(relDataSTART2(1):relDataSTOP2(1));   %initiate vector of relevant forces [N]
FyRelev2(:,1) = Fy(relDataSTART2(1):relDataSTOP2(1));   %initiate vector of relevant forces [N]
FzRelev2(:,1) = Fz(relDataSTART2(1):relDataSTOP2(1));   %initiate vector of relevant forces [N]
displRelev2 = D_AC(relDataSTART2(1):relDataSTOP2(1));   %initiate vector of relevant
displacements [um]

for k = 2:Nrel2
    lengthToAdd = relDataSTOP2(k) - relDataSTART2(k) + 1;    %length of current vector to be
appended (add 1 so it includes both start and end positions' data)

    lengthFx = length(FxRelev2);
    FxRelev2(lengthFx+1:lengthFx+lengthToAdd,1) = Fx(relDataSTART2(k):relDataSTOP2(k)); %append
force data [N]

    lengthFy = length(FyRelev2);
    FyRelev2(lengthFy+1:lengthFy+lengthToAdd,1) = Fy(relDataSTART2(k):relDataSTOP2(k)); %append
force data [N]

    lengthFz = length(FzRelev2);
    FzRelev2(lengthFz+1:lengthFz+lengthToAdd,1) = Fz(relDataSTART2(k):relDataSTOP2(k)); %append
force data [N]

    lengthdispl = length(displRelev2);
    displRelev2(lengthdispl+1:lengthdispl+lengthToAdd,1) =
D_AC(relDataSTART2(k):relDataSTOP2(k)); %append displacement data [um]
end

tRelev2 = 0:dT:(length(FxRelev2)-1)*dT;                 %does NOT correspond to real time, but
can be used for time integration [s]
FRelev2Sq = [FxRelev2.^2, FyRelev2.^2, FzRelev2.^2];    %matrix with Fx^2, Fy^2, Fz^2 in columns
[N]
FRelev2SqSum = sum(FRelev2Sq,2);                        %sum across rows
FRelev2Res = sqrt(FRelev2SqSum);                        %resultant force considering Fx, Fy, Fz
[N]
% < > < > < > < > < > < > < > < >

%Peak/Mean Forces = = = = = = =
%   Peak: Use plateau data (mean +/- 1std)
%   Mean: Use parent (ie relev1) data (mean +/- 1std)

switch f   %f is input freq to piezo
case 0  % : : : : : : :
%Fy PEAK
FyMaxPlusMinus = std(FyPlateaus);                       %std of entire plateaus data set (only 1
plateau b/c f=0) [N]
FyMaxDC = mean(FyPlateaus);                             %mean of entire plateaus data set (only 1
plateau b/c f=0) [N]

%Fy MEAN
FyAvgPlusMinus = std(FyRelev1);                         %std of entire rel1 data set (only 1
plateau b/c f=0) [N]
FyAvgDC = mean(Fy(relDataSTART1:relDataSTOP1));         %mean of entire rel1 data set (only 1
plateau b/c f=0) [N]

%Fz PEAK
FzMaxPlusMinus = std(FzPlateaus);                       %std of entire plateaus data set (only 1
plateau b/c f=0) [N]
FzMaxDC = mean(FzPlateaus);                             %mean of entire plateaus data set (only 1
plateau b/c f=0) [N]

%Fz MEAN
FzAvgPlusMinus = std(Fz(relDataSTART1:relDataSTOP1));   %std of entire rel1 data set (only 1
plateau b/c f=0) [N]
FzAvgDC = mean(Fz(relDataSTART1:relDataSTOP1));         %mean of entire rel1 data set (only 1
plateau b/c f=0) [N]
```

```
% : : : : : : : : : : : :

otherwise  % + + + + + +
for ck = 1:Nrel2
   FyMax(ck) = max(Fy(FyLeftIndex2(ck):FyRightIndex2(ck))); %max force of current Alg2 chunk
   FzMax(ck) = max(Fz(FzLeftIndex2(ck):FzRightIndex2(ck))); %max force of current Alg2 chunk

   FyAvg(ck) = mean(Fy(FyLeftIndex2(ck):FyRightIndex2(ck)));%average force of current Alg2 chunk
   FzAvg(ck) = mean(Fz(FzLeftIndex2(ck):FzRightIndex2(ck)));%average force of current Alg2 chunk
end

%Fy PEAK
FyMaxPlusMinus = sqrt(sum((lengthPlatFy-1).*(FyPlateauStds.^2)) ./ sum(lengthPlatFy-1));
%pooled standard deviation of plateau standard deviations
FyMaxDC = mean(FyPlateauAvgs);
%pooled means of average plateau values [N]

%Fy MEAN
FyAvgPlusMinus = std(FyRelev1);                                                     %std
of entire rel1 data set (not pooled b/c only 1 range)
FyAvgDC = mean(FyRelev1);                                                           %mean
of parent forces [N]

%Fz PEAK
FzMaxPlusMinus = sqrt(sum((lengthPlatFz-1).*(FzPlateauStds.^2)) ./ sum(lengthPlatFz-1));
%pooled standard deviation of plateau standard deviations
FzMaxDC = mean(FzPlateauAvgs);
%pooled means of average plateau values [N]

%Fz MEAN
FzAvgPlusMinus = std(FzRelev1);
%standard deviation of parent forces [N]
FzAvgDC = mean(FzRelev1);                                                           %mean
of parent forces [N]
% + + + + + + + + + + + +
end
% = = = = = = = = = = = = = = = = = =


%                         F R I C T I O N   C O E F F

%Use plateau values (xxPlateauAvg1) to prevent zero/inf values.  The
%friction coefficient is equal to Fz/Fy (ie Fq/Fp)

u = FzPlateauAvgs./FyPlateauAvgs;   %vector of friction coefficients
uDC = mean(u);                      %DC component of friction coefficient
uPlusMinus = std(u);                %plus minus component of friction coefficient

%% LINEAR LEAST SQUARES W/ INPUT FREQ

freqINdist = 1 / ((1/(2*pi*f))*wSI*(diaCut/1000/2));%primary frequency of oscillation in distance
cut metric [rad/m]

d = displRelev1;                                    %set vector d to Alg1's relevant displacement
[um]
C = [cos(freqINdist*distCutRelev1), sin(freqINdist*distCutRelev1),
ones(length(distCutRelev1),1)]; %set matrix C
xIn = inv(C'*C)*C'*d;                               %get coefficients by solving linear equation
resnormIn = norm(d-C*xIn)^2;                        %squared 2-norm of residual
Ainput = xIn(1); Binput = xIn(2); Cinput = xIn(3);  %COEFFICIENTS TO GIVEN EQUATION
sineFitIn = Ainput*cos(freqINdist*distCutRelev1) + Binput*sin(freqINdist*distCutRelev1) + Cinput;
    %Determine R^2 value - Note: R^2 = 1 - RSS/TSS where resnorm=RSS
    RSSIn = resnormIn;                              %residual sum of squares
    TSS = sum((d - mean(d)).^2);                    %total sum of squares
    RsquaredIn = 1 - (RSSIn/TSS);                   %R^2 value for least squares linear fit on
input frequency

%% LINEAR LEAST SQUARES W/ FFT FREQ
C = [cos(freqFFTdist*distCutRelev1), sin(freqFFTdist*distCutRelev1),
ones(length(distCutRelev1),1)]; %set matrix C
xFFT = inv(C'*C)*C'*d;                              %get coefficients by solving linear equation
```

```
resnormFFT = norm(d-C*xFFT)^2;                      %squared 2-norm of residual
Afft = xFFT(1); Bfft = xFFT(2); Cfft = xFFT(3);     %"BEST" COEFFICIENTS TO GIVEN EQUATION by LS
on fft primary frequency
sineFitFFT = Afft*cos(freqFFTdist*distCutRelev1) + Bfft*sin(freqFFTdist*distCutRelev1) + Cfft;
RSSFFT = resnormFFT;
RsquaredFFT = 1 - (RSSFFT/TSS);                     %R^2 value for least squares linear fit on
fft primary frequency

%% LINEAR LEAST SQUARES W/ MODIFIED FFT FREQ

freqMagnify = 50;                               %increase frequency resolution by freqMagnify
times, compared to fft
fft_freqResol = (fs/2)/(length(fft_freq)-1);    %resolution of fft frequency [Hz]
freqResolMagnify = fft_freqResol/freqMagnify;   %new resolution [Hz]
freqFFTModlb = freqFFT-fft_freqResol;           %lower bound on primary frequency spectrum [Hz]
freqFFTModub = freqFFT+fft_freqResol;           %upper bound on primary frequency spectrum [Hz]
%Initialize Rsquared value for lower bound of frequency - to be used in loop
freqFFTModdist = 1 / ((1/(2*pi*freqFFTModlb))*wSI*(diaCut/1000/2)); %primary frequency of
oscillation in distance cut metric [rad/m]
C = [cos(freqFFTModdist*distCutRelev1), sin(freqFFTModdist*distCutRelev1),
ones(length(distCutRelev1),1)]; %set matrix C
xMod = inv(C'*C)*C'*d;                          %get coefficients by solving linear equation
resnormMod = norm(d-C*xMod)^2;                  %squared 2-norm of residual

RSSMod = resnormMod;
RsquaredMod = 1 - (RSSMod/TSS);                 %R^2 value for least squares linear fit
RsquaredModMax = RsquaredMod;                   %initializes max Rsquared: this could change
during convergence loop below
freqFFTModBest = freqFFTModdist;                %initializes best frequency: this could change
during convergence loop below


if f==0  %in this case the following terms are irrelevant. Thus set == 0
freqFFTModBest = 0;
AAmod = 0; Bmod = 0; Cmod = 0;
RSSModBest = 0;
RsquaredModMax = 0;

else
for i = 1:2*freqMagnify-1                                               %allows us to test
frequencies between but not including upper and lower bounds
    freqFFTMod(i) = freqFFTModlb + i*freqResolMagnify;                  %adds number of magnified
delta frequencies to the lower bound frequency [Hz]
    freqFFTModdist = 1 / ((1/(2*pi*freqFFTMod(i)))*wSI*(diaCut/1000/2));%primary frequency of
oscillation in distance cut metric [rad/m]
    C = [cos(freqFFTModdist*distCutRelev1), sin(freqFFTModdist*distCutRelev1),
ones(length(distCutRelev1),1)]; %set matrix C
    xMod = inv(C'*C)*C'*d;                                              %get coefficients by
solving linear equation
    resnormMod = norm(d-C*xMod)^2;                                      %squared 2-norm of
residual

    RSSMod = resnormMod;
    RsquaredMod(i)= 1 - (RSSMod/TSS);                                   %R^2 value for least
squares linear non negative fit

    if RsquaredMod(i) > RsquaredModMax
        freqFFTModBest = freqFFTMod(i);                                 %if this frequency yields
higher Rsquared, save this as best frequency
        RSSModBest = RSSMod;
        RsquaredModMax = 1 - (RSSModBest/TSS);                          %if new test frequency
yields higher Rsquared, save this freqs parameters as the max
        AAmod = xMod(1); Bmod = xMod(2); Cmod = xMod(3);                %"BEST" COEFFICIENTS TO
GIVEN EQUATION by LS on "best" modied frequency
    end
end
end

freqFFTModBestdist = 1 / ((1/(2*pi*freqFFTModBest))*wSI*(diaCut/1000/2)); %primary frequency of
oscillation in distance cut metric [rad/m]
```

```
sineFitMod = AAmod*cos(freqFFTModBestdist*distCutRelev1) +
Bmod*sin(freqFFTModBestdist*distCutRelev1) + Cmod;

%% EFFECTIVE DISPLACEMENT

D_ACfit = sineFitMod;       %use displacement from best fit on modified fft
D_DC = V_DC*tCut;           %turret displacement
D_ACDC = D_DC+D_ACfit;      %cumulative (ie effective) displacement [um]
D_ACDC = D_ACDC-D_ACDC(1);  %offset all values such that ramp starts at D_ACDC = 0

%% DISPLACEMENT OF PREVIOUS PASS

D_ACDCprev(stepsPerRot+1:length(D_ACDC)+stepsPerRot,1) = D_ACDC;    %same as D_ACDC but shifted
the equivalent of one spindle rotation
D_ACDCprev(length(D_ACDC)+1:end)='';                                %eliminate remaining entries
on vector

%% INSTANTANEOUS UNDEFORMED CHIP THICKNESS
% aka "depth of cut"

h = zeros(length(Fy),1);                            %initialize variable
h(relDataSTART1:relDataSTOP1) = D_ACDC-D_ACDCprev;  %undeformed chip thickness [um]
h(h<0)=0;                                           %set all negative values to zero

%% ACTUAL POWER

P = FyRelev1*V;         %Power vector [W]
PAvgRep = FpAvgRep*V;   %Average power [W]

%% PREDICTED POWER
% Use forceFitLathe.m to determine function and its best fit
% coefficients. Force is fit according to F=m*x+b-(b/(A*x+1)), thus
% P=V*(m*x+b-(b/(A*x+1)))

coeff = [8.6860e-01 3.2289e+01 2.5792e-01];         %from forceFitLathe.m
Pest = zeros(length(Fy),1);                         %initialize
Pest = V*(coeff(1)*h + coeff(2) - (coeff(2)./(coeff(3)*h + 1)));    %predict power [W]
Pest(h==0)=0;                                       %set power to zero when tool is separated
from workpiece

%% ENERGY

%                              W   O   R   K   (VOL SPEC.)

l = ((repStop-repStart)*dT*V_DC)/1000;              %length of workpiece cut based on V_DC [mm]
vol = (pi*(do^2)/4 - pi*(di^2)/4)*(l);              %total volume machined [mm^3]
WtotRep = sum(FpRep)*dT*V/vol;                      %total specific work required [J/mm^3]
WtotRepPred = sum(Pest(repStart:repStop))*dT/vol;   %total predicted specific work required
[J/mm^3]
WtotRepTrap = trapz(FpRep)*dT*V/vol;                %check value if trapezoidal integration were
used instead

%% SINGLE SINE FUNCTION
% A*sin(x) + B*cos(x) + C = D*sin(x+p) + C
%   where:
%   D = sqrt(A^2+B^2)
%   p = atan(B/A)

switch SScase
    case 1
    xSS = [distCut(relDataSTART1:relDataSTOP1)]';   %relevant x data
    fCase = f;      %defines which frequency this case uses [Hz]
    freqSS = fCase*360; %piezo frequency [degrees/s]
    y1SS = Ainput*sind(xSS*freqSS) + Binput*cosd(xSS*freqSS) + Cinput; %two sinusoidal terms
    DSS = norm([Ainput,Binput],2);       %magnitude of A & B = D
    pSS = atan2(Binput,Ainput)*180/pi;   %phase shift
    y2SS = DSS*sind(xSS*freqSS+pSS)+Cinput;           %single sinusoidal term
    strSS = 'Input Frequency';  %to be used to ID which case was used in a figure
    case 2
    xSS = [distCut(relDataSTART1:relDataSTOP1)]';   %relevant x data
    fCase = freqFFT;    %defines which frequency this case uses [Hz]
    freqSS = fCase*360;  %fft frequency [degrees/s]
    y1SS = Afft*sind(xSS*freqSS) + Bfft*cosd(xSS*freqSS) + Cfft; %two sinusoidal terms
    DSS = norm([Afft,Bfft],2);       %magnitude of A & B = D
    pSS = atan2(Bfft,Afft)*180/pi;   %phase shift
    y2SS = DSS*sind(xSS*freqSS+pSS)+Cfft;            %single sinusoidal term
    strSS = 'FFT Frequency';    %to be used to ID which case was used in a figure
    case 3
    xSS = [distCut(relDataSTART1:relDataSTOP1)]';   %relevant x data
    fCase = freqFFTModBest;    %defines which frequency this case uses [Hz]
    freqSS = fCase*360;  %modified fft frequency [degrees/s]
    y1SS = AAmod*sind(xSS*freqSS) + Bmod*cosd(xSS*freqSS) + Cmod; %two sinusoidal terms
    DSS = norm([AAmod,Bmod],2);       %magnitude of A & B = D
    pSS = atan2(Bmod,AAmod)*180/pi;   %phase shift
    y2SS = DSS*sind(xSS*freqSS+pSS)+Cmod;           %single sinusoidal term
    strSS = 'Modified FFT Frequency';    %to be used to ID which case was used in a figure
end

%% FIGURE: DISPLACEMENT

hfig1 = figure(1);  %initialize figure window
fullscreen = get(0,'ScreenSize');
set(hfig1,'position',[0,-50 fullscreen(3) fullscreen(4)])
set(hfig1,'color','w');
set(hfig1,'Name','Data Analysis');
ax(2) = subplot(4,1,1);

hold on
plot(tCut,D_ACDC,'b','linewidth',2)
plot(tCut,D_ACDCprev,'--b','linewidth',2)
hold off

set(gca,'fontsize',18);
ylabel('Displacement (um)');
legend('Current Pass','Previous Pass')

if testCase==1                          %if using artificial data...
ylimits = ylim;                         %get current y limits
ylim([ylimits(1) ylimits(2)+50])        %set maximum y limit to .5 + old y limit for plotting
end

%% FIGURE: FORCES

ax(1) = subplot(4,1,3);
hold on

% X-AXIS: DISTANCE CUT [m]
% hFx = plot(t,Fx,'k','parent',ax(1));    %plot ch0
hFy = plot(t,Fy,'m','parent',ax(1));    %plot ch1
hFz = plot(t,Fz,'b','parent',ax(1));    %plot ch2
hStart1 = plot([t(relDataSTART1);t(relDataSTART1)],[min(min([Fx,Fy,Fz])),max(max([Fx,Fy,Fz]))],'-
-g','linewidth',3);   %plot vertical line at relDataSTART
hStop1 = plot([t(relDataSTOP1);t(relDataSTOP1)],[min(min([Fx,Fy,Fz])),max(max([Fx,Fy,Fz]))],'--
r','linewidth',3);   %plot vertical line at relDataSTART

hStart2 =
plot([t(repStart);t(repStart)],[min(min([Fx,Fy,Fz])),max(max([Fx,Fy,Fz]))],':g','linewidth',3);
%plot vertical line at relDataSTART
hStop2 =
plot([t(repStop);t(repStop)],[min(min([Fx,Fy,Fz])),max(max([Fx,Fy,Fz]))],':r','linewidth',3);
%plot vertical line at relDataSTART

htext = gca; set(htext,'fontsize',18);  %set text fontsize
% xlabel('Time(s)');

for k = 1:length(relDataSTART2)
    %Identify relevant data (Algorithm 2)
    hStart2 = plot(t(relDataSTART2(k)),0,'g.','markersize',20,'linewidth',2.5,'parent',ax(1));
%plot relevant data start markers according to algorithm 2
    hStop2 = plot(t(relDataSTOP2(k)),0,'rx','markersize',8,'linewidth',2.5,'parent',ax(1));
%plot relevant data stop markers according to algorithm 2
```

```
    %Identify plateau in Fy Data (Algorithm 1)
    hPlatFy1 =
plot([t(FyLeftIndex2(k)),t(FyRightIndex2(k))],[FyPlateauAvgs(k),FyPlateauAvgs(k)],'m--
','linewidth',2,'parent',ax(1));   %plot horizontal line in between markers

    %Identify plateau in Fz Data (Algorithm 1)
    hPlatFz1 =
plot([t(FzLeftIndex2(k)),t(FzRightIndex2(k))],[FzPlateauAvgs(k),FzPlateauAvgs(k)],'b--
','linewidth',2,'parent',ax(1));   %plot horizontal line in between markers
end

htext = gca; set(htext,'fontsize',18);  %set text fontsize
ylabel('Force (N)');

%Switch/Case for z Drift Compensation - - - - - - - -
switch showdriftZComp
    case 0
        %LEGH =
legend([hFx,hFy,hFz,hPlatFy1,hPlatFz1,hStart1,hStop1,hStart2,hStop2],'Fx','Fy','Fz','Fy
Plateau','Fz Plateau','Continuous','Continuous','Discrete','Discrete');
        LEGH = legend([hFy,hFz,hPlatFy1,hPlatFz1],'Fp','Fq','Fp Plat','Fq Plat');
    case 1
        hFzIni = plot(distCutNonRel,FzNonRel,'.c','markersize',0.1);
        hFzDriftLine = plot(distCut,FzDriftLine,'r','linewidth',0.1);
        [LEGH,OBJH,OUTH,OUTM] = legend([hFy,hFz,hFzDriftLine],'Fp','Fq','Drift Comp');
end
%- - - - - - - - - - - - - - - - - - - - - - - - - -

if testCase==1
ylimits = ylim;                     %get current y limits
ylim([ylimits(1) ylimits(2)+.5])    %set maximum y limit to .5 + old y limit for plotting
end
hold off

%% FIGURE: DEPTH OF CUT

ax(3) = subplot(4,1,2);

hold on
plot(tCut,h(relDataSTART1:relDataSTOP1),'b','linewidth',2)
hold off

set(gca,'fontsize',18)
strY = sprintf('Undeformed Chip\nThickness (um)');
ylabel(strY)
ylim([-1,max(h(relDataSTART1:relDataSTOP1))*1.1])

%% FIGURE: POWER

ax(4) = subplot(4,1,4);

hold on
plot(tCut,P,'k','linewidth',2)
plot(tCut,Pest(relDataSTART1:relDataSTOP1),'r','linewidth',2)
hold off

set(gca,'fontsize',18)
xlabel('Time (s)')
ylabel('Power (W)')
legend('Actual','Predicted')

linkaxes(ax,'x');   %synchronize limits of x-axes on 1st and 2nd subplots

%% REPORT DATA

disp(sprintf('\n'))  %2 blank lines
disp('___________________________________________________________________')
disp(sprintf('\n'))  %2 blank lines
disp('      O   U   T   P   U   T   S   ')
disp(sprintf('\n'))  %2 blank lines
disp('MODULATION PARAMETERS - - - - - - - - - - -')
str1 = sprintf('fm                       = %.4f Hz',freqFFTModBest);
str2 = sprintf('Modulation Amplitude K   = %.3f um',norm([AAmod,Bmod],2));
disp(str1)
disp(str2)
disp('- - - - - - - - - - - - - - - - - - - - - -')
disp(sprintf('\n'))  %2 blank lines

disp('FORCE   - - - - - - - - - - - - - - - - - -')
str5 = sprintf('Mean Representative Fp = %.3f N', FpAvgRep);
switch f
    case 0
        str6 = sprintf('Mean Plateau Fps       = %.3f +/- %.3f
N',mean(FyPlateauAvgs(1:end)),std(FyPlateauAvgs(1:end)));
        str7 = sprintf('Mean Plateau Fqs       = %.3f +/- %.3f
N',mean(FzPlateauAvgs(1:end)),std(FzPlateauAvgs(1:end)));
    otherwise
        str6 = sprintf('Mean Plateau Fps       = %.3f +/- %.3f
N',mean(FyPlateauAvgs(2:end)),std(FyPlateauAvgs(2:end)));
        str7 = sprintf('Mean Plateau Fqs       = %.3f +/- %.3f
N',mean(FzPlateauAvgs(2:end)),std(FzPlateauAvgs(2:end)));
end
disp(str5)
disp(str6)
disp(str7)
disp('- - - - - - - - - - - - - - - - - - - - - -')
disp(sprintf('\n'))  %2 blank lines

% disp('FRICTION COEFF  - - - - - - - - - - - - - -')
% str1 = sprintf('Friction coefficient = %.3f +/- %.3f',uDC,uPlusMinus);
% disp(str1)
% disp('- - - - - - - - - - - - - - - - - - - - - -')
% disp(sprintf('\n'))  %2 blank lines

disp('POWER - - - - - - - - - - - - - - - - - - -')
str1 = sprintf('Average  = %.3f J/s',PAvgRep);
disp(str1)
disp('- - - - - - - - - - - - - - - - - - - - - -')
disp(sprintf('\n'))  %2 blank lines

disp('WORK REQUIRED   - - - - - - - - - - - - - -')
str1 = sprintf('Representative Actual    = %.4f J/mm^3',WtotRep);
str2 = sprintf('Representative Predicted = %.4f J/mm^3',WtotRepPred);
disp(str1)
disp(str2)
disp('- - - - - - - - - - - - - - - - - - - - - -')
disp(sprintf('\n'))  %2 blank lines

disp('LEAST SQUARES - - - - - - - - - - - - - - -')
% str1 = sprintf('On Input Frequency: f=%.4f [Hz]',f);
% str2 = sprintf('\t LSBestFit = %.4f*Cos + %.4f*Sin + %.4f',Ainput,Binput,Cinput);
% str3 = sprintf('\t RSS = %.2E',RSSIn);
% str4 = sprintf('\t R^2 = %.4f',RsquaredIn);
% disp(str1)
% disp(str2)
% disp(str3)
% disp(str4)
% disp(' ')
%
% str5 = sprintf('On Primary FFT Frequency: f=%.4f [Hz]',freqFFT);
% str6 = sprintf('\t LSBestFit = %.4f*Cos + %.4f*Sin + %.4f',Afft,Bfft,Cfft);
% str7 = sprintf('\t RSS = %.2E',RSSFFT);
% str8 = sprintf('\t R^2 = %.4f',RsquaredFFT);
% disp(str5)
% disp(str6)
% disp(str7)
% disp(str8)
% disp(' ')

str9 = sprintf('On %ix Magnified Resolution FFT Frequency: f=%.4f
[Hz]',freqMagnify,freqFFTModBest);
```

```
str10 = sprintf('\t LSBestFit = %.4f*Cos + %.4f*Sin + %.4f',AAmod,Bmod,Cmod);
str11 = sprintf('\t RSS = %.2E',RSSModBest);
str12 = sprintf('\t R^2 = %.4f',RsquaredModMax);
disp(str9)
disp(str10)
disp(str11)
disp(str12)
disp('- - - - - - - - - - - - - - - - - - - - -')
disp(sprintf('\n'))  %2 blank lines

disp('DISPLACEMENT FIT EQUATION - - - - - - - -')
str1 = ('X-axis: Distance Cut [m]  |  Y-axis: Displacement [um]');
str2 = sprintf('Best Fit Equation Using %s (%.3f Hz)',strSS,fCase);
str3 = sprintf('%.3f*Sin(%.0f*X+%.3f)+%.3f',DSS,freqSS,pSS,Cinput);
disp(str1)
disp(str2)
disp(str3)
disp('- - - - - - - - - - - - - - - - - - - - -')

disp(sprintf('\n'))  %2 blank lines
disp('___________________________________________________________________')
```