ABSTRACT

NOWAK, WILLIAM JOHN. Ultrasonic Nanocoining of Sub-micrometer Surface Features. (Under the direction of Dr. Thomas A. Dow).

The goal of this work is to develop a machining process to rapidly generate large areas of sub-micrometer surface features in industrially-feasible amounts of time. These surface features, inspired by designs found in nature, can be sized to produce desired behavior including anti-reflective (AR), superhydrophobic, or syperhydrophilic properties. Current methods of producing these features produce high-fidelity replications, but often require several steps to complete and are not economically feasible in a mass-production setting.

This process uses a diamond die containing thousands of subwavelength features mounted to a high speed actuator on the axis of a diamond turning machine. The die is pressed into the workpiece with an elliptical path that matches the surface speed of the moving workpiece and minimizes distortion caused by mismatches in velocity. These die indents are tiled together to create large areas of uniform features, which can then be easily transferred to pliable coatings. Because the die is small (20x20 μm), the indentations must occur rapidly (40 kHz) to make nanocoining an industrially feasible process.

To produce the ultrasonic 2-D motion for indenting, an actuator was designed that resonates in two orthogonal directions at the same frequency. Analytical solutions were first used to determine the approximate geometry to achieve the desired resonant behavior. Finite element analysis (FEA) is then employed to fine-tune the geometry of the device to meet amplitude requirements. Prototypes were constructed and measured to demonstrate the utility of the proposed design method.

A controller is designed and implemented to automatically track the resonant frequency and maintain the desired actuator behavior to ensure the indents are formed uniformly over large areas despite piezoelectric self-heating effects and other environmental affects that impact device resonance. Other control strategies that are required for maintaining uniform spacing

of indents over large areas are proposed including a surface-following stage to compensate for runout errors in the workpiece, spindle speed compensations and actuator amplitude feedback.

This ultrasonic nanocoining system was demonstrated to yield well-formed features at 40 kHz on both 6061 aluminum and 360 brass cylindrical workpieces. The die features are examined qualitatively using a scanning electron microscope. This dissertation demonstrates the feasibility of producing sub-micrometer surface features by ultrasonic indentation.

Ultrasonic Nanocoining of Sub-micrometer Surface Features

by
William John Nowak, Jr.

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Mechanical Engineering

Raleigh, North Carolina

2014

APPROVED BY:

_____	_____
Dr. Thomas Dow Committee Chair	Dr. Ronald Scattergood
_____	_____
Dr. Paul Ro	Dr. Jeffrey Eischen

BIOGRAPHY

John Nowak was born in Rochester, NY and is currently a Ph.D. candidate in the Department of Mechanical and Aerospace Engineering at North Carolina State University. He received his Bachelor of Science in Mechanical Engineering from Rochester Institute of Technology in 2009 and continued on to earn his Master of Science in Mechanical Engineering from RIT in 2011. During his Masters, he worked at the Hybrid Sustainable Energy Systems (HySES) lab at RIT where his research involved developing control strategies for hybrid solid oxide fuel cell power systems. John currently works at the Precision Engineering Center at NCSU under the advisement of Dr. Thomas Dow where his research involves developing a novel precision machining process to rapidly create sub-micrometer surface features by indentation.

ACKNOWLEDGMENTS

Special thanks to Dr. Dow. I will forever be thankful for the opportunity to perform research under his guidance. His insight and expertise were, and will continue to be, invaluable to my professional development. Thanks for putting up with me.

Thank you to my committee members Dr. Ronald Scattergood, Dr. Jeffrey Eischen and Dr. Paul Ro for their guidance during my graduate career.

Thanks to Ken Garrard and Alex Sohn for answering all of my questions and helping me with just about everything.

Thanks to Erik Zdanowicz, Guillaume Robichaud, Brandon Lane, Zach Marston, Sean Gunning, Greg Hesler, David Gebb and Stephen Furst. Despite everything I've ever said about you guys behind your backs, I have truly enjoyed working with all of you.

Thanks to Jim Nelson, Viv Jones and Qunyi Chen from 3M for providing professional guidance through the course of my graduate work.

A special thanks to my family and friends who support me, even though they have no idea what I do or why I'm still in school.

This project was funded by the National Science Foundation grant CMMI-100005 monitored by Khershed Cooper.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

- *a* – horizontal amplitude of elliptical tool path
- *AR* – anti-reflective
- *b* – vertical amplitude of elliptical tool path
- *BLT* – bolt-clamped Langevin transducer
- *BVD* – Butterworth Van Dyke (equivalent electrical model for ultrasonic transducers)
- *DDS – direct digital synthesis*
- *die* – diamond indenter containing nanofeatures that is pressed into a workpiece
- *digipot* – digital potentiometer
- *distortion* – dragging of the die along the workpiece creating elongated features due to a velocity mismatch between the die and workpiece
- *DOF* – degree of freedom
- *f* – frequency of indenting (Hz)
- *FIB* – focused ion beam (machining)
- *fr – crossfeed rate*
- L_{die} – length of die in the direction of the workpiece motion
- *Lock-in* – amplifier device that synchronizes an output signal to a reference signal in frequency and phase
- *mold* – final indented workpiece used to form features on a pliable coating
- *nanofeatures* – sub-micrometer features machined into diamond die
- *overlap* – when a die impression re-indents over an area that was previously indented
- *PEC* – Precision Engineering Center
- *PZT* – lead zirconate titanate
- *r* – radius of cylindrical workpiece
- *SEM* – scanning electron microscope
- *SPI* – serial peripheral interface
- W_{die} – width of die orthogonal to workpiece motion

- *workpiece* – substrate the die is pressed into to create the desired features
- $\omega_{spindle}$ – rotational velocity of the spindle holding the cylindrical workpiece

1 INTRODUCTION

1.1 BACKGROUND

Bio-inspired structured surfaces are commonly used in developing coatings that can exhibit a variety of traits, including anti-reflective (AR), superhydrophobic or superhydrophilic. AR surfaces are of particular interest due to their applications in optical lenses, solar cells, and photosensitive detectors [1]. To achieve anti-reflective properties, these surfaces are designed to mimic the nanopillar structures found on the surfaces of moth and butterfly eyes, or cicada and hawkmoth wings. In nature, these AR properties offer survival advantages including camouflage from predators and increased visibility in low light settings [2, 3]. Smaller than the wave length of light, these periodic features introduce a gradual index of refraction profile that reduces the amount of reflection compared to an unstructured surface [4]. Moth eye inspired structures have demonstrated an order of magnitude less reflectance then traditional AR thin films in the visible light and near-infrared spectrum [5].

1.2 ANTIREFLECTIVE STRUCTURED SURFACES

As light passes from a medium with a given index of refraction to another medium with a different index of refraction, reflection and transmission occurs. Considering for simplicity the approaching light to be normal to a given surface (Schnell's law is used for approach angles not equal to 90°), the amount of reflection that occurs is governed by the Fresnel equation:

$$R = \left(\frac{n_1 - n_2}{n_1 + n_2} \right)^2 \tag{1-1}$$

where R is the fraction of light reflected, and n_1 and n_2 are the refractive indices of the two mediums. Likewise, the amount of light transmitted through the surface is defined as $T = 1 - R$. From Equation (1-1) it can be concluded that as the difference between n_1 and n_2

increases, the amount of reflected light increases and the amount transmitted decreases. An illustration of this phenomenon is given below in Figure 1-1:

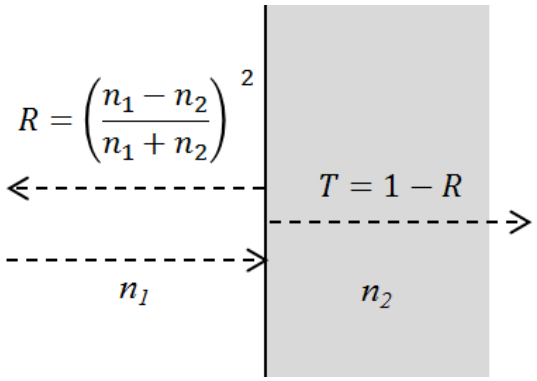


Figure 1-1. Light reflected and transmitted at an interface.

The goal of antireflective surface coatings is to minimize the reflectance of a surface to a specific wavelength, or range of wavelengths of light. This is often done using thin film coatings that can achieve AR properties in different ways. The first method, index-matching, uses multiple thin surface coatings to gradually transition the index of refraction between two mediums, which can reduce the reflectance of a surface. It has been observed, however, that it is difficult to achieve necessarily low indices of refraction using this method [6]. The other thin film method involves using destructive interference to cancel out reflected light. In this technique, a single film layer is sized to be a quarter-wavelength thick, depending on the wavelength of light that is to be canceled out. The light enters the thin film and reflects off of a medium where, since the thin film is a quarter wavelength, its reflected light is 180° out-of-phase with the incoming light resulting in destructive interference [7]. Several layers can be added for different wavelengths of light to result in broadband anti-reflection; however, the added layers increase the complexity of manufacturing process.

Periodic structured surfaces present a solution to creating antireflective surfaces by mixing substrate material with air on a subwavelength scale. If the periodic structures are large compared to the wavelength of light, diffraction occurs. If the periodic structures are small

compared to the wavelength of light, the structures combine with air to act as a homogenous film. Using this property, the structures can be designed to taper from a small cross-section to a large cross-section, resulting in a gradual transition between the two indices that will minimize reflectance. This concept is illustrated in Figure 1-2.

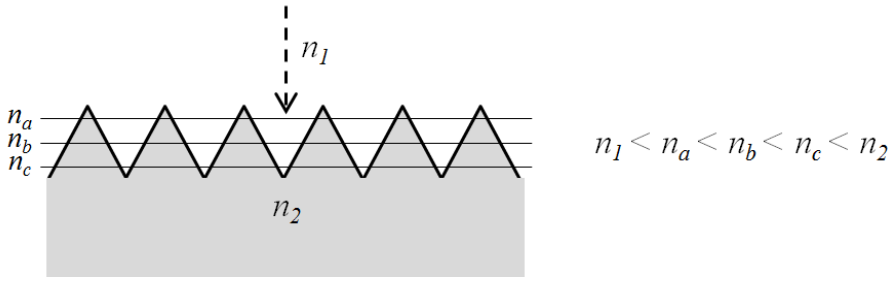


Figure 1-2. Structured subwavelength antireflective surface profile.

The size and spacing of the features are governed by the desired wavelength range of antireflection desired. For the simple case of normal incident light, the period of the structures must be smaller than the vacuum wavelength of the light, while the depth of the features is set by the longest wavelength required to be antireflective. According to Gombert [6], increasing the structure depth decreases the reflectance of the surface assuming that the period is sufficiently small. This tall aspect ratio requirement tends to add difficulty in manufacturing these types of AR surface treatments.

In nature, AR surface structures are found in such places as moth and butterfly eyes, as well as cicada and hawkmoth wings. These surfaces offer survival advantages including camouflage from predators and increased visibility in low light settings [2, 3]. Figure 1-3 includes SEM images of the subwavelength surface structures on the surface of a hawkmoth wing and moth eye.

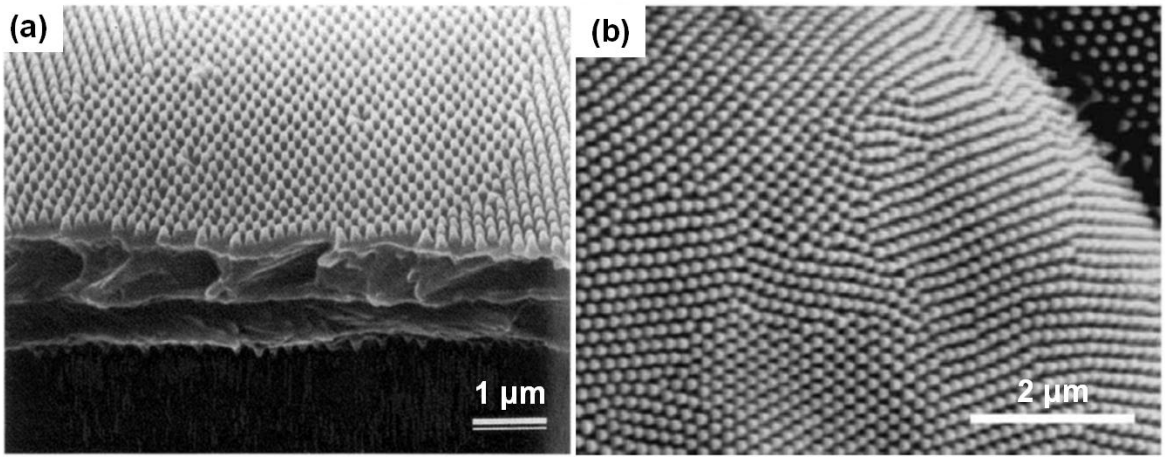


Figure 1-3. SEM images of a hawkmoth wing (a) and a moth eye (b) [2, 8]

Figure 1-4 demonstrates the reduction in reflectance over all wavelengths and varying incident angles with the moth eye structures compared to a 100 nm thick alumina thin film on silicon. The results show an order of magnitude reduction in reflectance over the entire visible light spectrum, as well as less than 1% reflectance in the moth eye structures for 600 nm wavelength light with incident angles up to 75° [5]. These results suggest that the use of moth eye structures could drastically improve the performance of light-sensitive devices such as optical systems or photovoltaics where reducing the reflected light is desired.

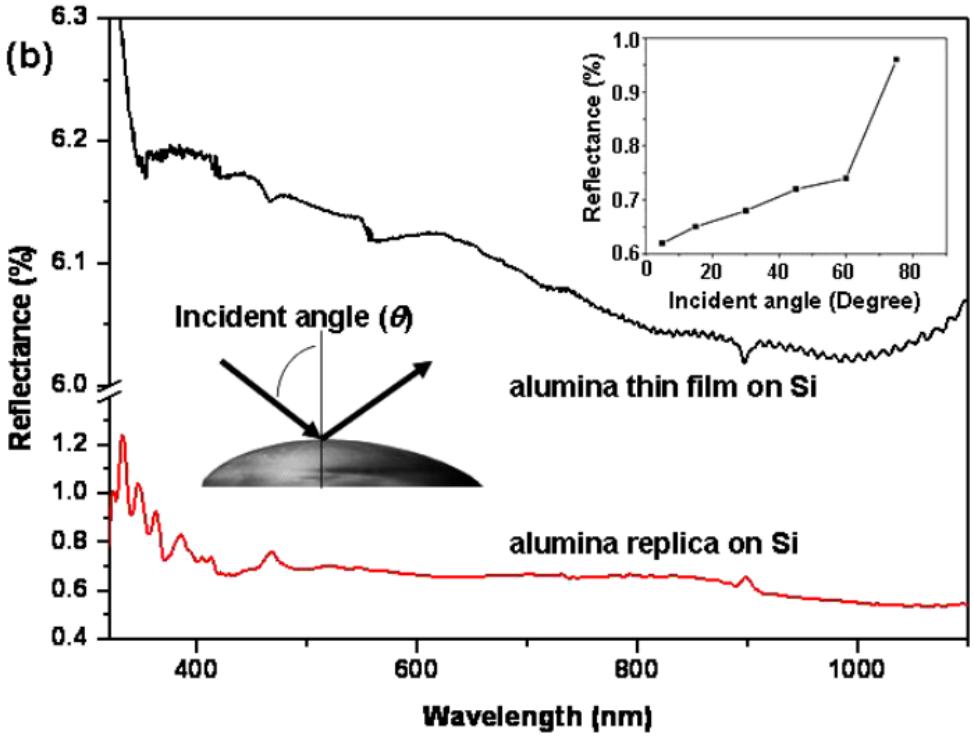


Figure 1-4. Thin film coatings versus moth eye structure replicates on silicon [5].

1.3 SUPERHYDROPHOBIC/SUPERHYDROPHILIC SURFACES

The terms hydrophilic and hydrophobic generally describe the behavior of fluid on a solid surface. Surfaces that exhibit these traits are often categorized by the angle a drop of water makes when in contact with a nominally flat surface. The term hydrophilic is often employed when the contact angle is less than 90° where hydrophobic refers to contact angle greater than 90°. Superhydrophilic surfaces are characterized by a contact angle of less than 5° while an angle larger than around 150° is generally considered superhydrophic [9]. Figure 1-5 illustrates the measurement and characterization of the contact angle between a surface and a drop of water.

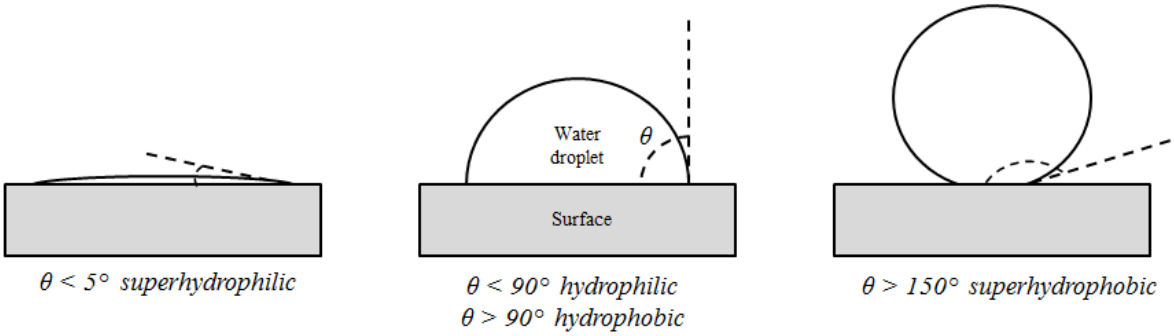


Figure 1-5. Contact angle measurement and classification of a water drop on a smooth surface.

The contact angle of a liquid drop on an ideally smooth surface is often quantified using Young's relation. Young's relation is an energy balance between the surface energies and the liquid surface tension as a result of molecular interactions and can be expressed as $\gamma cos\theta = \gamma_{SA} - \gamma_{SL}$ where $\gamma_{SA}$is the solid/air surface tension, $\gamma_{SL}$is the solid/liquid surface tension, γ is the liquid/air surface tension, and θ is the contact angle (Figure 1-6) [10].

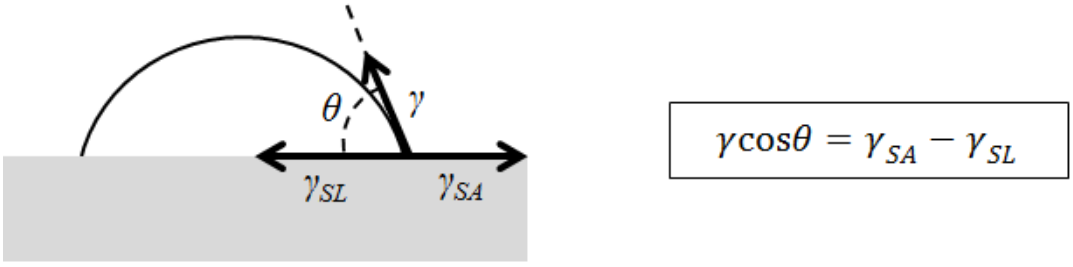


Figure 1-6. Young's relationship for contact angles.

This relationship is important in understanding the influence of the substrate on the behavior of the liquid in contact with it. By inspection of Young's relationship (considering 0° < θ < 180°), it can be shown that when the solid/air surface energy is larger than the solid/liquid surface energy, θ will be less than 90° (hydrophilic). In contrast, for solid/air surface energies less than solid/liquid surface energies, θ will be greater than 90° (hydrophobic).

Most high surface energy natural and man-made materials are hydrophilic. This includes biological membranes, inorganic minerals such as silicates, hyrdoxylated oxides, ionic crystals, metallic surfaces and most polymers. Additionally, anything that adds polarity to a polymer or molecule (adding oxygen to the structure of hydrocarbons, for example) reduces a material's hydrophobicity due to an added wicking effect that attracts the water to the surface. Only materials such as wax, polyethylene, polypropylene, self-assembled monolayers with hydrocarbon functional groups, fluorine-based polymers and hydrocarbons tend to be naturally hydrophobic due to low surface energies [9].

The addition of roughness to a surface influences the behavior of the water drop upon contact as compared to an ideally smooth surface. When a liquid is formed or dropped on the rough surface, it can either wick into the small disparities and spread or sit on top of the rough surface, trapping air in the areas between the higher parts of the surface. It is assumed in both cases that the features are sufficiently small relative to the size of the liquid droplet.

The two models illustrated in Figure 1-7 are commonly used to describe the behavior of a water droplet on a rough surface.

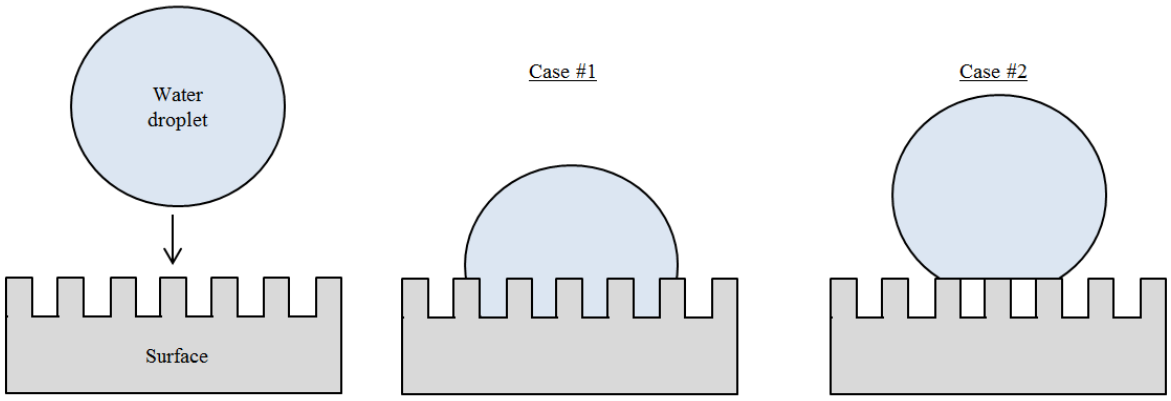


Figure 1-7. Water on a rough surface for Wenzel (Case #1) and Cassie-Baxter (Case #2) models.

Case 1: When the spaces between the features are filled with liquid, the Wenzel model states [9]:

$$\cos\theta_a = r\cos\theta \tag{1-2}$$

where θ is the contact angle for a smooth surface (illustrated in Figure 1-5), θ_a is the apparent contact angle on the rough surface and r is the roughness factor (ratio of actual surface area to projected surface area). Since no surface is perfectly flat, r will always be greater than one. Therefore, this model states that surface roughness always reinforces hydrophobicity or hydrophilicity.

Case 2: If air is trapped in the spaces between the features and forces the drop to sit on top of the surface, the Cassie-Baxter model is used [9]:

$$\cos\theta_a = rf\cos\theta + f - 1 \tag{1-3}$$

where f is the area fraction of wetted area. It can be concluded from the Cassie-Baxter model that as f decreases (more air gaps), apparent contact angle or hydrophobicity increases.

Superhydrophobic surfaces can be found nature, such as on the lotus leaf. The small proturbances on the surface of the plant leaf result in a superhydrophobic property that causes water droplets to bead up and easily roll off of the surface. This behavior acts as a self-cleaning mechanism where the water rolls off the surface, collecting dirt and debris [11]. An SEM image and a water droplet sitting on the hydrophobic lotus leaf surface are shown in Figure 1-8.

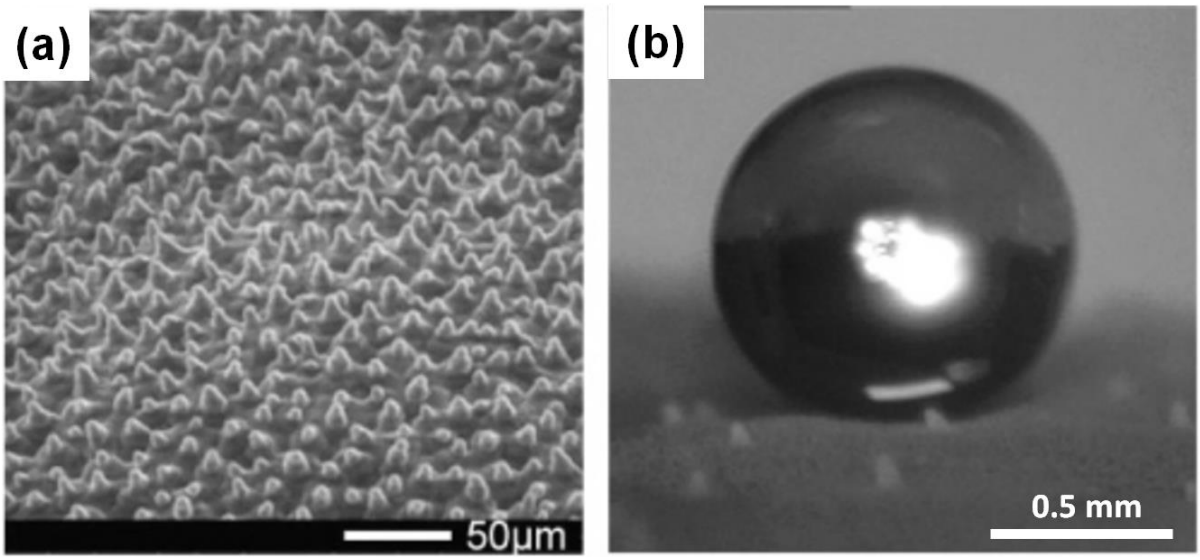


Figure 1-8. SEM image of the surface of a lotus leaf (a) and a water droplet on the lotus leaf surface (b) [11].

1.4 EXISTING FABRICATION METHODS

Some of the existing technologies used to generate nanostructures include interference lithography, etching, and bio-templates [1, 6, 11-13].

1.4.1 Interference Lithography

Linear gratings can be generated using interference lithography, where two coherent argon ion laser beams are interfered to create a sinusoidal intensity pattern. Gombert, et al. demonstrated this technique using glass panes coated with standard positive photoresist that are then exposed to this interference pattern resulting in linear grating periods from 205 to 301 nm [6]. To achieve subwavelength nanopillars, the photoresist was exposed to the light source twice, with the second exposure orthogonal to the first exposure (cross-grating). This interference lithography technique is limited to areas of 1200 cm^2 in a single exposure. Nickel replicates are produced from the master mold which can be used in the subsequent embossing process (Figure 1-9(a)). Using the nickel replica, acrylic glass (PMMA) and polycarbonate (PC) are hot embossed to transfer the master features. The parameters

reported state a pressure of 2-4 MPa and temperatures of 150-210 °C. The processing area was limited to 225 cm^2 due to the embossing tool. The 300 nm, cross grated feature replicates are shown in Figure 1-9(b).

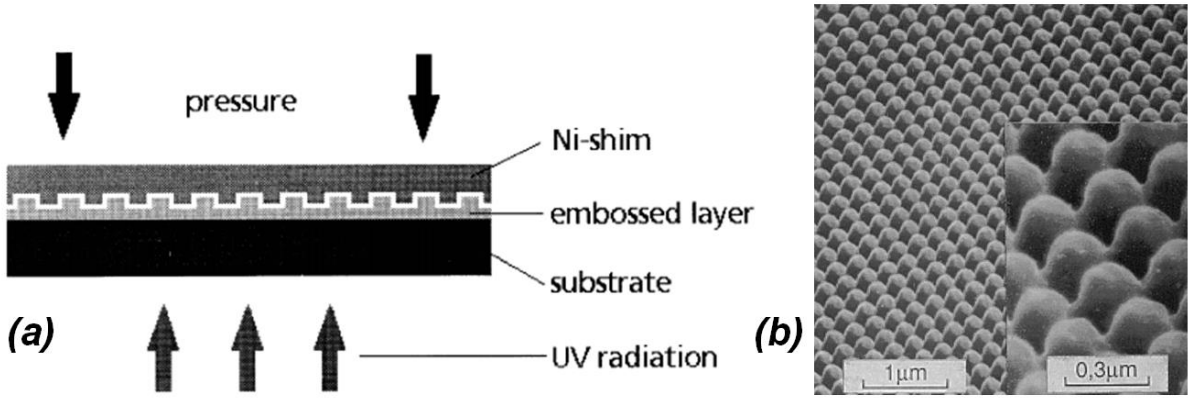


Figure 1-9. Schematic of embossing process (a) and resulting crossgrated structures (b) [6].

1.4.2 Biotemplates

Generating subwavelength structures via bio-templates involves using a biological sample containing the desired features (moth eye, cicada wing) as the mold. For example, Xie, et al. used a cicada's wing as a biotemplate to produce 450 nm feature replications [1]. First the wings were sonicated in dionized water for 15 minutes to remove contaminants and then sonicated in acetone for 20 minutes to remove stains. After the acetone, the wings were sonicated again in dionized water to remove residual acetone and then blow-dried with nitrogen. Once the wings were cleaned, a thin layer of gold was deposited on the specimen using a thin-film coating system which results in a gold mold of the negative of the features. The gold template structures were transferred to a PMMA film by pressing the mold into the substrate and then heating it in an oven at 90 °C for 30 minutes. The gold mold is then removed leaving the wing features in the PMMA. Figure 1-10 shows both the cicadas wing (a) and the resulting PMMA replicates made using bio-templating (b).

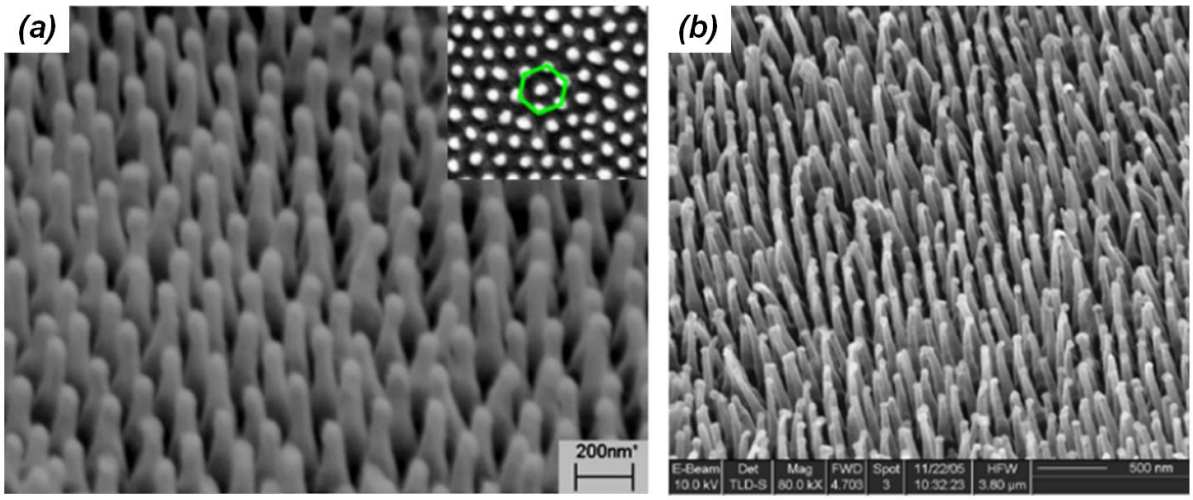


Figure 1-10. SEM images of the cicadas wing (a) and the PMMA replicate (b)[1].

1.4.3 Etching

Non-lithographic etching techniques can be used to produce moth eye structures on a 12" silicon wafer. Sun, et al. used monolayer silica colloidal crystals with non-close-packed structures that were created using a spin-coating technique. The resulting crystal arrays can be used as etching masks on the surface of a silicon wafer [13]. Once an array of silica particles is positioned on the wafer, reactive-ion etching (RIE) is used to remove the silicon exposed around the particle mask. After the RIE is complete, the silica particles are removed using a hydrofluoric acid wash, revealing the array of silicon nanopillars resembling moth eye structures. Figure 1-11 shows a schematic of the process to generate moth-eye structures.

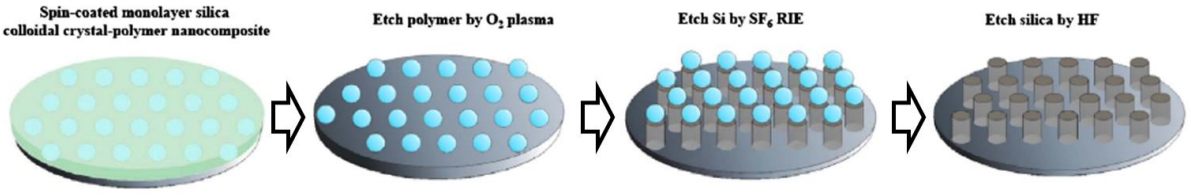


Figure 1-11. Procedure for fabricating moth-eye structures on a silicon substrate [13].

The height of the features can be adjusted by changing the RIE time to allow more material to be removed. Figure 1-12 shows SEM images of silicon moth-eye structure arrays for RIE times of both 6 minutes (a) and 9 minutes (b).

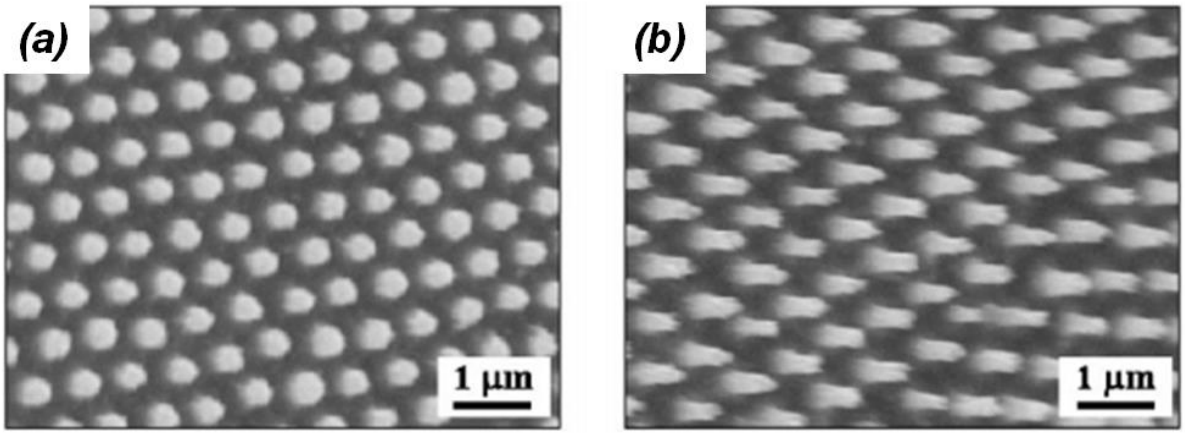


Figure 1-12. Silicon feature array after etching for a total of 6 minutes (a) and 9 minutes (b).

1.4.4 Summary of Fabrication Techniques

There are several methods of creating sub-micrometer features currently used by researchers. These methods include lithography, bio-templates and etching, which yield features that produce antireflective and non-wetting surfaces. However, while existing techniques are successful in replicating bio-inspired patterns, they often are long in duration and require several manufacturing steps to complete (some steps in the lithographic process can take up to 11 hours [14]). Many of the techniques reviewed also are restricted in the area of features produced at a single time. For instance, etching a 12" silicon wafer may take 9 minutes, but each wafer must undergo several steps prior to the etching process.

1.5 NANOCOINING DEFINITION

Nanocoining is a mechanical approach to rapidly create surface structures using a small diamond die (20 x 20 µm) containing an array of 6400 nanofeatures machined with a focused

ion beam (FIB). This die is pressed into a diamond turned surface, leaving a field of sub-micrometer features (Figure 1-13).

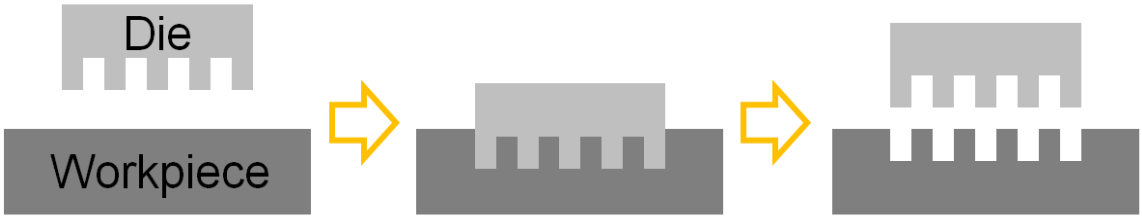


Figure 1-13. A prefabricated diamond die is pressed into a diamond turned surface to create desired features.

To create a continuous area of features, the individual die indents must be tiled together. One way this is done is by indenting the surface of a rotating drum on a diamond turning machine (DTM) as shown in Figure 1-14(a). The die must be held by an actuator that pushes it into the surface to replicate the die features while moving tangentially at a speed that matches the surface of the drum during contact to avoid smearing of the individual features. The drum surface speed is controlled by the spindle motor rpm and the radius of the drum. In addition to matching the drum speed during impact, the die must land one die width ahead of its position on the last cycle. This defines an "upfeed" distance that sets the relationship between the frequency of the die motion, its size and the surface speed of the drum. The "crossfeed" motion of the linear axis holding the die must advance along the drum at one die-width per revolution.

The registration problem is lessened due to the independent requirements for upfeed and crossfeed of the die. There is no requirement that the dies line up on each rotation of the drum only that the surface is fully covered by features. In fact, small overlap between successive die indents is desirable to avoid regions without features which would produce visible diffraction at a frequency related to size of the die. Once the drum is uniformly

covered with the desired structures, it is used as a mold in a roll-to-roll process to rapidly generate features on a pliable medium (Figure 1-14(b)).

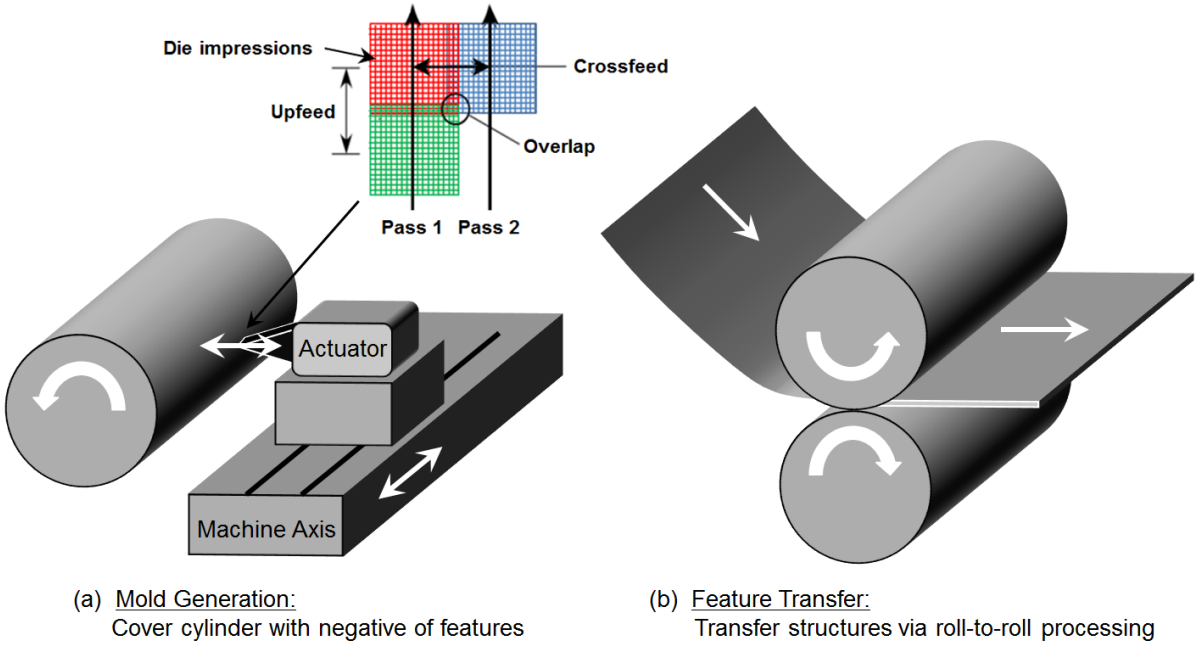


Figure 1-14. Nanocoining of subwavelength surface features on a cylindrical workpiece (a) and roll-to-roll replication using indented roller.

Since the die is small (20 x 20 μm), indents must occur quickly (40 – 50 kHz) to cover 1 m^2 in reasonable amounts of time (14-18 hours). Generating tiled arrays of indents on the mold at 40 kHz requires the surface to move at a velocity such that it travels the length of the die (20 μm) during every period of the actuator oscillation (25 μsec). The work surface is moved a constant rate based on the actuator frequency and die size and the result will be a succession of indents precisely following the previously formed features. When indenting into a translating workpiece, the die must match the tangential velocity of the drum while the die is in contact with the surface. If this condition is not achieved, the shape of the die features will be distorted or smeared.

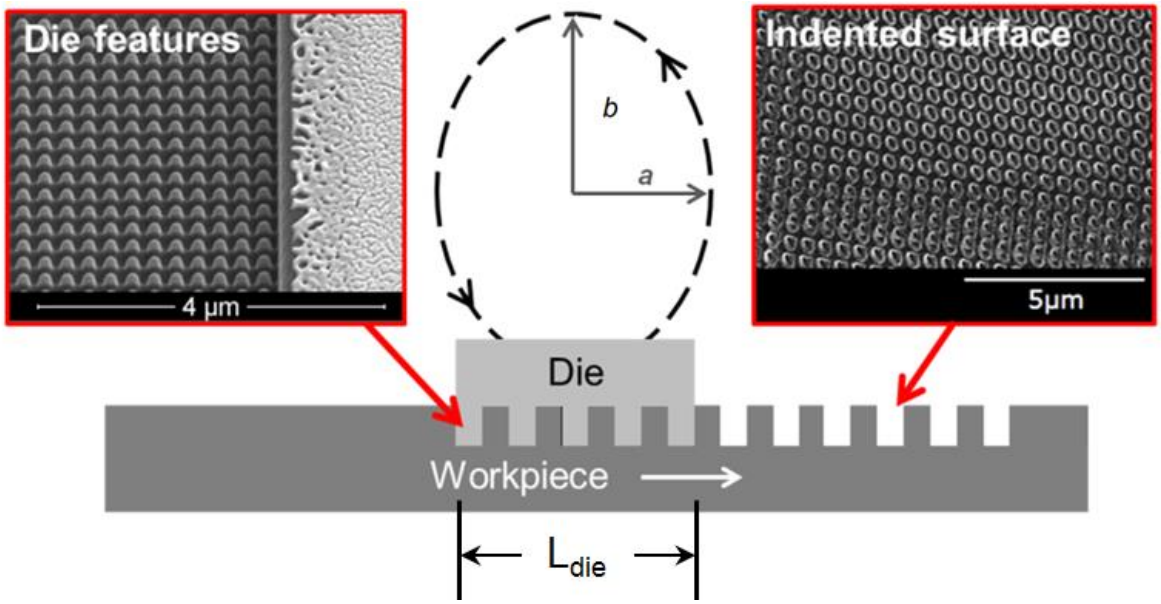


Figure 1-15. Vertical and horizontal dimensions of elliptical indenting path.

To minimize the distortion caused by a velocity mismatch, the die must be moved in a 2D path that allows the die to be pressed normal in to the surface while simultaneously matching the tangential speed of the workpiece. This goal can be achieved by moving the die in an elliptical path as shown in Figure 1-15 but only approximately. If the workpiece is moving at a constant speed, the die actuator motion can be designed to match that speed at the bottom of the elliptical path in Figure 1-15. However, there has to be some small indentation distance so the die and workpiece will contact over some finite time. This means that an elliptical path cannot achieve a pure indentation motion. However, if the ellipse is taller than it is wide and the indentation distance is on the order of 100 nm, the smearing of individual die features will be less than 10 nm. While not perfect, the elliptical die motion is the only reasonable die path for the 2D actuators developed at the PEC and elsewhere.

Assuming the maximum horizontal speed of the die and the workpiece surface are equal during contact, the required dimension of *a* is a function of the die size and can be expressed mathematically in the following form [15]:

$$a = \frac{L_{die}}{2\pi} \tag{1-4}$$

where L_{die} is the length of the die measured in the work surface's direction of motion. If the required dimension a cannot be achieved with the actuator, a smaller indenting ellipse can still be used by slowing the speed of the workpiece. This, however, results in overlapping of successive indents and can increase the time needed to cover a given area. The spindle speed is set to match the measured dimension a using the expression:

$$\omega_{spindle} = \frac{60\, f a_{meas}}{r} \tag{1-5}$$

where

$\omega_{spindle}$ = spindle speed in RPM

f = actuator frequency in Hz

a_{meas} = horizontal dimension of the die path in mm

r = radius of the cylindrical workpiece in mm.

If the amplitude of a_{meas} is too large, the spindle will have to rotate too quickly which will result in spaces between the indents. Therefore, the following requirement must be met to use Equation (1-5) for generating uniform fields of features:

$$a \geq a_{meas} \tag{1-6}$$

The crossfeed speed fr is set based on the spindle speed $\omega_{spindle}$ and the width of the die W_{die}, where:

$$fr = \omega_{spindle} W_{die} \tag{1-7}$$

The ellipse dimension vertical to the worksurface *b* (shown in Figure 1-15) dictates the time it takes for the die to enter and exit contact with the part, whereby increasing the vertical amplitude decreases the amount of indentation time between the die and workpiece. This relationship between amount of smear and the vertical dimension b is shown in Figure 1-16.

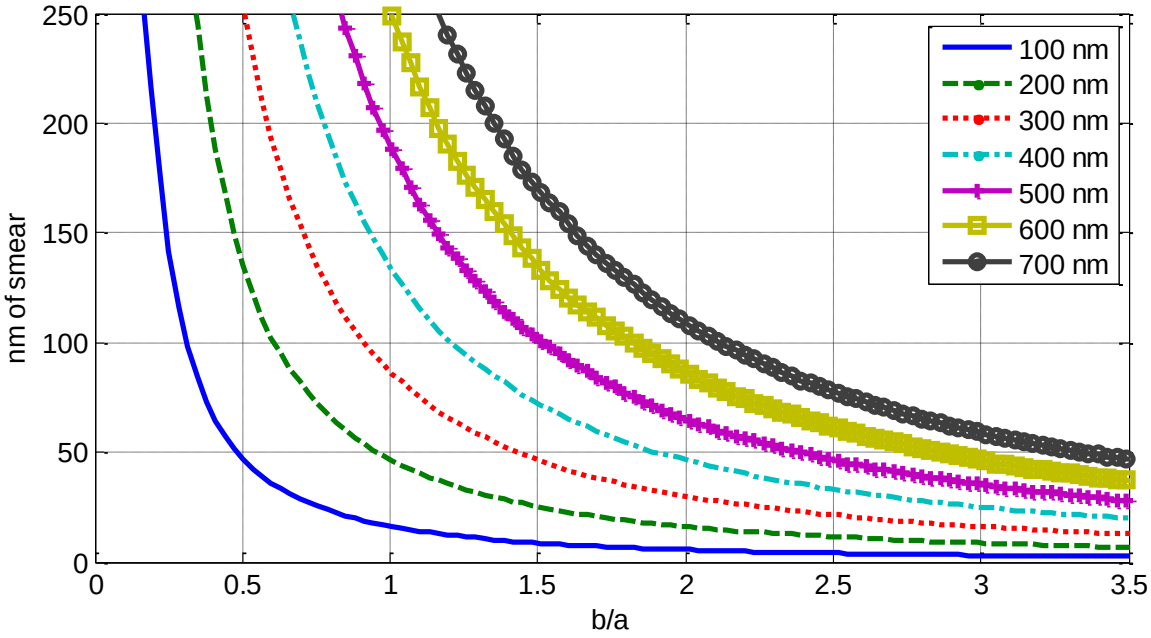


Figure 1-16. Ellipse dimensions versus estimated feature smear.

To keep the smearing of features under 10 nm for indent depths of 200 nm, the b/a ratio of the ellipse must be approximately 3. This means that for well-formed indents, the die must be moved in a 3 x 9 μm ellipse.

1.6 PROBLEM STATEMENT

The goal of this work is to develop an actuator capable of indenting a diamond die covered with nanofeatures into a moving mold surface, resulting in a continuous uniform array of indentations on the surface of the mold. Since the die is small (20 x 20 μm), the indenting must occur quickly (40-50 KHz) to uniformly cover large surface areas in reasonable amounts of time (less than 12 hours). To avoid the distortion that occurs due to velocity

mismatch between the die and worksurface, the indentation path of the die must contain a horizontal velocity component that matches the speed and direction of the workpiece. The horizontal motion, combined with the vertical stamping motion, results in an elliptical die path. This elliptical path must be designed to meet specified dimensions to minimize smearing of the features during die-worpiece contact. To ensure uniform registration of indents over the entire surface of the workpiece, the actuator must remain synchronized with the surface motion.

The work herein will include:

- A design methodology for an ultrasonic 2D actuator capable of performing nanoindentations
- Applications of the design methodology to the construction of working prototypes
- Design and implementation of an actuator controller to ensure consistent, ultrasonic elliptical motion
- Additional discussion on process control to synchronize the actuator motion with the machine axes and ensure uniform indent registrations
- Ultrasonic indenting results on various substrate materials

2 BACKGROUND

Machining and tooling processes that require relatively large actuation at ultrasonic frequencies (micrometers at more than 20 kHz) often utilize resonant motion. These actuators are designed to exploit the amplification in displacement that occurs at system resonance. These actuators are designed to possess a high Q-factor, or low damping coefficient, to maximize the amplitude response at the expense of a narrow bandwidth. Depending on the application, resonant actuators can be designed to operate in one, or several, directions of motion simultaneously.

2.1 SINGLE-DOF ULTRASONIC DEVICES

Linear ultrasonic transducers are generally designed such that the length is a half-wavelength of the speed of sound through the material. When excited, this results in a standing wave in the material which produces harmonic motion. The wave length λ is a function wave speed and frequency, and can be expressed as the following:

$$\lambda = \frac{c}{f} \tag{2-1}$$

where c is the wave speed in a given material and f is the frequency. Wave speed is related to the material the wave is propagating through by the following relationship:

$$c = \sqrt{\frac{E}{\rho}} \tag{2-2}$$

where E and ρ are the Young's modulus and density of the material, respectively.

The standing wave motion can be visualized by considering a simple 1-D, half-wavelength beam model as shown in Figure 2-1.

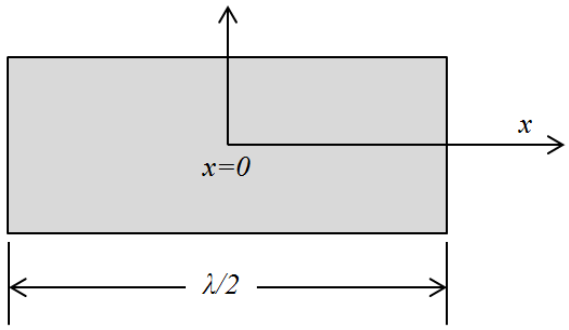


Figure 2-1. Half-wavelength uniform beam.

When the beam is excited, the displacement μ of a particle on the beam in the x-direction can be represented by the wave equation, given as [16]:

$$\frac{\partial^2 \mu}{\partial t^2} = c^2 \frac{\partial^2 \mu}{\partial x^2} \tag{2-3}$$

where x is the length from the displacement node in the center of the length along the axis out towards the ends, t is the time and c is the speed of through the medium. Considering λ is the wavelength, solving the PDE for displacement along beam length $\lambda/2$ yields:

$$\mu = \mu_0 \sin(kx)\sin(\omega t) \tag{2-4}$$

where k is the wave number ($k = f/c$), μ_0 is the vibration amplitude and ω is the angular frequency of vibration. Using Hooke's law, stress can be represented as in Equation (2-5) where E is the modulus of elasticity [16]:

$$T = E\mu_0 k \cos(kx)\sin(\omega t) \tag{2-5}$$

The expression for 1-D particle displacement axially along an ideal beam given in Equation (2-4), along with the stress expression in Equation (2-5), can be used to plot the displacement

and stress along the length of the vibrating beam over one period of oscillation as shown in Figure 2-2. In Figure 2-2(a), it can be seen that either end of the transducer are oscillating between relative maximums while the center (shown as 0) is stationary as a function of time. Likewise, Figure 2-2(b) shows the stress due to one cycle of oscillation where the maximum is occurring in the center, while the ends of the beam show zero stress due to unconstrained boundary conditions.

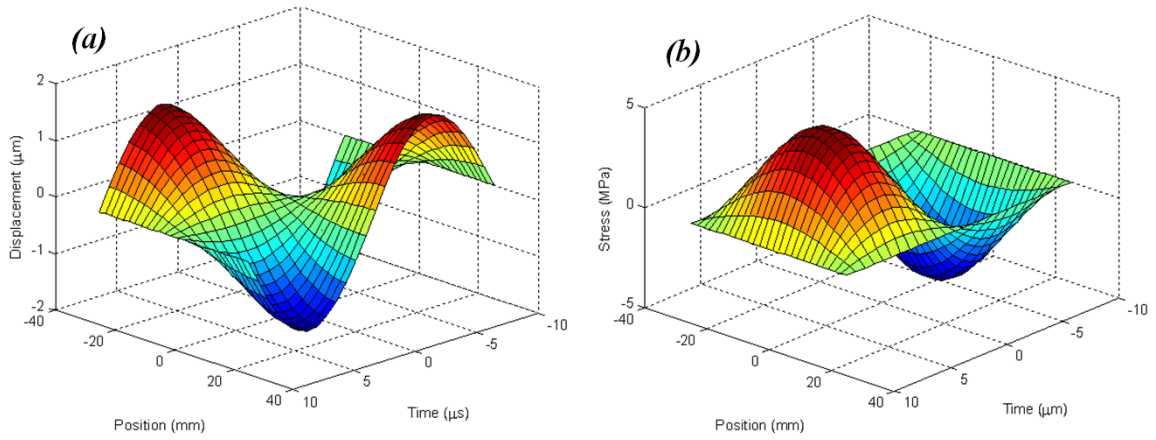


Figure 2-2. Displacement (a) and stress (b) of the 1-D vibrating beam model in the x-direction.

2.1.1 Bolt-clamped Langevin Type Transducer

Bolt-clamped Langevin type (BLT) transducers are large amplitude, ultrasonic vibration sources commonly used in industrial applications such as plastic-welding, cleaning and machining [17]. They are half-wavelength resonant devices that are sized according to the desired operating frequency, typically ranging between 20 and 100 kHz [18]. A typical BLT transducer consists of piezoelectric rings sandwiched between two metal masses. Electrode leads are positioned between the rings which are then powered by a sinusoidal voltage signal. The structure is fastened together using a bolt and nut, also serving to apply a prestress on the piezo material. Since most piezoelectric ceramics have compressive and tensile fatigue strengths of approximately 125 MPa and 25 MPa, respectively, a 50 MPa prestress is usually

desirable to prevent the material from exceeding the fatigue limits [16]. The prestress on the piezo prevents damage that may occur during large resonant vibrations of the transducer and increase the longevity of the material. A schematic of a BLT transducer is given in Figure 2-3.

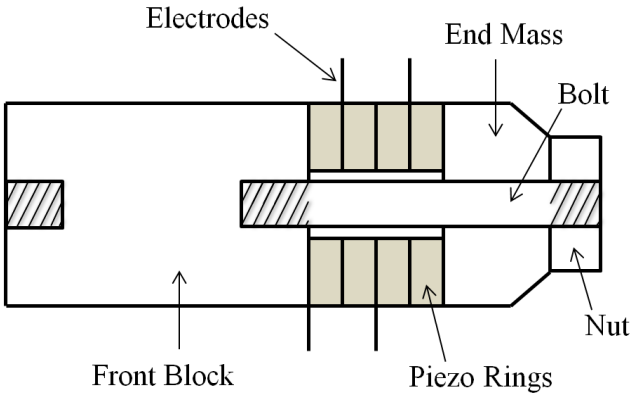


Figure 2-3. Schematic of a BLT transducer.

BLT's operate at resonance and are designed to have large resonant amplitude responses. This response is characterized using the mechanical quality factor (*Q)*, which describes system behavior during resonant excitation. The length is constrained to half a wavelength of the pressure waves through the material median. This results in a standing wave when the input signal is equal to a longitudinal mode frequency. Because of this standing wave behavior, resonant maximum deflection occurs at either end of the transducer while a displacement node is located near the center. In contrast, the maximum stress occurs near the center at the displacement node while minimum stress occurs at either end (as illustrated in Figure 2-2). To avoid unwanted damping and additional resonances due to a mounting structure, BLT systems are most often mounted at the displacement node.

2.1.2 Existing Applications

Ultrasonic transducers are used for many machining applications and offer advantages over traditional methods. These applications, including ultrasonic welding, ultrasonic drilling and vibration assisted machining (VAM) are discussed below.

2.1.2.1 Ultrasonic Welding

Traditional welding techniques, such as fusion welding, are based on molten metal transfer, where two metals are joined by applying enough heat that the materials melt and flow together. However, this method has several limitations such as the occurrence of solidification defects (porosity, inclusion, incomplete fusion, etc.) and undesirable material transformations (softening, embrittlement, etc.) [19]. In instances where these limitations must be avoided, ultrasonic welding is often used.

Ultrasonic welding offers advantages over traditional fusion techniques. It is capable of joining not only soft aluminum foil, but also copper, nickel, titanium, zirconium and other alloys. Joining dissimilar metals also showed good bond strength, as well as joining metals to non-metals such as metal leads to germanium, silicon and glass [20]. Ultrasonic welding can be used either in atmosphere or vacuum, which adds to its versatility [21]. Cleaning the two workpieces to be welded is also not necessary since the friction at the interface pulverizes and removes contaminants and oxide layers [20].

This type of welding involves using an ultrasonic transducer to provide vibration energy and heat that will bond two materials. The vibration direction can be either orthogonal or normal to the compression force and mechanical amplifiers (sonotrode horns) are often used to increase the vibrational amplitude of the transducer [20, 22]. Shown in Figure 2-4 are the two types of ultrasonic welding configurations. Both setups contain the same components, namely a vibration source, a mechanically-amplifying horn, an anvil or holding plate which acts as the work surface and a welding force normal to the joint. These ultrasonic transducer systems generally operate between 10 and 100 kHz and generate between 10 and 100 μm of displacement at the tip of the sonotrode horn [20, 23].

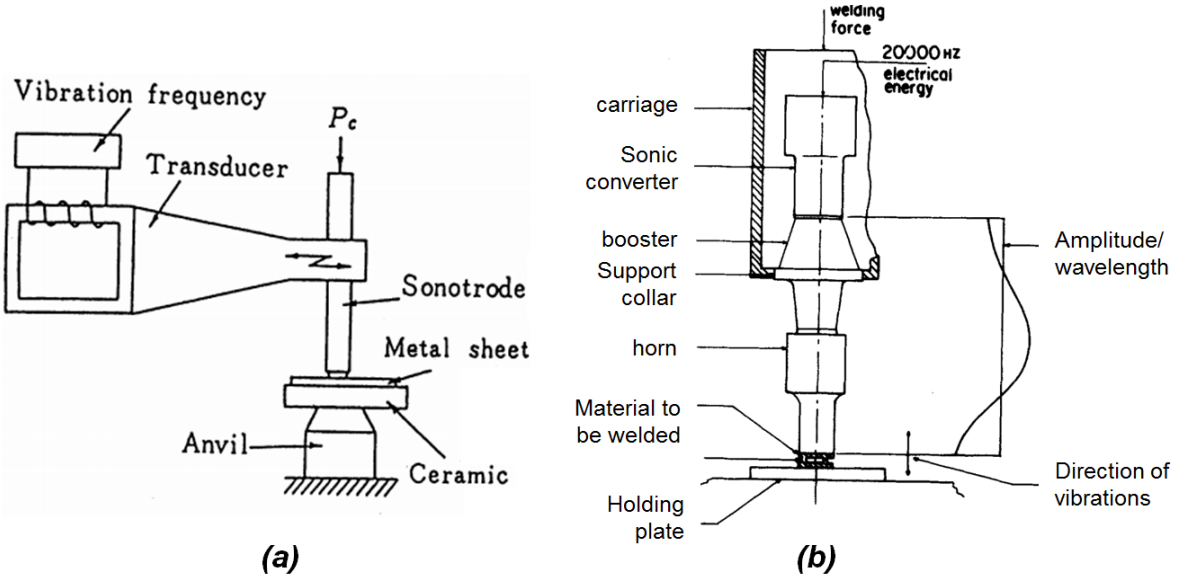


Figure 2-4. Ultrasonic welding schematics for vibration orthogonal to workpiece [21](a) and normal to workpiece [22](b).

Ultrasonic welding is a solid-state "cold" joining process where the heat generated is typically only 40% of the melting temperature of the materials. This joining process takes place in three stages. First, the surfaces are pressed together by the normal pressure on the joint. Secondly, the atoms of the joining surfaces are activated and the electrons are exchanged between the two workpieces. Lastly, atoms along the weld and in neighboring areas are diffused on a microscopic scale, resulting in a strong bond [23]. When welding thermoplastics, melting of the material does occur very locally at the bond interface. When the ultrasonic vibrations begin, the surface asperities at the interface heat very rapidly because stresses at the contacting asperities are much greater than the stresses in the material further away from the interface. This causes the interface to heat rapidly, causing the two thermoplastic polymers to soften and flow into each other to produce a strong bond [22].

2.1.2.2 Ultrasonic Drilling

Ultrasonic drilling is a non-thermal, non-chemical process that results in no metallurgical, chemical or physical property changes in the workpiece. Because of this, it offers virtually

stress-free machining which allows brittle materials such as semiconductors, optical glasses and ceramics to be easily formed into increasingly intricate shapes and workpiece profiles [24]. In contrast to traditional drilling techniques, studies have shown that the use of ultrasonic vibration when drilling leads to a reduced cutting forces, longer tool life, and requires thrust force [25].

Two types of ultrasonic drilling are common in industry: traditional ultrasonic machining rotary ultrasonic machining [24]. The traditional method uses abrasive particles suspended in a fluid, where a grit-loaded slurry is circulated between the workpiece and tool that is vibrating at 10 to 20 μm and 20 to 50 kHz. The vibrational motion is generated by a linear ultrasonic transducer attached to a sonotrode horn that, while under light pressure, transmits the vibrational motion to the slurry which abrades the workpiece to create a negative image of the tool form [24].

The rotary ultrasonic drilling process is a combination of lateral ultrasonic vibration and the rotational motion of traditional drilling. The lateral motion comes from a linear ultrasonic transducer attached to an amplifying horn to increase the amplitude at the tip. The drill bit is fixed to the end of the horn. One common arrangement has the ultrasonic transducer held in place while the workpiece is rotated as shown in Figure 2-5. A variation of this involves rotating the ultrasonic transducer such that it provides both the lateral motion and rotational motion about a fixed workpiece [26].

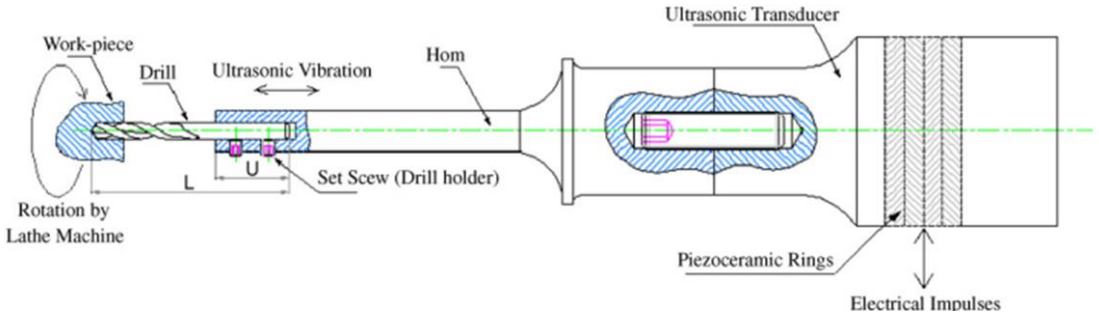


Figure 2-5. Schematic for an ultrasonic drilling setup [25].

2.1.2.3 Vibration Assisted Machining

The addition of a high frequency, low amplitude linear vibration to a cutting tool is called vibration assisted machining. When a cutting tool is vibrating, it periodically loses contact with the chip, reducing average cutting force and tool temperature. This reduction in contact time between the tool and chip results in near-zero burr compared to conventional machining and increased tool life [27]. An illustration of this VAM process is shown in Figure 2-6, where the tool travels into the workpiece forming a chip and then reverses its motion to come out of contact with the chip before contacting the chip again and repeating the process.

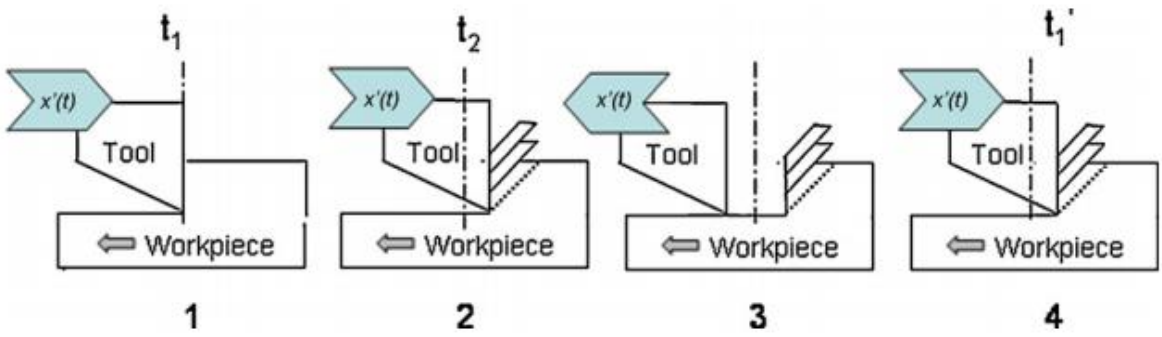


Figure 2-6. VAM process diagram [27].

Typical ultrasonic VAM devices consist of a linear transducer (BLT), and amplitude-amplifying (ultrasonic concentrator) horn to generate the necessary vibratory motion. Most VAM devices operate at 20 – 40 kHz and produce 3 – 20 μm of amplitude [27]. An example of an ultrasonic cutting tool is shown in Figure 2-7, where it is designed to possess high rigidity such that large cutting forces exerted on the tool will not change the vibration direction. The tool utilizes a linear ultrasonic transducer and horn to produce the cutting motion; however, is mounted at two node locations along the length of the device. This mounting technique, in contrast to traditional VAM tool designs that use a single mounting flange that can more easily deflect under large cutting forces, increases the rigidity and helps ensure that the device is vibrating in the desired direction during the cutting process. This

actuator demonstrated precision finishing cutting on a high-hardness workpiece with a tool life of about a 36 times more than conventional cutting [28].

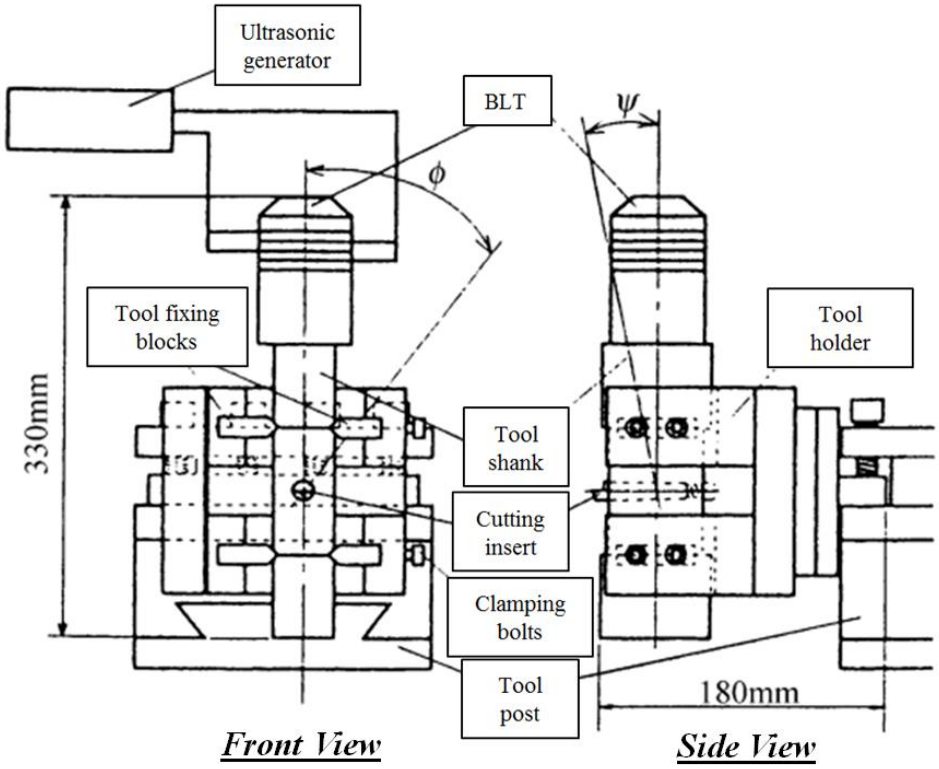


Figure 2-7. Front and side view of an ultrasonic cutting tool system with a BLT transducer mounted at two node point to increase mount stiffness [28]

2.2 MULTI-DOF ULTRASONIC DEVICES

Ultrasonic devices that simultaneously vibrate in more than one direction are also used in industry for advanced machining and welding operations. Like the single degree of freedom devices, they operate at a single resonant frequency that is determined by the size of the actuator. One way to achieve this complex vibration is to use a geometry cross-section that is symmetrical in two directions, such as a beam with a square cross-sectional area. Figure 2-8 illustrates a bending mode shape where the tip motion is directed in the x-direction. Since the beam is symmetrical in both the x and the y directions, the same bending mode will

occur at the same frequency in the y-direction. If these two modes are excited out-of-phase, the combined motion will results in an elliptical path in the x-y plane.

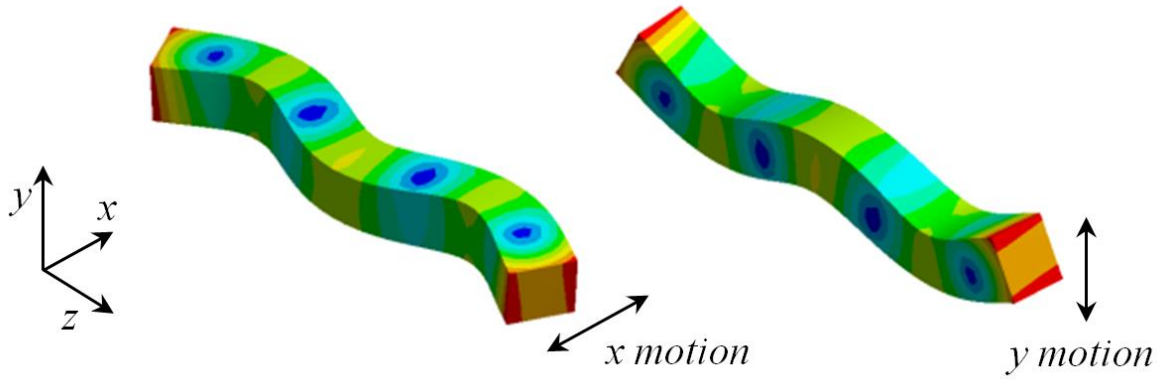


Figure 2-8. Combined-mode motion in x-y plane.

Another method of generating two-dimensional resonant motion is to design the geometry such that a longitudinal mode shape and a bending mode shape occur at the same frequency. Since the longitudinal mode will produce vibrational motion in the z-direction, and the bending mode in the y-direction, combining these two modes will results in an elliptical path in the y-z plane (Figure 2-9).

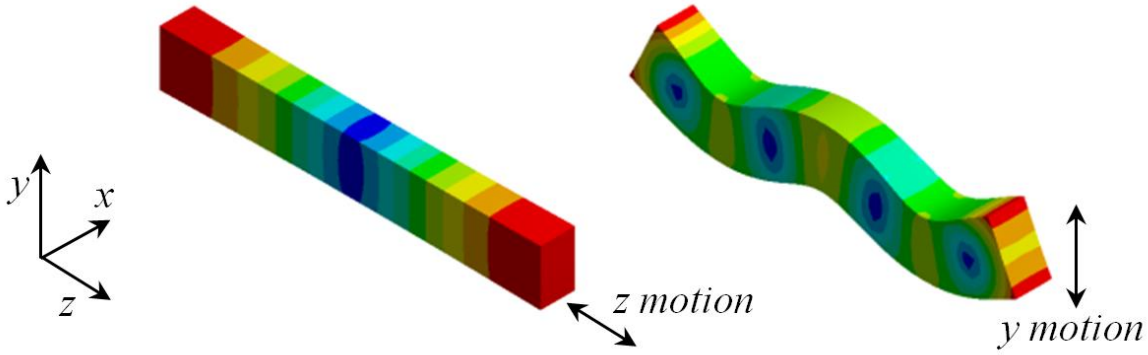


Figure 2-9. Combined-mode motion in the y-z plane.

While these two concepts are common in multi-DOF resonant actuators, designs have also utilized a combination of both arrangements in Figure 2-8 and Figure 2-9 to generate three-dimensional motion [29].

2.2.1 Ultrasonic EVAM

Elliptical vibration-assisted machining (EVAM) is an extension of the VAM process described in Section 2.1.2.3 that involves using a small amplitude, high frequency elliptical cutting path such that the cutting tool enters the workpiece periodically to remove material. Due to the elliptical tool path, the tool moves cyclically out of the workpiece along with the chip (Figure 2-10). Because the relative speed between the chip and tool is small, frictional effects on the tool are small, which results in an increased shear angle and reduces the cutting force, reducing temperature and prolonging tool life [27, 30].

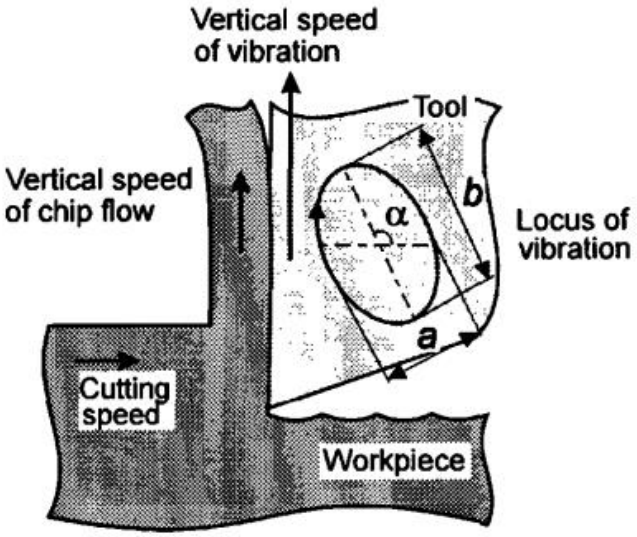


Figure 2-10. Schematic of EVAM.

To increase the machining rate and avoid audible high frequency noise, some EVAM tools are designed to operate at ultrasonic frequencies (> 20 kHz). One example of an elliptically-vibrating resonant actuator designed to operate at 20 kHz is shown in Figure 2-11. The actuator body is symmetric about two bending mode directions, allowing for 2-D elliptical

motion when excited at the resonant frequency. Two sets of opposing piezoelectric elements are excited with sine wave voltages out-of-phase to produce the elliptical path in the plane transverse to the length of the actuator. To avoid unwanted damping, the structure is mounted at vibration nodes [30, 31].

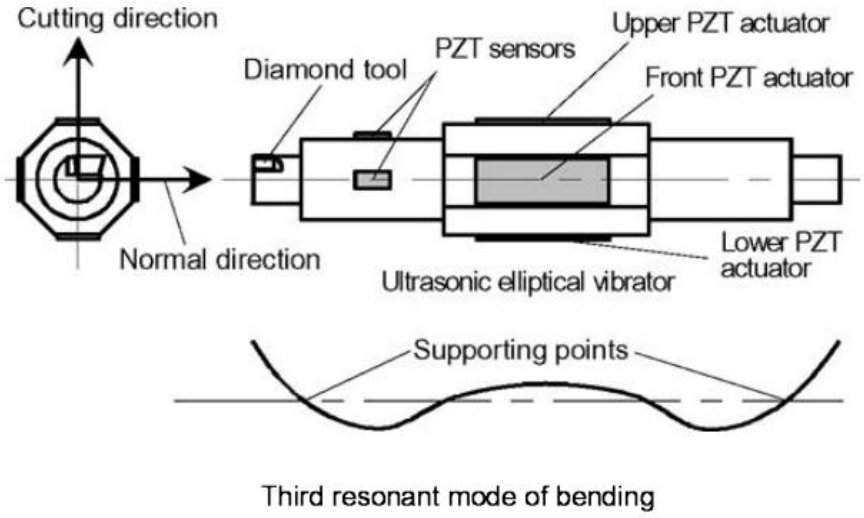


Figure 2-11. Ultrasonic bending mode elliptical vibrator for EVAM [31].

Another design proposed by Li, et al. [32] uses a single piezo actuator and relies on an asymmetrical design to produce elliptical tool motion. This design is based on a linear piezo transducer and sonotrode horn arrangement that is designed to operate in the longitudinal resonance mode. The position of the tool, which is mounted off to one side of the tip of the horn, results in a longitudinal-bending motion. When the linear transducer portion is excited, the tool tip vibrates in an elliptical path due to the off-center mass of the tool. The asymmetrically-mounted tool tip is used as a tuning mechanism where the size can be adjusted to change the vibrational response of the actuator until the desired motion is obtained. This actuator was designed to operate at 22,500 kHz and can achieve an elliptical path of 16 and 2 μm in the longitudinal and bending directions, respectively. An illustration of this device is given in Figure 2-12.

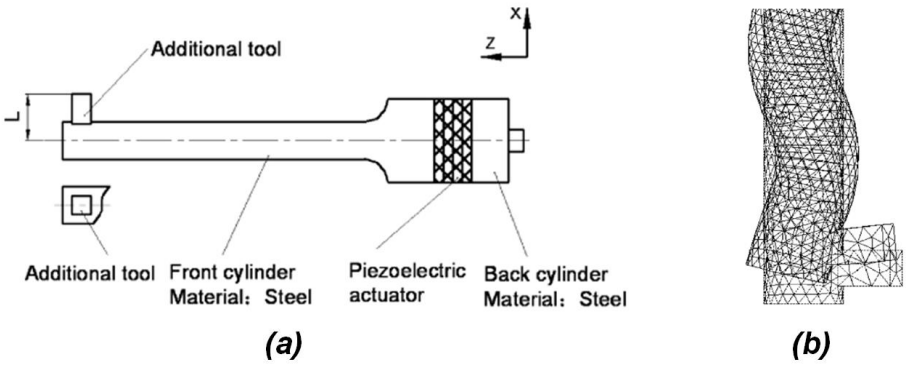


Figure 2-12. Asymmetric actuator schematic (a) and longitudinal-bending compound mode (b) [32].

A more complex ultrasonic EVAM tool capable of 3 degrees of freedom was developed by Shamoto, et al [29]. This tool was proposed as an alternative to a traditional ball end mill for use in machining hardened steel for dies and molds. Compared to a traditional 2-D elliptical tool that operates in a fixed plane, the 3-D motion is necessary when machining sculptured surfaces such that the elliptical tool path plane can be adjusted to follow the contour of the surface. This actuator design is based on combining both the bending mode behavior shown in Figure 2-11, and a longitudinal mode to produce the third degree of freedom. An illustration of this actuator is shown in Figure 2-13(a).

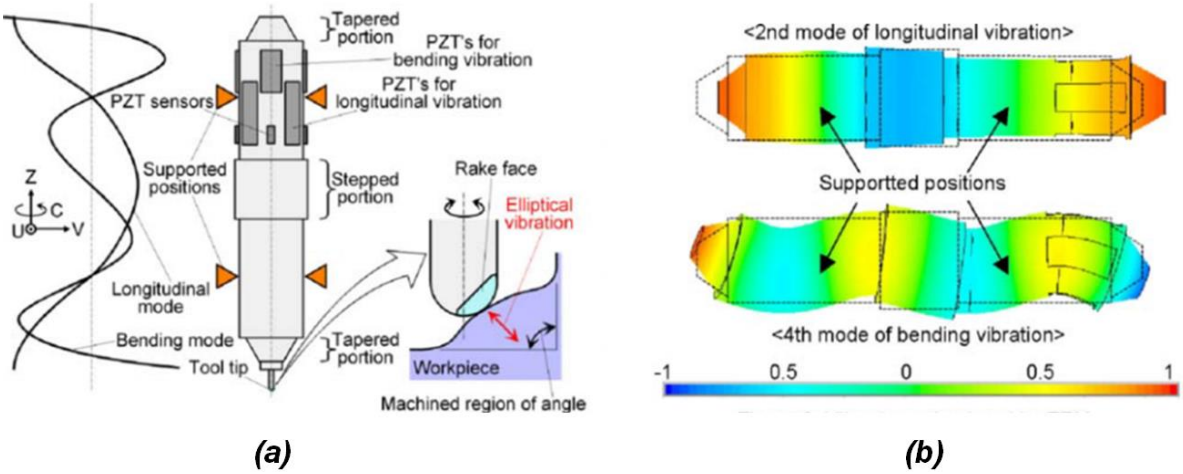


Figure 2-13. Schematic of 3-D ultrasonic EVAM tool (a) and vibration modes (b) [29].

This 3-D vibration tool is sized such the 4th bending mode and 2nd longitudinal mode occur at 34.4 kHz (Figure 2-13(b)), allowing both modes to be excited with the same frequency. Since the design is essentially axisymmetric, the bending mode can occur in several directions at the same frequency. Piezo elements are mounted on the side of the actuator to excite the bending mode in two orthogonal directions (piezo elements mounted 90° from each other around the periphery of the actuator) and longitudinal mode independently, which allows for control over the tip path. This actuator is capable of 15 μm amplitude in the bending directions and 12 μm in the longitudinal direction at 34.4 kHz.

2.2.2 2-D Ultrasonic Welding

Elliptical vibrators are also used in ultrasonic welding applications. The process mechanisms are identical to the 1-D case discussed in Section 2.1.2.1, however; with complex vibration welding systems it is possible to weld thicker specimens with a larger, more uniform weld area [33]. An example of a 2-D ultrasonic welding device was designed by Tsujino, et al. [34], and utilizes a linear BLT-type transducer. A specially designed horn allows a portion of the lateral vibration from the transducer to be converted into torsional motion. The combination of longitudinal and torsional motion is transmitted to the weld specimen through a disk-type welding tip which results in an elliptical velocity profile. This apparatus is shown in Figure 2-14.

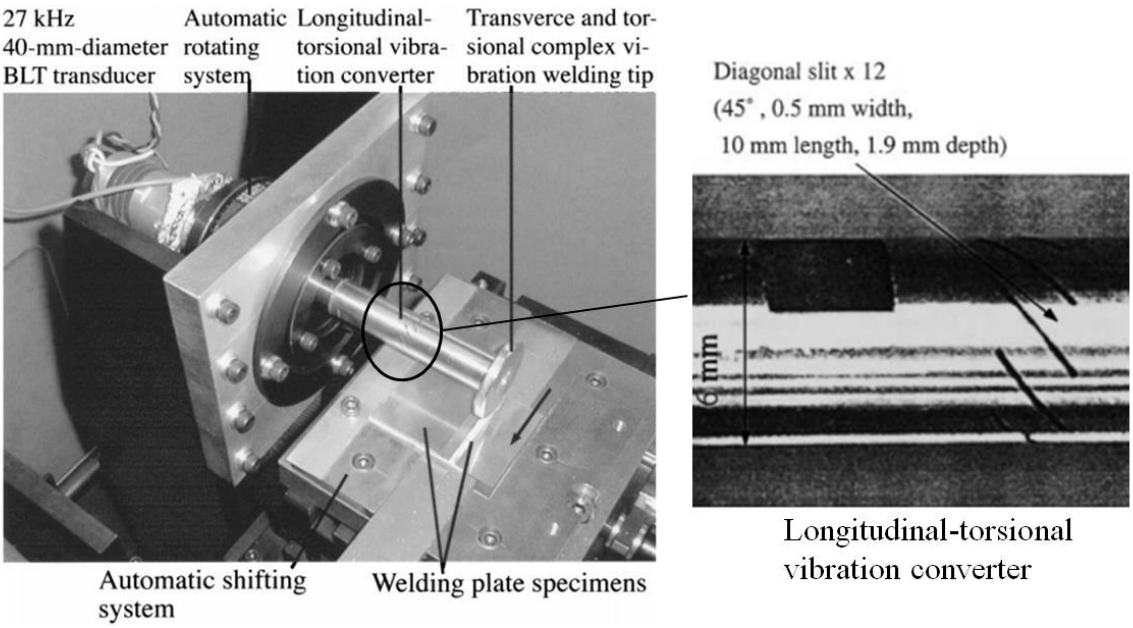


Figure 2-14. Longitudinal-torsional vibrator for ultrasonic welding [34].

The longitudinal-torsional converter consists of 12 diagonal slits on a stepped-down horn of the transducer between the BLT and the welding tip at a 45° angle relative to the longitudinal direction of the actuator. These slits convert the linear longitudinal motion into a torsional motion, resulting in the two velocity components. This design uses different converter diameters (24 or 20 mm), which results in different operating frequencies (27 and 40 kHz, respectively). The tips have enough area to weld 6-8 mm^2 and are capable of welding aluminum plate specimens up to 1 mm thick with weld strengths almost equal to the aluminum specimen strength [34].

Another application of two-dimensional, resonant actuators for welding is in high frequency wire bonding systems. These systems are useful in the direct welding of semiconductor tips and packaging of electronic devices [35]. An example of a device to perform this task is proposed by Tsujino [35], where it uses vibration rod that vibrates in two bending directions simultaneously. The vibration comes from two linear piezo-driven transducers that are mounted in the center of the rod, 90° apart from each other. This orientation allows the two

vibrators operating in the longitudinal direction to excite the bending mode of the beam in two directions. For this application, the operation frequency must be very high (350-980 kHz), however; the concept is scalable to lower frequencies. An illustration of this complex transverse vibration rod design is given in Figure 2-15.

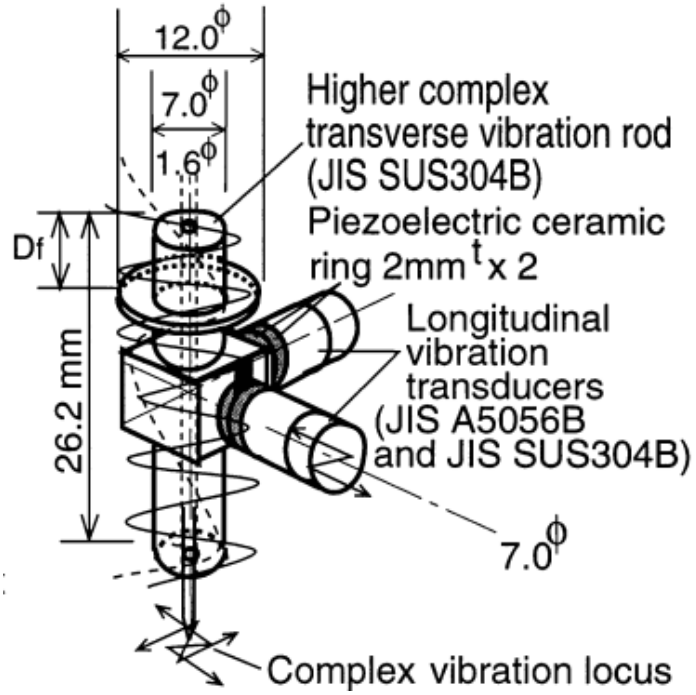


Figure 2-15. Complex transverse vibration rod design [35].

2.2.3 Ultrasonic Motor

In situations where high precision and accuracy positioning is required, linear ultrasonic piezoelectric motors are often implemented. These motors are 2-D resonant devices that use one vibrational component to provide a normal force, and another component to generate a friction-based driving (or thrust) force. This combined ultrasonic motion creates a friction-based drive system between the fixed motor and a moving slideway that provides high velocity (1 m/s) and submicrometer resolution with sensors [36]. A design for generating two vibration components of motion at around 40 kHz where a longitudinal and bending

mode occur at the same frequency is proposed by Markus Bauer [37]. This design schematic is shown in Figure 2-16.

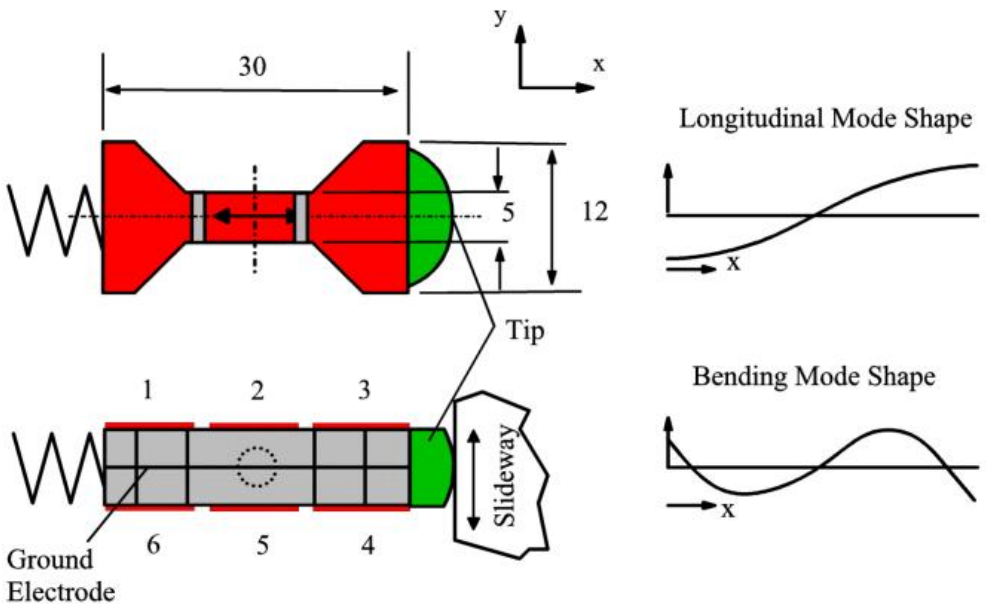


Figure 2-16. Combined-mode ultrasonic motor design (dimensions in mm) [37].

The geometry is sized such that both the longitudinal and mode occur at the same frequency, allowing simultaneous excitation with sine wave voltage signals. The body of the actuator is made from two piezoelectric elements, where the outwards facing electrodes are the voltage input and the center layer is electrical ground. The input electrodes are split up in to three sections each (labeled one through six in Figure 2-16) which allows for the independent excitation of the longitudinal and bending modes. For longitudinal excitation, pads two and five are excited in-phase. Bending motion is produced by exciting pads one and four simultaneously and 180° out-of-phase with pads three and six. Since the motor must provide friction force on a slideway to produce motion, the motor is mounted using flexures that prohibit rotation of the device while still allowing the desired longitudinal and bending motion.

2.3 PRELIMINARY DESIGNS FOR NANOCOINING

Based on previous work for ultrasonic EVAM devices that generate elliptical tool paths at more than 20 kHz, an elliptically-vibrating actuator design suitable for nanocoining was investigated by considering a resonating, beam-type actuator. Initial designs consisted of uniform aluminum beams with piezoelectric elements fixed to the sides to excite the desired vibration modes. Both cantilever and free-free beam orientations were explored with finite element models and prototyping to test feasibility for use in the nanocoin process. The goal for these preliminary designs is to achieve the desired 3 x 8 μm ellipse at ultrasonic frequencies (> 20 kHz).

2.3.1 Cantilever

Due to simple configuration and ease of mounting, a cantilever-type actuator was first considered to produce 2-D motion. FEA was used to size the beam such that a transverse and a longitudinal mode occurred at (or near) the same frequency to produce 2-D motion and the free end. The FEA modal analysis, shown in Figure 2-17, resulted in a bending mode located at 30,882 Hz and a longitudinal mode of 30,899 Hz.

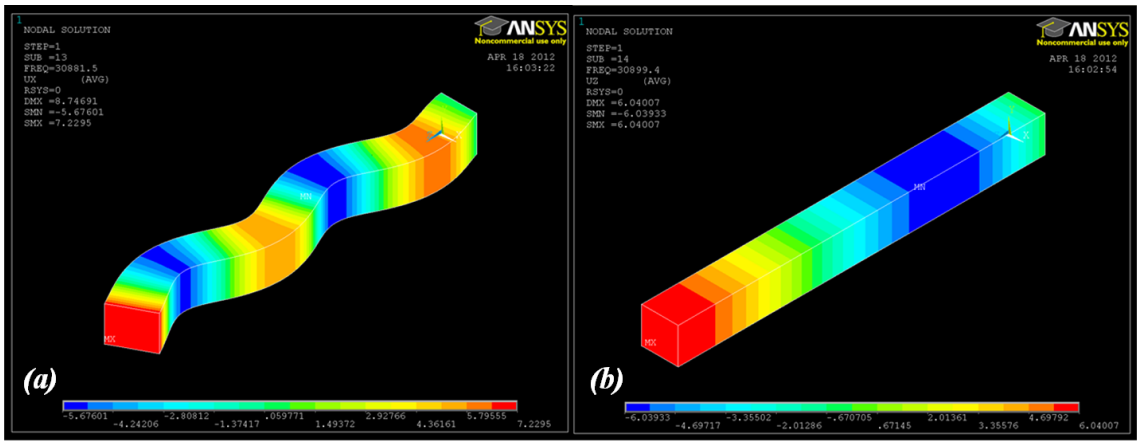


Figure 2-17. FEA analysis for cantilever beam vibrating in the transverse (a) and longitudinal modes (b).

The beam actuator was constructed using a single piece of 6061-T6 aluminum which measured 12.96 mm x 12.68 mm x 130.6 mm. Two pieces of piezoelectric (20 mm x 11 mm) were attached to the beam using CircuitWorks CW2400 conductive epoxy. Once the epoxy cured, wires were soldered onto the outer electrode of each piezo. To ground the actuator, a drilled and 10-32 tapped hole was used on the fixed end of the beam to attach a common ground wire. This construction of the cantilever actuator is shown in Figure 2-18.

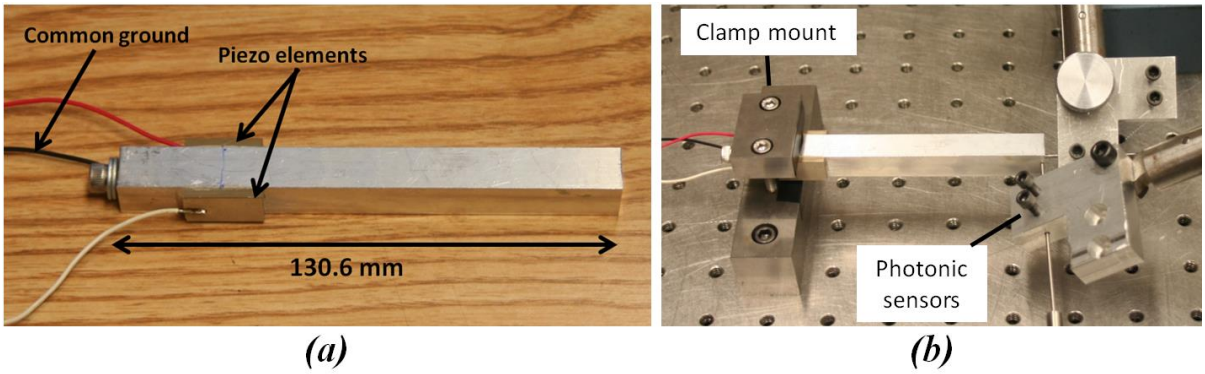


Figure 2-18. Components (a) and measurement method (b) of the cantilever beam.

The behavior of the actuator was measured by mounting the fixed end of the beam in a clamp-type mount. The tip motion was captured in the longitudinal and bending direction simultaneously using a two channel, MTI-2100 photonic sensor with a bandwidth of 150 kHz. The piezos were driven using two, 90° out-of-phase sine wave voltage signals via a two channel signal generator amplified by two TREK Model PZD350A M/S piezo drivers (one for each drive signal) for a total amplitude of 300V. The resonance of the actuator was determined by varying the frequency of the sine wave voltage source, and finding the largest amplitude response. The measured tip displacements in the longitudinal and bending directions are given in Figure 2-19.

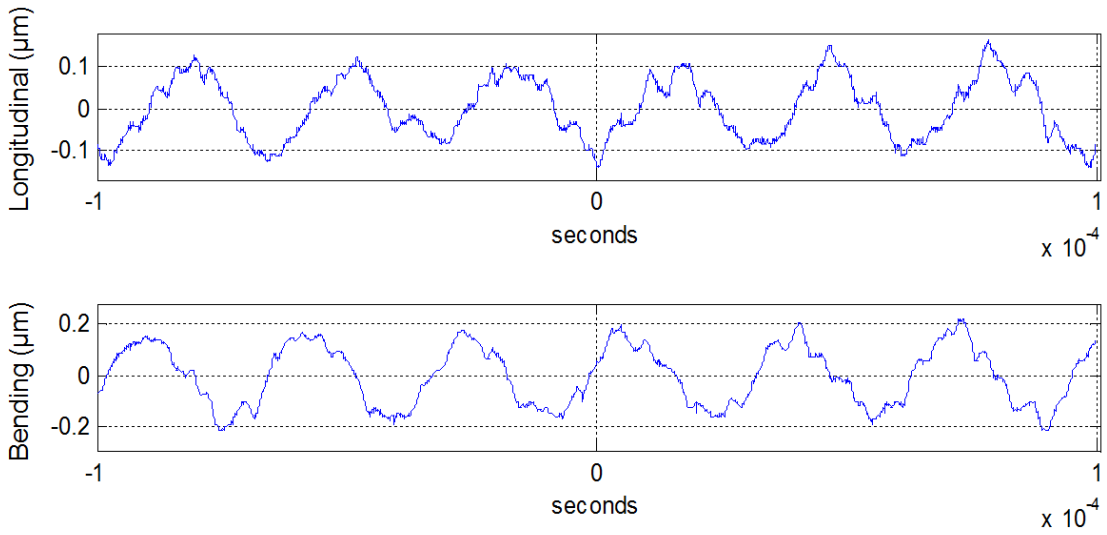


Figure 2-19. Longitudinal and bending amplitude for cantilever beam actuator at 31 kHz.

The results clearly show that the displacement of the cantilever-type actuator is much smaller than is required for the nanocoining process. In addition, the motion appears to be noisy, where higher order harmonics may be getting excited. This behavior is undesirable for the indenting operation.

2.3.2 Free-free beam

Since the cantilever-type actuator design yielded unpromising preliminary results including poor displacement characteristics and higher order frequency noise, a free-free beam design was tested to determine its feasibility for nanocoining. This design concept is common in ultrasonic devices for EVAM [29, 31].

To test the feasibility of the free-free concept, the behavior of a uniform beam with a rectangular cross-section was investigated. FEA analysis was used to determine a beam shape that would yield suitable mounting points, where two vibration nodes occurred at the same point along the beam in two vibration modes. This was done to minimize damping due to the mounts during these preliminary tests. The modal analysis performed is shown in Figure 2-20.

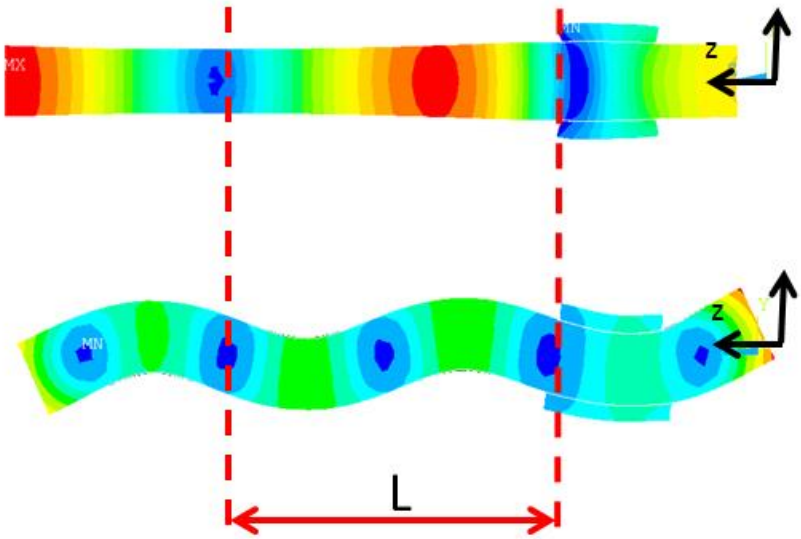


Figure 2-20. Modal analysis of longitudinal and bending mode.

The two operating modes for this prototype were determined to be the 2nd longitudinal mode and 4th bending mode, where the distance between the mutual node locations for the two modes is given as *L*. The resulting mode frequencies were determined to be 39,367 Hz and 30,559 Hz for the longitudinal and bending mode, respectively. Modifying the geometry such that the mode frequencies match would result in node positions that did not occur in the same place; that is, there are no suitable mounting points for both of the modes of vibration. Since the modes do not occur at the same frequency, elliptical motion will not be present, however; this preliminary design demonstrated desirable behavior that motivated future designs.

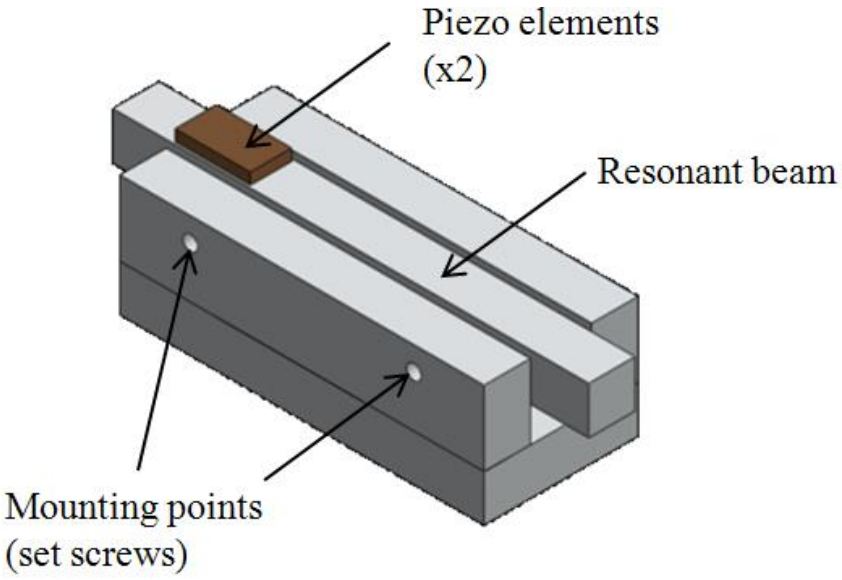


Figure 2-21. Free-free beam prototype assembly for testing.

Figure 2-21 illustrates the actuator assembly, where a 130.9 x 13.3 x 12.8 mm, 6061 aluminum beam is fastened by contacting set screws at the node locations along the length of the beam. Two 20.7 x 11.5 mm PKI802 piezo elements are fixed to opposing sides of the beam with conductive CircuitWorks CW2400 epoxy, such that the outer-facing surfaces of the piezos are excited with a voltage and the vibrating beam acts as electrical ground.

Experiments were run to measure the displacement at the tip and resonant frequency of both the bending and longitudinal modes (Figure 2-22). Sine wave signals from the signal generator are amplified via TREK piezo drivers (100:1V) and sent to the piezo elements. MTI Instruments dual-channel photonic sensors are used to capture dynamic displacement at the beam tip. The probes are positioned to measure both the bending and longitudinal deflections of the beam. The results for the output amplitude of this setup are shown in Figure 2-23.

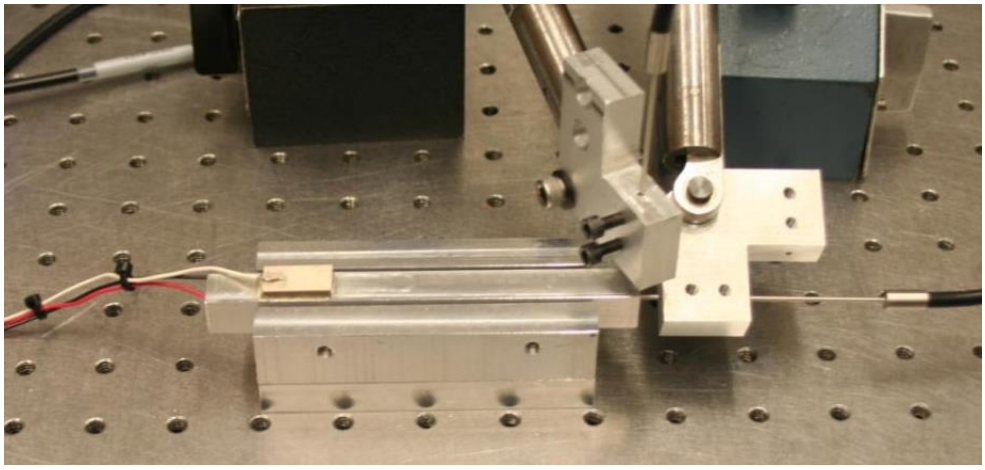


Figure 2-22. Experimental setup of beam actuator assembly with photonic sensors measuring the longitudinal and bending vibration directions.

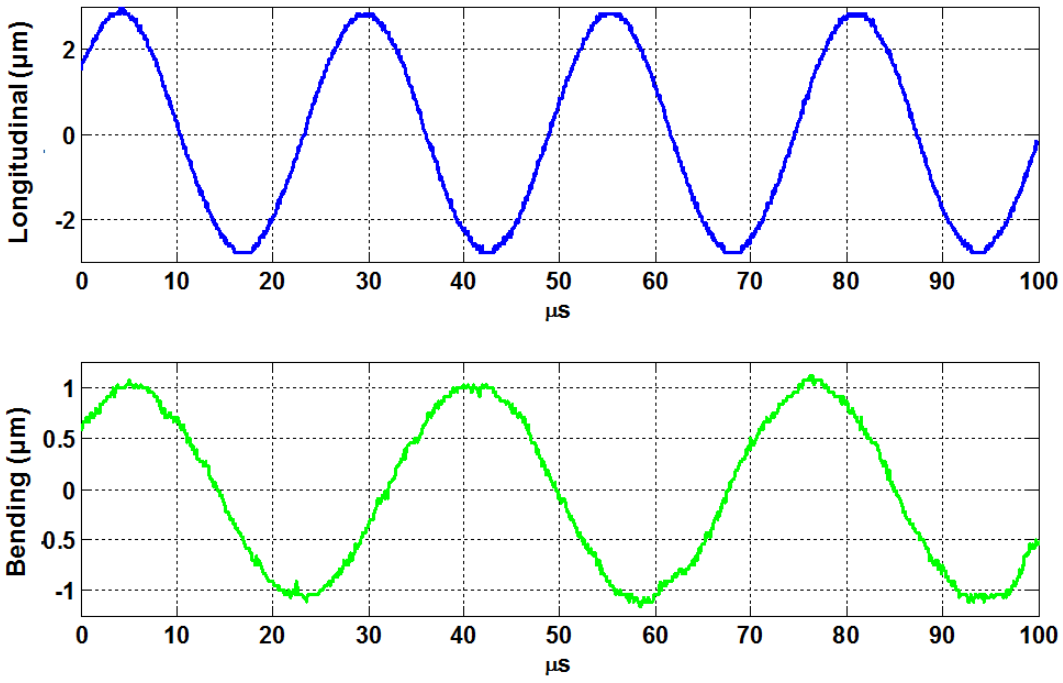


Figure 2-23. Longitudinal and bending amplitude for free-free beam prototype.

The resonant frequencies of the longitudinal and bending modes were measured to be at 39,010 Hz and 28,190 Hz, respectively. The amplitude of vibration was measured to be

about an order of magnitude larger than the cantilever arrangement, as well as considerably cleaner. This suggests that the higher frequency noise was successfully reduced, resulting in motion from only the two desired modes.

2.4 SUMMARY

Resonant devices have many applications, including ultrasonic welding, drilling and vibration assisted machining. The size and shape of these devices dictate the operating frequency, as well as the output amplitude. 1-D devices are used for linear actuation in processes such as ultrasonic drilling and welding. Multi-dimensional ultrasonic actuators have been developed primarily for EVAM processes, although benefits for using elliptical vibration welding have also been demonstrated. These actuators are sized such that two orthogonal vibration modes occur at the same frequency. For the application of nanocoining, the longitudinal-bending mode actuator seems most feasible for creating the required elliptical path.

Two nanocoin actuator prototypes were designed using FEA analysis and constructed to investigate the feasibility of potential designs. The first prototype incorporated a cantilever beam-type vibrator that was sized to vibrate in the longitudinal and bending mode at the same frequency. Two piezo elements were epoxied to opposing sides and excited 90° out-of-phase to produce 2-D motion. The results showed not only insufficient amplitude, but also high frequency noise on top of the excitation frequency.

The second prototype was based on a free-free uniform beam configuration that can vibrate in the longitudinal and bending direction. FEA was used to design a beam where the nodes from a longitudinal and bending mode occur in the same position to maximize amplitude. Since the prototype was constrained by this mounting scheme, the mode frequencies could not be matched to occur at exactly the same frequency using only a rectangular cross-section, uniform beam. The final measurements for this system showed the amplitude at the tip was

about an order of magnitude larger than the cantilever setup. The measurements also appeared to be less noisy with the free-free beam system.

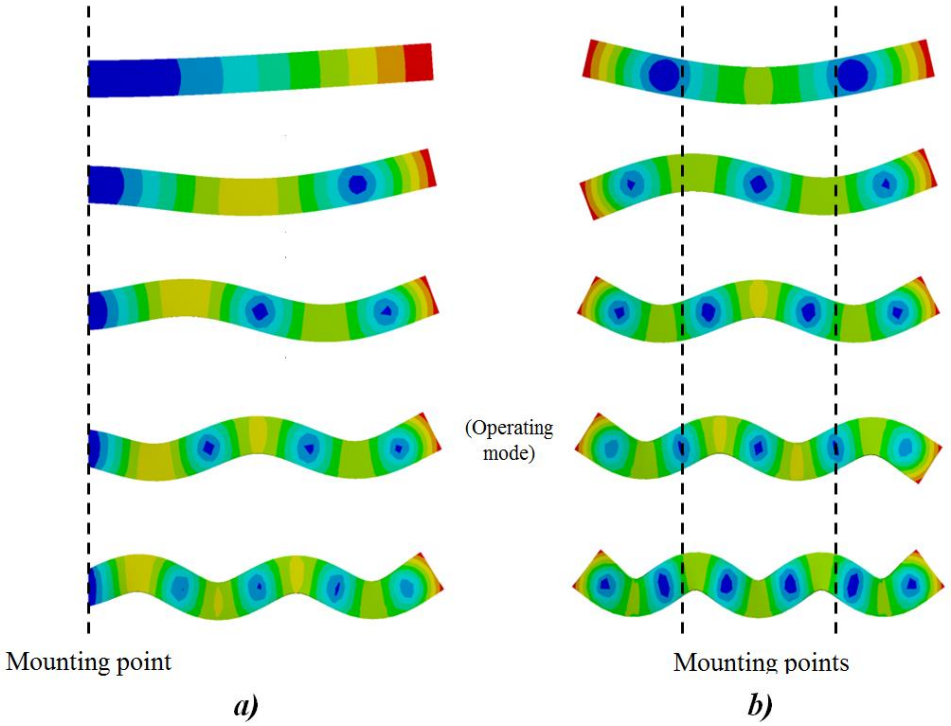


Figure 2-24. Mounting a cantilever beam (a) and a free-free beam (b).

Figure 2-24 illustrates why the free-free beam results may have performed better than the cantilever beam results. Considering the cantilever arrangement in Figure 2-24(a) where the mounting point is at the left edge of the beam, it is clear that all the transverse modes illustrated share that same node point. This means that there is no constraint that isolates the desired vibration mode, such that external forces could potentially excited higher and lower harmonics. In contrast to this, the free-free beam mounting points illustrated in Figure 2-24(b) are unique to only the 4^{th} vibration mode. This suggests that the rigid mounting structure prevents the undesired modes from vibrating due to disturbances and generate a higher quality tip vibration. Since the free-free beam prototype performed better in the initial tests, this design was pursued in the design of a nanocoining actuator.

3 ACTUATOR DESIGN THEORY

3.1 DESIGN

The elliptical indenting motion is generated using a 2D beam-type resonant actuator. The 2-D motion was achieved at resonance by designing the geometry of the actuator such that two orthogonally vibrating modes occur at the same frequency. Since the two modes occur at the same frequency, piezoelectric elements can be used to excite the beam at a single frequency while activating both vibration modes. This bimodal operation is illustrated in Figure 3-1 for a uniform beam.

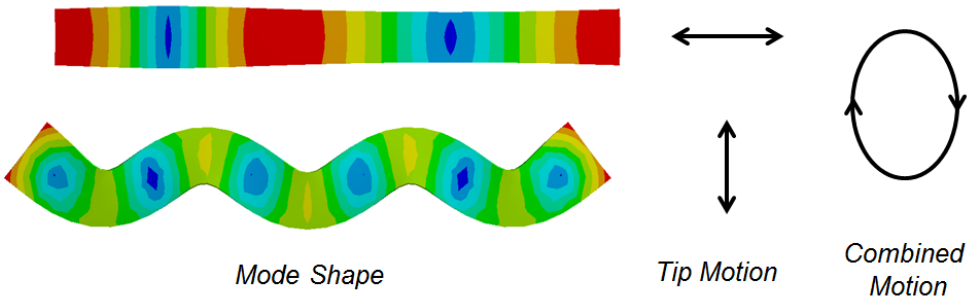


Figure 3-1. Finite element results for orthogonal beam modes and combined tip motion.

The design is based on uniform beam in a free-free orientation, where the longitudinal natural frequency $\omega_{n,L}$ and bending natural frequency $\omega_{n,B}$ can be expressed analytically with the following equations [38]:

$$\omega_{n,L} = \frac{n\pi c}{l}$$
$$\omega_{n,B} = (\beta l)^2 \sqrt{\frac{EI}{\rho A l^4}} \tag{3-1}$$

where

n = mode number

c = wave speed of the beam material

E = Young's Modulus

I = area moment of inertia

ρ = material density

l = length of the beam

A = cross-sectional area of the beam

βl = function of the mode number and boundary conditions

The actuator dimensions were chosen to match the longitudinal and bending modes of interest. Considering a uniform aluminum beam with a width of 12.7 mm and a height of 10 mm, the 2nd longitudinal and the 5th bending mode frequencies are plotted as a function of beam length in Figure 3-2. This graph shows there is a point at 40 kHz where the two modes vibrate the same frequency and can be simultaneously excited.

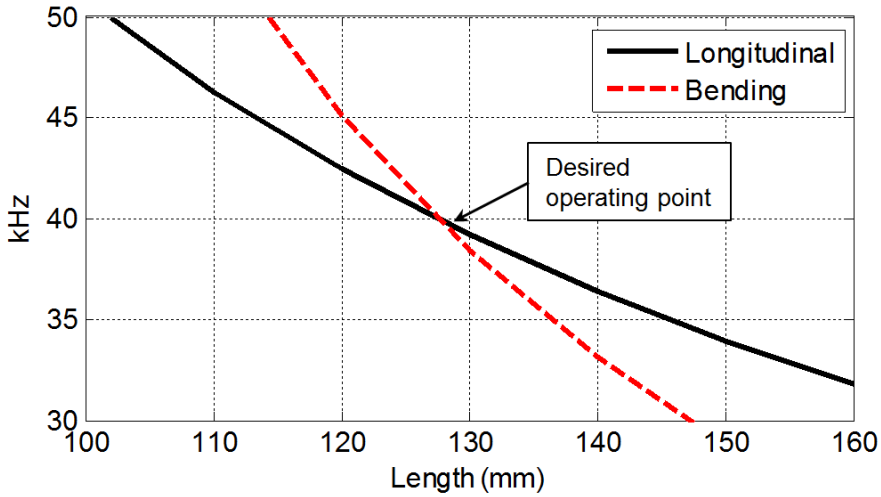


Figure 3-2. Theoretical longitudinal and transverse mode frequencies of a uniform aluminum beam of varying length.

The beam is harmonically excited using piezoelectric elements driven by sinusoidal voltages at the resonant frequency. For both modes to be excited simultaneously, two piezos are mounted on opposing side of the actuator. These piezos are excited with the identical

frequency and amplitude, but out-of-phase relative to each other to excite both the longitudinal and bending modes. A schematic of a basic 2-D resonant actuator is shown in Figure 3-3(b), where the piezo inputs are signals A and B. Phase φ (Figure 3-3(a)) is set between 0° and 180°, where φ = 0° would excited primarily the longitudinal mode and φ = 180° will excited primarily the bending mode. The forthcoming sections will detail the piezo placement, mounting point placement, geometry optimization and tuning the structure.

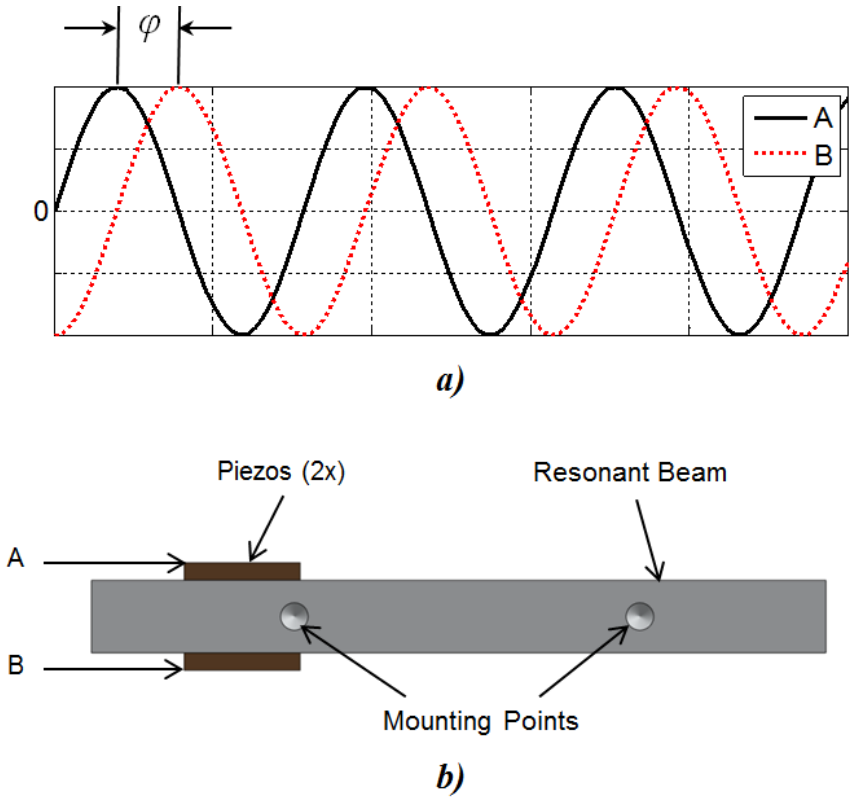


Figure 3-3. Input signals (a) and basic elliptically-vibrating actuator design (b).

3.1.1 Piezo Location

The piezoelectric elements required to excite the resonant beam modes are mounted directly on to opposite sides of the structure. However, since the piezo's displacement is limited to about 500 nm (by the size of the piezo and input voltage), the placement of these elements could greatly affect the dynamics of the resonating beam (where the goal is 3 and 9 μm of

amplitude in the bending and longitudinal direction, respectively). Due to the large difference in amplitudes, the piezos can potentially restrict the vibration of the beam and adversely affecting the performance. To minimize these effects, the piezo elements are mounted near a mutual vibration node (for both the longitudinal and bending mode) where the local amplitude is relatively small. Figure 3-4 shows the positioning the piezo elements along the beam.

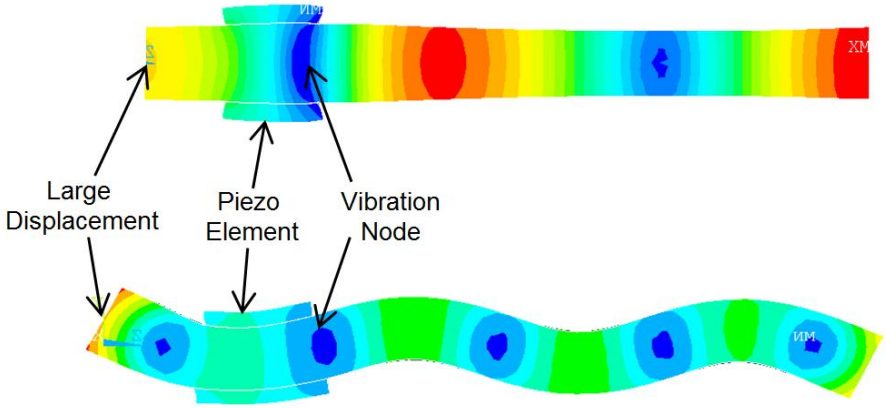


Figure 3-4. Illustration of piezo location relative to displacement node and actuator end.

3.1.2 Geometry

The overall dimensions of the actuator are determined using Equation (3-1) and are a function of the desired vibration modes and operating frequency. To increase the amplitude of the tool tip at resonance as compared to a uniform beam design, the geometry of the beam is reshaped to act as an ultrasonic concentrator. Ultrasonic concentrators reduce the cross-sectional area of the actuator and act as a strain amplifier. Under the assumption that momentum is conserved along the vibrating beam, a reduction in cross-sectional area means that less mass needs to move compared to a large cross-sectional area. The reduced mass (cross-section) means that the vibrational velocity must increase to conserve momentum, resulting in larger vibrational amplitude. This concept often utilized in high-powered ultrasonics as a means of increasing the amplitude of vibration at resonance [39-41]. Typical concentrators, or *horns*, are designed to amplify motion exclusively in the longitudinal

direction, however; in previous work they have also been shown to amplify combined elliptical motion [32]. Figure 3-5 shows an example of a linear ultrasonic actuator with an amplifying horn (a), and three examples of different horn geometries (b).

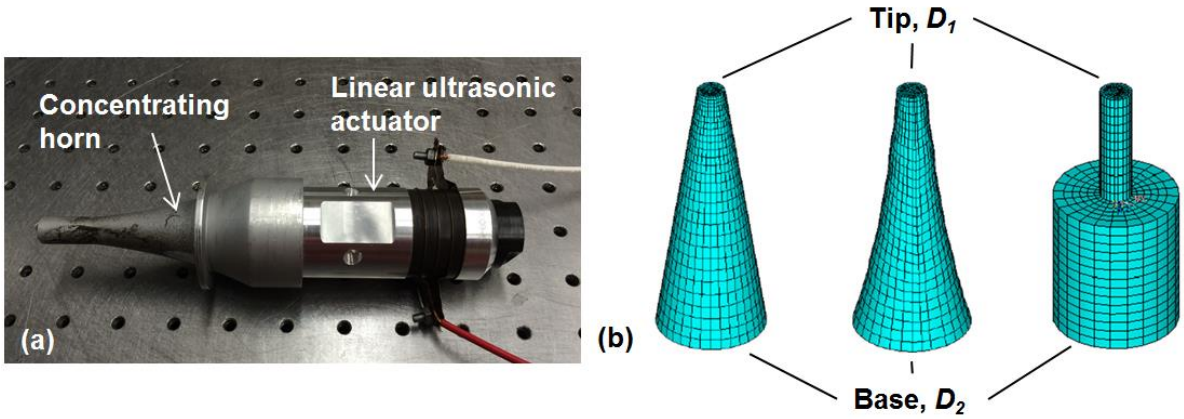


Figure 3-5. Linear resonant actuator with concentrating horn (a) and examples of concentrating horn designs (b) [40].

The strain magnification of a horn is a function of the base and tip diameters and the shape of the transition between the base and tip. The more severe the transition between diameters D_1 and D_2 in Figure 3-5 (where the cone would be least severe and the step would be most severe), the greater the amplification that occurs. For example, in Figure 3-5(b) the conical shaped horn produces the least amplification while the stepped design produces the most, relative to the other geometries [40]. However, as a result of the large amplitude increase, large stress concentrations are located around the step-down in the horn. These severe stresses are often avoided by choosing the conical or exponential profile in high-impact applications such as ultrasonic drilling where the large stress concentrations may result in failure [40]. For this application, the stepped horn design was chosen since it theoretically offers the largest amplitude gain and is also much easier to machine than the more complicated horn geometries. The stepped-horn amplitude gain can be approximated using the following relationship [41]:

$$Gain = \left(\frac{D_2}{D_1} \right)^2 \tag{3-2}$$

Since the other geometries are more complicated, there is no closed-form approximation of the amplification gain; therefore, more advanced modeling techniques must be used to estimate the gain.

For simplicity, the stepped horn design was utilized to maximize vibrational amplitude in the longitudinal and transverse directions. The cross-sectional area was reduced at the tool tip as shown in Figure 3-6, where $h_1 > h_2 > h_3$ and $w_1 > w_2$.

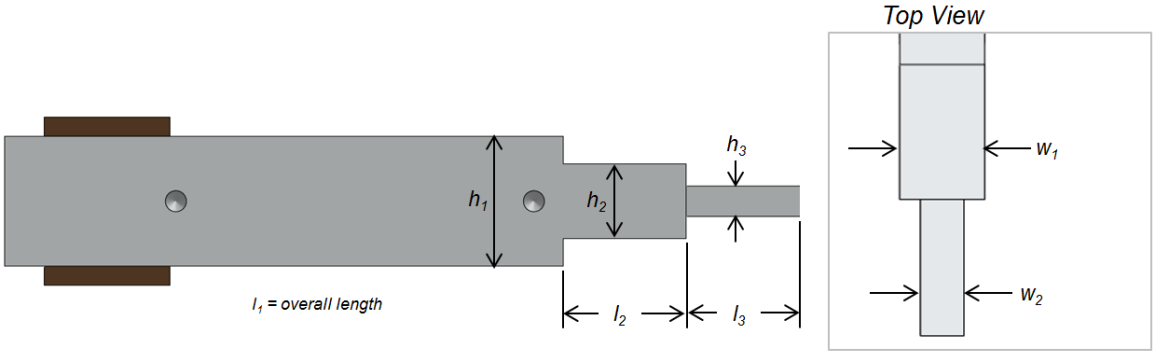


Figure 3-6. Beam geometry with concentrating horn design.

This geometry selection is also influenced by factors such as 1) the mounting points, and 2) ensuring that the two desired vibration modes remain at the same frequency. An iterative design method is used to optimize for these factors using finite element modeling. Further details regarding mounting point selection and piezo location are contained in the forthcoming sections.

3.1.3 Mounting Points

The actuator must be mounted in a manner that is axially stiff to prevent deflections during the indentation process. Additionally, the locations of the mounting points must not interfere

with the resonant vibration of either the longitudinal or bending modes. Since the goal is to generate two directions of motion simultaneously, the selected mounting points must be located at vibration nodes that are mutual to the two desired mode shapes.

Figure 3-7 illustrates the node locations for a uniform beam.

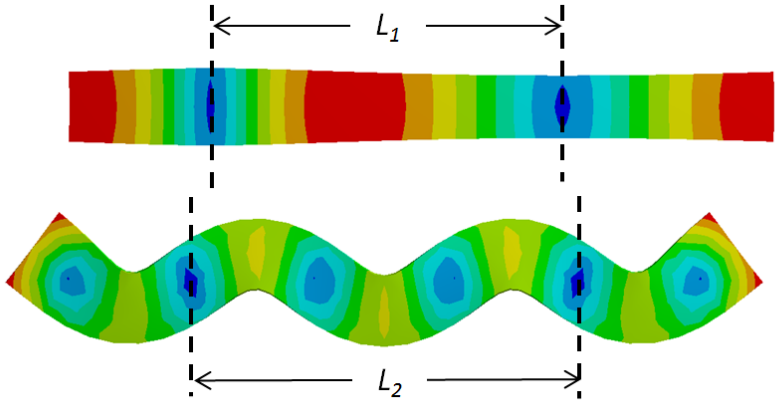


Figure 3-7. Displacement node locations for a uniform beam.

In Figure 3-7, the only two nodes for the longitudinal mode are separated by distance L_1, whereas the two 'matching' nodes in the bending modes are separated by distance L_2. Since L_1 does not equal L_2, if the two mounting points were to be located at the nodes for the longitudinal modes, then the bending motion would be restricted by the mount contacts. Likewise, if the nodes were chosen for the bending mode (separated by L_2) the longitudinal mode would be restricted. To address this issue, the parameters h_2, h_3, l_2, l_3, and w_2 shown in Figure 3-6 are chosen iteratively using finite element modeling to align the desired nodes for mounting. Modal analyses can be performed on the unconstrained model to measure where the theoretical nodes are located. By adjusting the tip dimensions, two node locations on the bending and longitudinal modes can be made to converge such that they are suitable mounting points for both modes. An illustration of the final geometry with the nodes aligned is given in Figure 3-8.

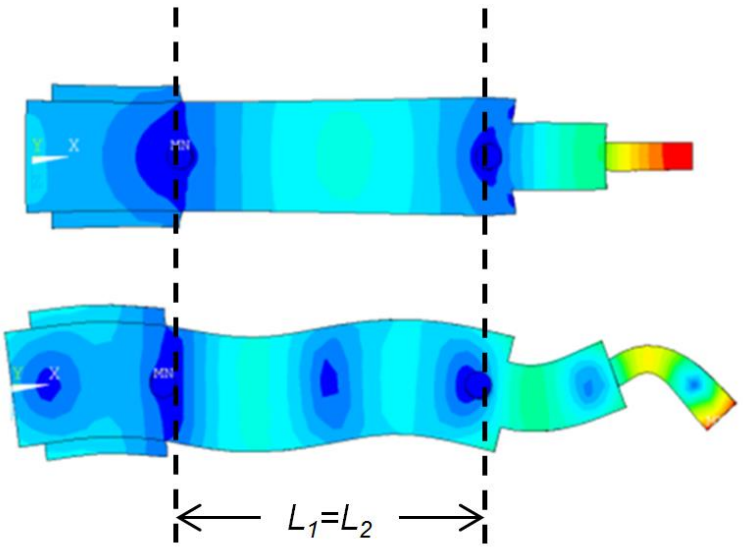


Figure 3-8. Actuator geometry modified for mutual mounting nodes.

3.1.3.1 Mounting Point Tolerances

Finite element simulations were used to investigate the sensitivity of the mount positions to the resonant frequency and amplitude of each operating mode by incrementally varying the mount position and simulating the actuator response. Shown in Figure 3-9, $L_{s,1}$ and $L_{s,2}$ are the distances from the back of the actuator to the two mounting points.

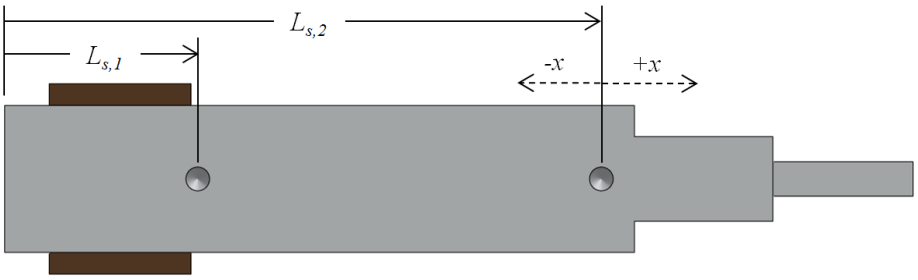


Figure 3-9. Actuator model for testing mount position sensitivity.

For this study, only the front node position ($L_{s,2}$) was varied where a positive increase in the x-direction means moving the mount closer to the tip of the actuator, while a negative

increase in the x-direction means moving the mount closer to the rear of the actuator. The simulations results are presented in Figure 3-10, where the x=0 position is the original mounting position for $L_{s,2}$. The change in the resonant frequencies and amplitude are given as a deviation from the original model results.

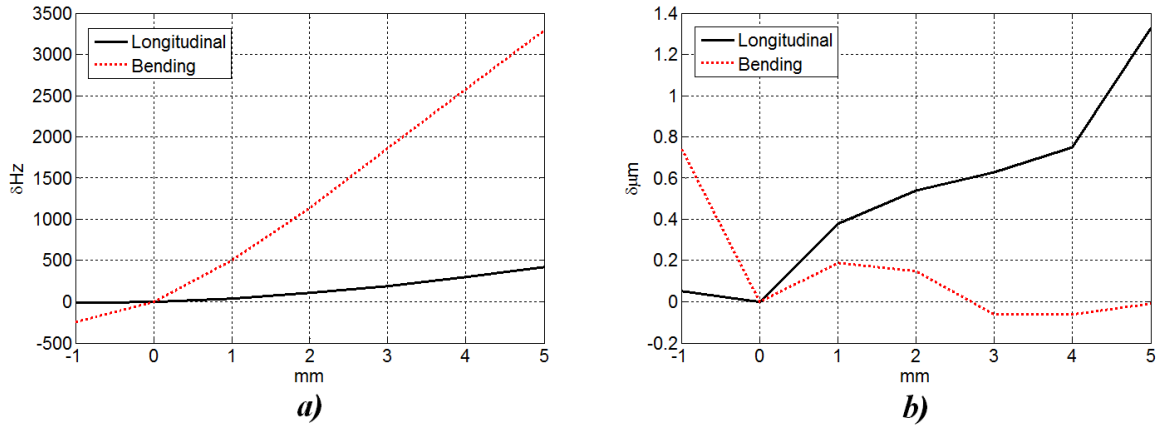


Figure 3-10. Simulation results of resonant frequency (a) and pk-pk amplitude (b) for varying the position of $L_{s,2}$.

The results in Figure 3-10(a) show the deviation of the longitudinal and bending frequency from the original model, where $\delta Hz = 0$ and $x = 0$. The results show that the bending mode is quite sensitive relative to the longitudinal mode to changes in the mount position. Increasing x by 5 mm results in shift in the resonant frequencies between the two modes of almost 3 kHz. In contrast, the amplitude response of the two modes to the mode change is relatively small where even a large change in x (up to 5 mm) results in an increase in amplitude of 1.3 μm. The bending mode shows even less sensitivity to changes in amplitude where increasing $L_{s,2}$ 5 μm shows little change.

These simulations demonstrate that the mount positions must be positioned correctly during the assembly of the actuator for the desired resonant behavior to occur. While the amplitude response shows little consequences for an error in the position, the mode frequencies appear to vary significantly with mount position. Based on the simulations, the mount locations

must be within 500 μm to keep the longitudinal and bending modes within 100 Hz of each other.

3.1.4 Design Summary

The goal of the actuator design process is to match the frequency of the longitudinal and bending mode for elliptical motion while maximizing the vibrational amplitude. The design steps for a 2D resonant actuator are as follows:

1. Calculate beam size: Use analytical solutions for uniform beam to estimate required dimensions based on desired operating frequency.
2. Select vibration modes: Use FEA to model the beam and ensure that the two desired operating modes share at least two displacement nodes for mounting. Resize using Step 1 if necessary.
3. Add piezo elements: Model piezoelectric elements to excite the beam in the desired directions. Since the deflection of the piezo is small compared to the beam (500 nm vs. 5 μm), the piezos can restrict the vibration of the resonating beam. The piezos are mounted close to the vibration nodes where the beam deflection is smaller (compared to the tip) to avoid a reduction in amplitude caused by the piezos restricting the beam motion.
4. Reshape tool end: Reducing the cross-sectional area near the tool end of the actuator increases the vibrational amplitude.
5. Adjust height/length: Since material was added or removed (piezo and reshaping), the overall height and length of the actuator must be adjusted to maintain the desired resonant frequency. The sensitivities of the two modes to dimension changes can be

deduced from Equation (3-1) where the longitudinal mode is represented as a function length, whereas the transverse mode is a function of length and cross-sectional area.

6. Verify node locations: Ensure the selected nodes for mounting are retained for both modes. Geometry modification/piezo placement can be used to realign nodes.

3.2 CONSTRUCTION

The actuator is constructed from a single piece of CNC machined 6061 aluminum and two Piezo Kinetics PKI802 piezoelectric elements. The piezos are fixed to the body of the actuator using electrically conductive epoxy, such that the exposed electrode of the each piezo is the signal voltage input and the epoxy side of the piezo and aluminum actuator body is electrical ground. A mounting jig is used to ensure the proper spacing between the back of the beam and the piezo elements during the epoxy process (Figure 3-11(a)).

3.2.1 Attaching the Piezos

The full assembly process of attaching the piezo elements to the actuator is as follows:

1. CNC machine the actuator body to spec and cut the appropriate size piezo elements.
2. Use fine grit sand paper (>100) or a pencil eraser to remove the outer layer of the electrode on one side of each piezo element, exposing the bare metal. Clean electrode and actuator body with acetone.
3. Mix CircuitWorks CW2400 conductive epoxy and coat the bare metal electro on each piezo element.
4. Position the piezo elements on the actuator body using the jig to space the piezos from the end of the actuator. The actuator body is clamped to a workbench and a C-clamp is used to keep pressure on the piezo elements during curing (Figure 3-11(b)).
5. Let sit 24 hours at room temperature. No high temperature curing was used.
6. After the epoxy is dried, use a razor blade and/or sand paper to remove the epoxy that is pushed out of the piezo-actuator body connection due to clamping down on the

piezos. *Ensure that none of the conductive epoxy dried on around the edges and shorts the electrodes on each piezo element.*

7. Remove a small portion of coating on the outward-facing conductor on both piezos to attach a wire using a pencil eraser.
8. Solder a wire to the outer electrode of each piezo element. Use flux and low-mid heat on the soldering iron to avoid damaging the electrode.

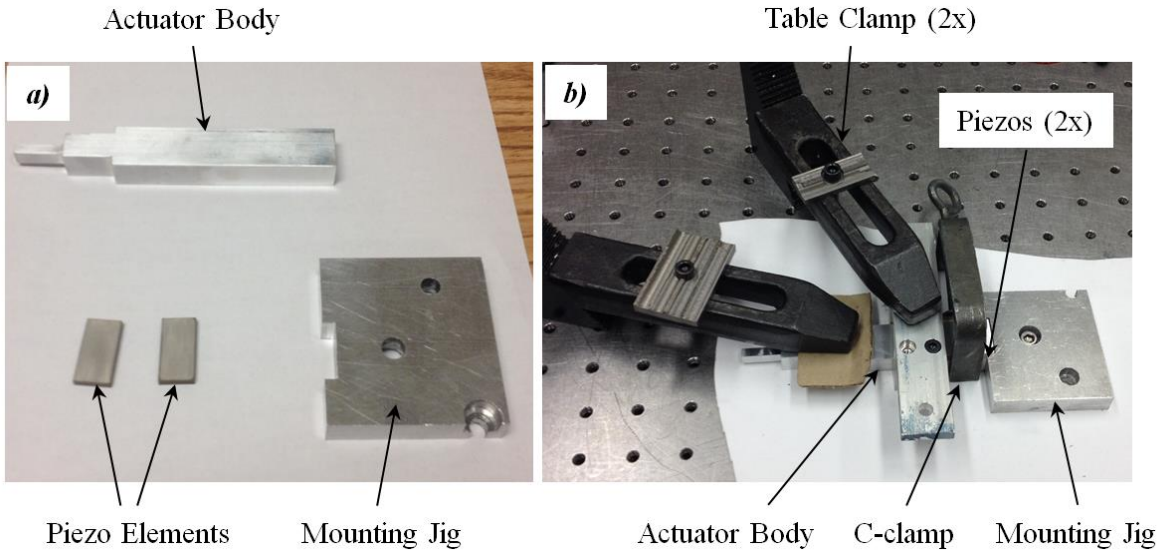


Figure 3-11. Actuator body, piezo elements and mounting jig (a) and setup to clamp piezos in place while epoxy cures (b).

3.2.2 Attaching the Die

The prefabricated diamond die is aligned and fastened to the tip of the actuator. As illustrated in Figure 3-12, an optical microscope with target cross-hairs is used to align the length of the die to be orthogonal with the front face of the actuator. A UV-curable adhesive was used to temporarily attach the diamond once it was aligned, after which 3M Scotch-Weld DP 460 structural adhesive was used.

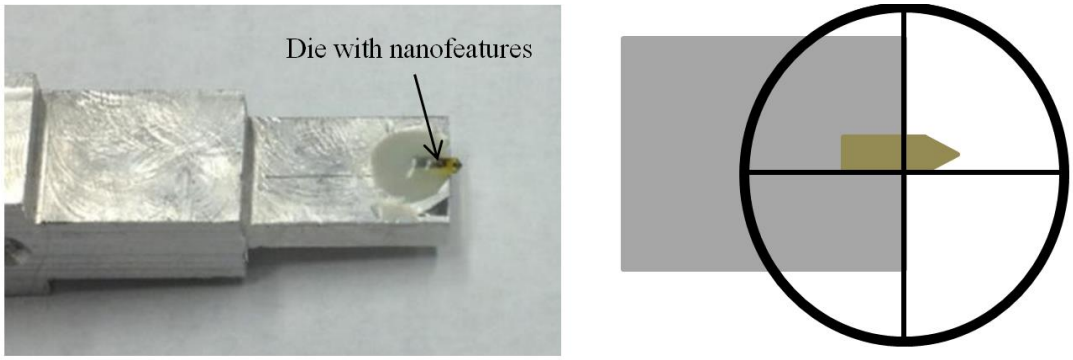


Figure 3-12. Fastening the die to the actuator.

1. A small amount of UV-curable Norland Optical Adhesive 60 is placed on the tip of the actuator where the die is to be mounted. The die is placed on top of the optical adhesive, ensuring that no adhesive adheres to the top of the die which could compromise the holding strength of the structural adhesive that is used for mounting.
2. An optical microscope with a cross-hair reticle is used to align the body of the die to be orthogonal with the front face of the actuator body.
3. Once positioned, a UV lamp is used to cure the optical adhesive by exposing the die area to the UV light for approximately 2 minutes.
4. Once the position is secured, 3M Scotch-Weld DP 460 structural adhesive can be used to fasten the die in place. Place a small amount over the die and allow it to flow around the three sides (preventing any epoxy from covering the nanofeatures at the tip).
5. Leave the epoxy to cure at room temperature for 24 hours.

3.2.3 Assembly

Four oval tip set screws are threaded though the sidewall and hold the actuator at the displacement node positions. The set screws are aligned with the conical-shaped notch at the node positions along the length of the actuator. The 10-32 set screws are chosen such that the oval tips make uniform contact with the node notches shown in Figure 3-13.

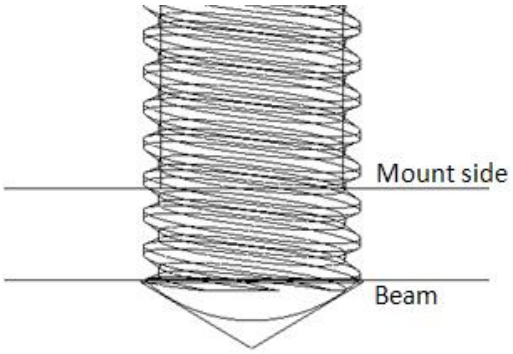


Figure 3-13. Oval point set screw contacting beam.

Clearance between the sidewalls and actuator is kept to minimum (less than 1 mm) to maximize longitudinal stiffness of the mount. The set screws were tightened incrementally to achieve a 3 Nm torque specification. An exploded view of the actuator assembly is shown in Figure 3-14.

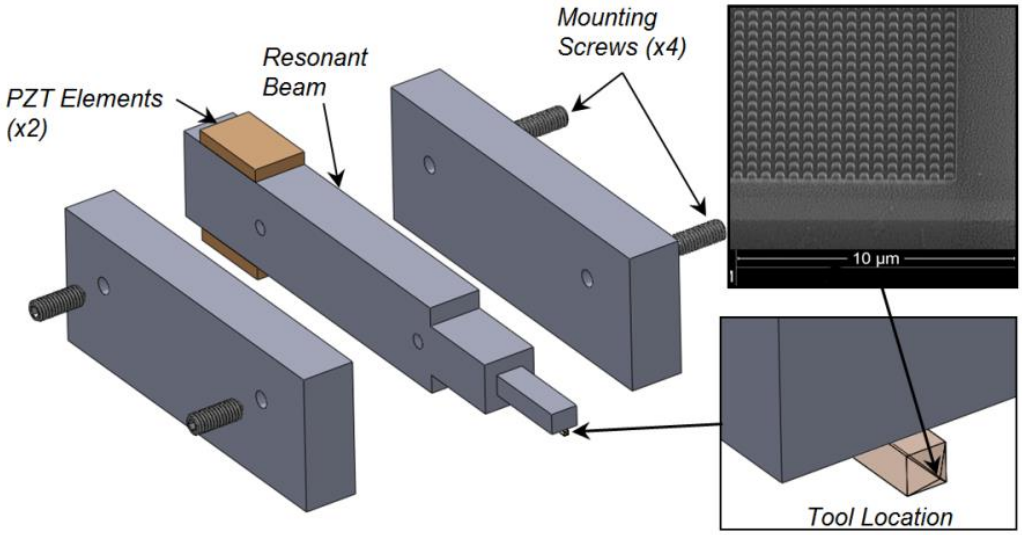


Figure 3-14. Assembly of actuator with die mounted at tip.

3.3 TUNING

While the goal is to match to orthogonal vibration modes, uncertainties such as dimensional errors in the construction process, behavior of the epoxy used to fix the piezo elements to the

beam, node locations, and mount contact points affect the resonant frequencies and are not easily captured by the finite element model. Several methods have been developed for tuning the resonant frequencies such that each of the active vibration modes can be adjusted to occur at the same frequency despite model/fabrication discrepancies. When a difference in the two desired resonances is measured, a systematic method of moving the mode frequencies relative to each other is required. Methods such as selectively modifying the geometry of the actuator after assembly and strategically adding mass to the system can be used to tune the resonant frequencies. However, this can also change the location of the nodes which will result in a decrease in amplitude.

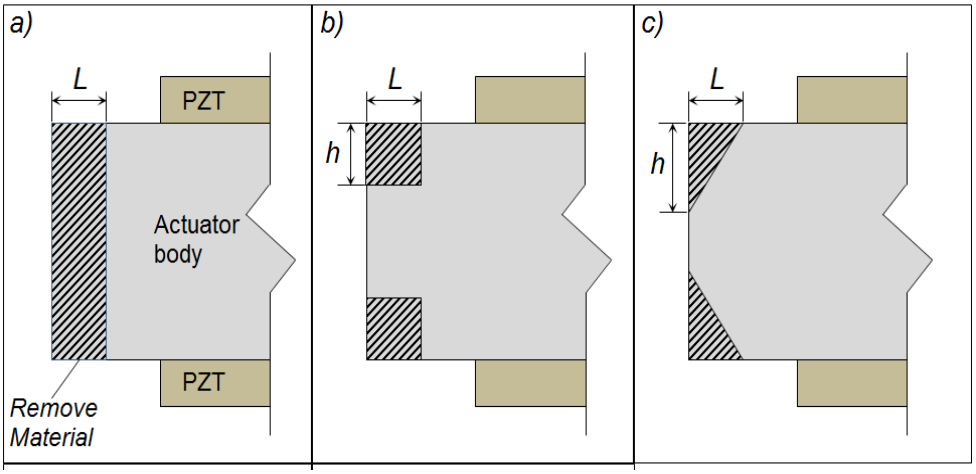


Figure 3-15. Methods of tuning the resonant frequencies of the actuator by removing material.

Figure 3-15 illustrates the methods used to tune the resonant response of the system where (a), (b), and (c) involve removing material to change the effective length and mass distribution of the structure. It can be deduced from the analytical solution of a uniform beam natural frequency presented in Equation (3-1) that changing the length of the beam will alter the natural frequency of both the longitudinal and bending mode. Additionally changing the mass distribution while retaining the effective overall length (as shown in Figure 3-15(b) and (c) should impact the bending mode differently than the longitudinal and

ultimately result in the mode frequencies changing relative to each other. Modal and harmonic analyses were performed using the Mechanical ANSYS FEA package to determine the effect each geometry change would have on the actuator.

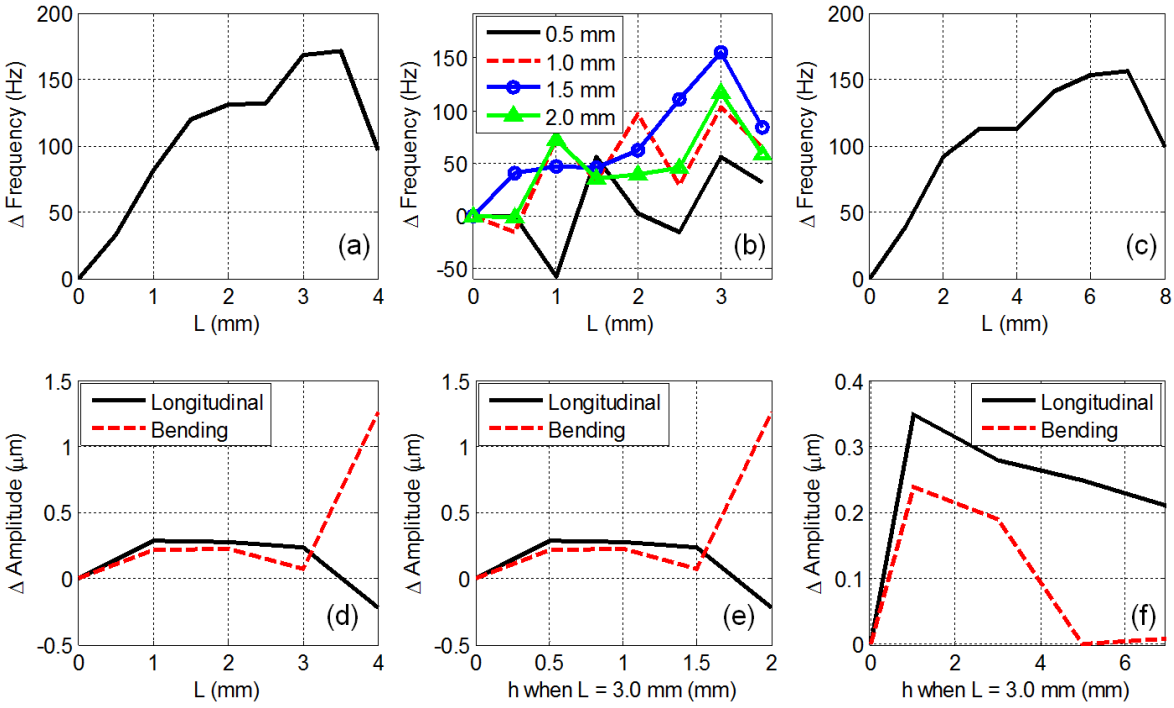


Figure 3-16. ANSYS results for frequency-tuning modifications where the bending mode is increased relative to the longitudinal mode.

Figure 3-16 illustrates these results for both changes in frequency and amplitude. The plots in Figure 3-16(a), (b) and (c) are the relative change in resonant frequency between the longitudinal and bending modes from the changes shown in Figure 3-15(a), (b) and (c), respectively, where ΔFrequency is the difference between the longitudinal and bending mode frequency. Figure 3-16(d), (e), and (f) are the change in output amplitude of the actuator for the longitudinal and bending mode. Modeling the amplitude response is important to ensure that the tuning modifications do not adversely interfere with the behavior of the actuator. It should be noted that these methods increase the bending mode relative the longitudinal mode,

so for instances where the initial response shows the longitudinal mode lower than the bending mode another strategy is needed.

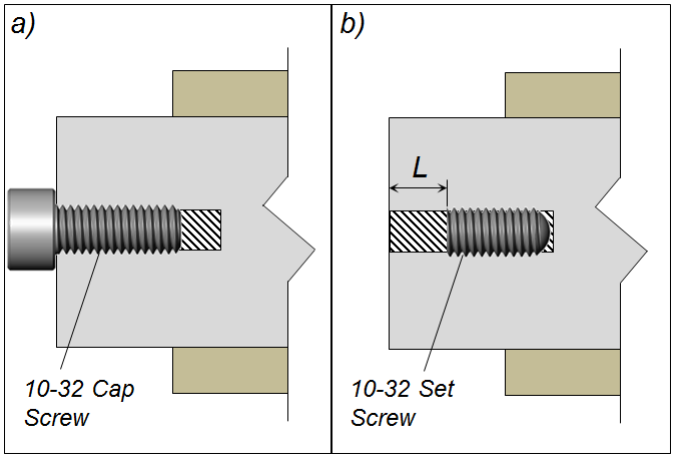


Figure 3-17. Tuning resonant frequency by adding mass.

To increase the longitudinal mode relative to the bending mode (the opposite gaol of the strategies illustrated in Figure 3-15), a method of adding mass to the structure was introduced. Figure 3-17 shows the technique using a steel cap screw (a) or set screw (b) threaded into the rear of the actuator. These various screw types vary the amount of additional mass added to the system, where the cap screw has a larger mass than the set screw. In addition, the dimension *L* in Figure 3-17(b) adjusts where the added mass is located which can be used to fine tune the mass distribution of the actuator. The addition of this steel mass increases the frequency of the bending mode and corrects for measured resonant frequency differences. The method illustrated in Figure 3-17(b) was employed to correct the measured mismatch in resonant frequencies. Figure 3-18(a) and (b) are the measured response of the actuator before and after tuning, respectively. It is apparent that prior to tuning, the longitudinal and bending resonant peaks were 300 Hz apart, whereas the tuned actuator shows the resonant peaks occurring at within 50 Hz of each other.

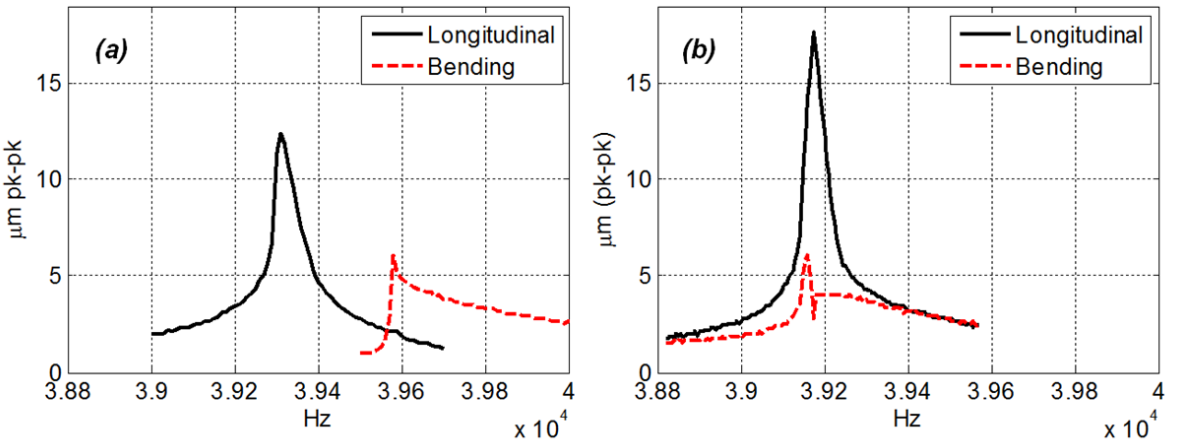


Figure 3-18. Frequency response of actuator before tuning (a) and after tuning (b).

3.4 SUMMARY

The design of a longitudinal-bending mode ultrasonic actuator for nanocoining is proposed based on previous research of combined-mode systems. The design of the geometry requires several considerations including frequency of operation, amplitude characteristics, mounting points, and piezo locations. The design process using FEA modeling is iterative where small changes in the geometry are made to achieve the necessary amplitude response and position the node locations for mounting the structure. A construction outline was also given where the final geometry was machined and assembled properly to achieve the desired performance.

4 50 KHZ DESIGN

The design process outlined in Section 3.1 is used to design an elliptically-vibrating actuator to perform indentations at 50 kHz.

4.1 DESIGN

The steps outlined in Section 3.1 were used to design a double-step down actuator to operate at 50 kHz as shown in Figure 3-6. FEA simulations were performed to create a geometry that would result in a bending and longitudinal mode at 50 KHz and two matching node locations to be used for mounting. The design process began with an idealized model of the beam which included only the beam and the two piezo elements (Figure 4-1(a)). The unconstrained model was necessary to determine the node positions of each vibration mode before the mount screws could be added to the model.

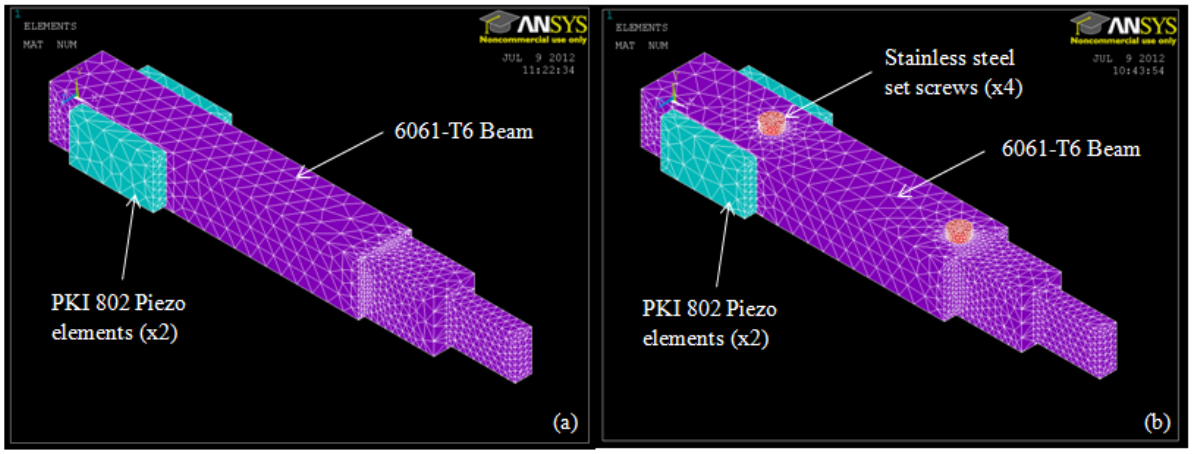


Figure 4-1. Free (a) and constrained (b) models.

Shown in Figure 4-1(b), the final design geometry is a double stepped shape that decreases in cross-sectional area towards the tip where the indenter will be located. This reduction in area not only aligns node positions between the desired vibration modes, but also serves as a horn to mechanically amplify the motion.

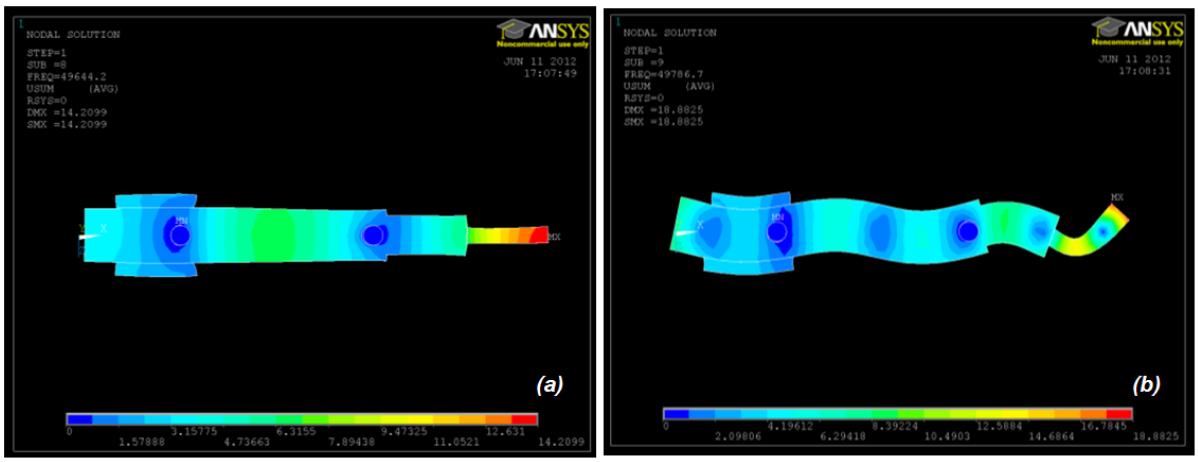


Figure 4-2. FEA simulation results for longitudinal (a) and bending (b) modes.

The final modal analyses of the longitudinal and bending modes are shown in Figure 4-2(a) and (b), respectively. The vibration modes were simulated to be within 200 Hz of each other, where the longitudinal mode was 49,644 Hz and the bending was 49,788 Hz. The final geometry is given in Figure 4-3.

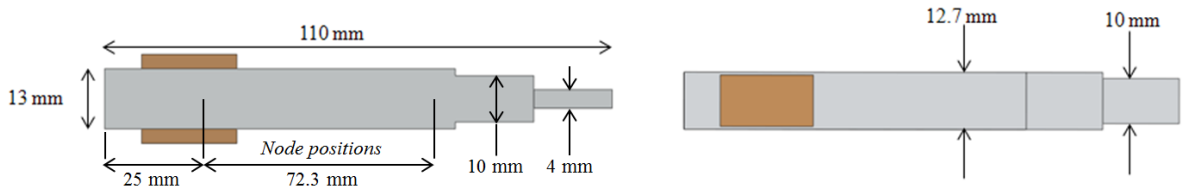


Figure 4-3. Dimensions of the 50 kHz actuator.

The beam shape was milled from a single piece of 6061-T6 aluminum and the piezo elements and lead wires were then applied as described in Section 3.2.1. MTI dual-channel photonic sensors were used to measure the dynamic displacement of the tip. Sine wave input signals are generated by a two channel signal generator and amplified by TREK piezo drivers. This measurement orientation of the sensors on the tip of the actuator is illustrated in Figure 4-4.

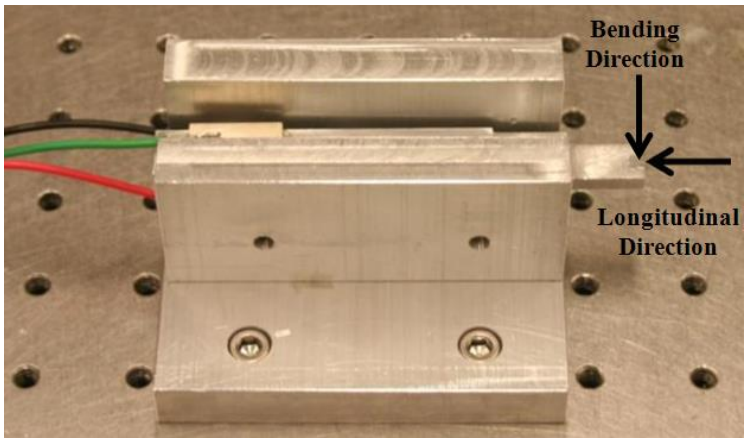


Figure 4-4. Experimental setup for stepped design.

To capture the longitudinal output at resonance, the two piezo elements were driven in-phase with amplified sine waves. The resonant frequency was measured by sweeping the input signal through a range of frequencies around 50 KHz to determine the largest amplitude. The result for an input voltage of 300 V at a frequency of 49.17 KHz is given in Figure 4-5.

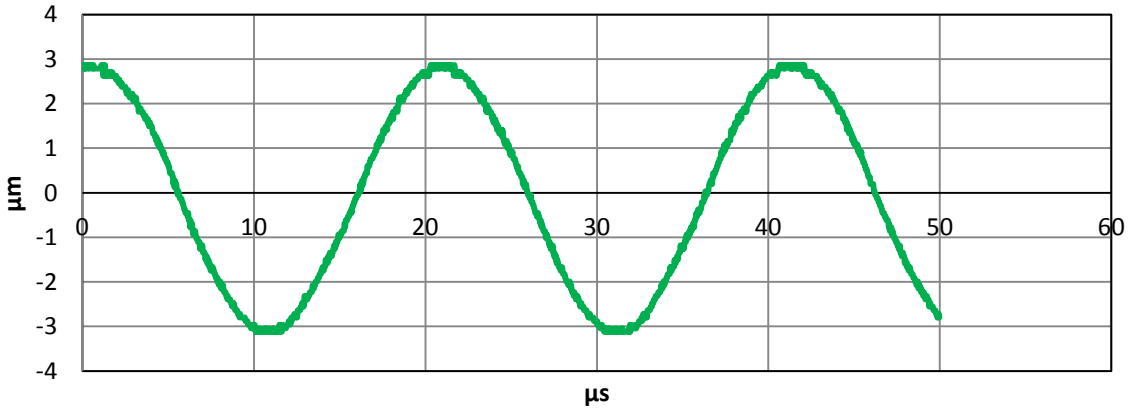


Figure 4-5. Longitudinal displacement measurement at 49.17 kHz and 300V.

The response in the bending direction was measured by setting the excitation signals to the piezo elements 180° out-of-phase, resulting in a purely bending motion. Using this excitation, the frequency of the input signals could be swept to determine the resonant

frequency of the bending mode. Once this process was performed, the bending mode was observed to occur at 45.34 KHz. The tip displacement response is plotted in Figure 4-6.

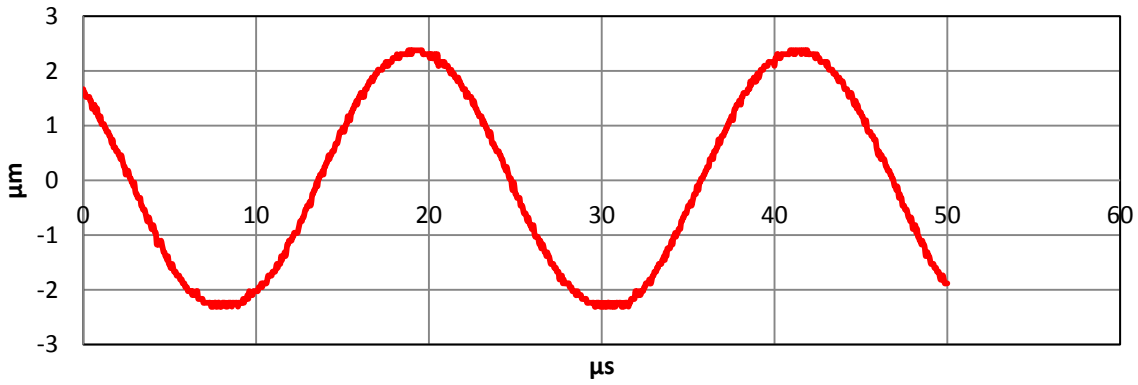


Figure 4-6. Bending displacement measurement at 45.34 kHz and 300V.

Based on the measured resonant frequencies of the bending and longitudinal modes, model discrepancies were apparent. The predicted longitudinal frequency was simulated to be 49,644 Hz whereas it was measured to be 49,170 Hz. The bending was predicted to be 49,788 Hz while the measurements yielded a result of 45,340 Hz. These results are summarized in Table 4-1.

Table 4-1. Comparison of measured and modeled resonant frequencies.

	Measured	Modeled
Longitudinal	49,170 Hz	49,644 Hz
Bending	45,340 Hz	49,788 Hz

The model error in predicting the mode frequencies could be attributed to the set screw contacts, which are modeled as fixed points. The epoxy that attaches the piezo elements to the beam is not included in the model and may influence the bending stiffness of the beam. Further actuator designs will compensate for this discrepancy by purposefully designing the bending mode to be larger than the longitudinal mode in the model.

4.1.1 Dimension Sensitivities

Accurately matching the resonant frequencies of longitudinal and bending modes requires a method of tuning that will allow adjustment of these frequencies in future design iterations. Initially, removing material from the front and back edges was considered since it can easily be done after the piezo elements are permanently attached. Simulations were performed to determine the effects of removing material from each side of the actuator. These variable dimensions are illustrated in Figure 4-7.

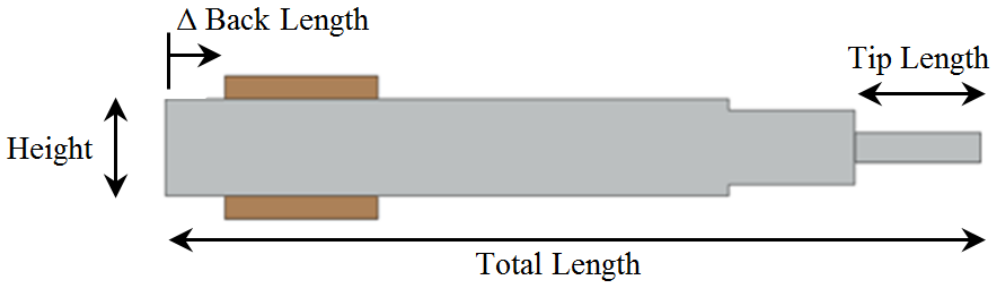


Figure 4-7. Variable dimensions for actuator.

Figure 4-8 show the results of changes in the back and tip length. The top plot shows that as the back length decreases (material is removed), both the longitudinal and bending frequencies increase but at different rates. This characteristic is favorable under conditions where the measured bending frequency is lower than the longitudinal. By removing material from the back, the bending natural frequency will increase at a quicker rate than the longitudinal to match the frequencies.

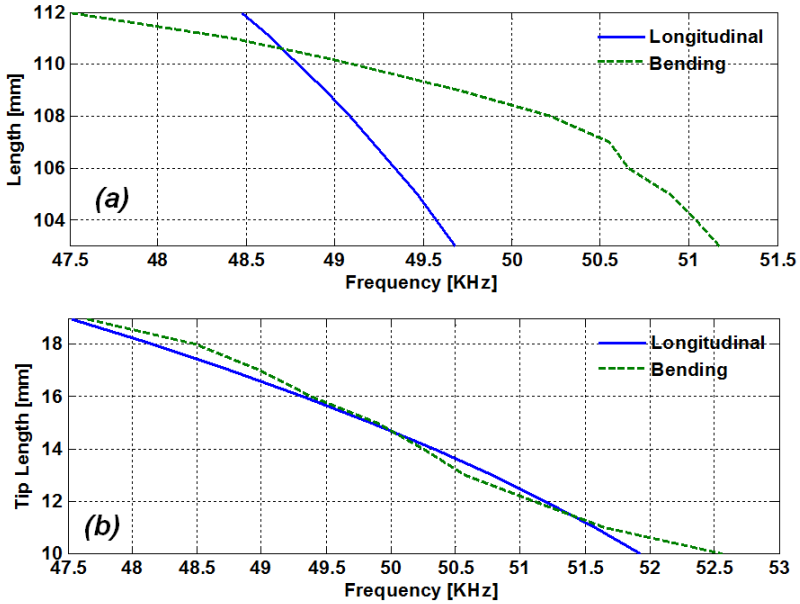


Figure 4-8. Change in resonant frequencies with total length (a) and tip length (b).

Figure 4-8(b) shows the results for a change in tip length. The simulations show an increase in resonant frequencies for both modes. The resonant frequencies increase at approximately the same rate, so little benefit would come from removing material at the tip as it would not cause the mode frequencies to change with respect to one another.

Another possibility to alter the response of the actuator is to increase the height of the beam (height dimension given in Figure 4-7). This modification, however, must be done prior to attaching the piezo elements. The results of changing the height dimension are given in Figure 4-9.

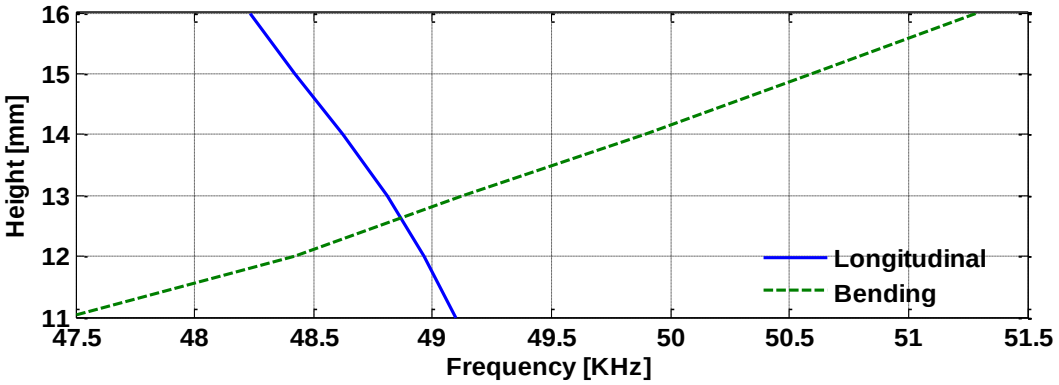


Figure 4-9. Resonant frequencies for change in height.

From these simulations it can be concluded that as the height of the actuator increases, the bending mode frequency increases, while the longitudinal decreases. The rate of change of the bending mode is significantly larger than the longitudinal over the range of heights simulated. Changing the height of the actuator could prove useful as design iterations occur because the bending mode can be adjusted with little effect on the longitudinal mode.

4.2 REVISED DESIGN

The second stepped design will seek to compensate the discrepancy in resonant frequencies by changing the geometry. Figure 4-9 shows the resulting change in frequencies for a change in height. According to the simulations, by increasing the height of the actuator from 13 mm to 16 mm, the bending should increase by 3 KHz. From the previous design results, this should compensate for the model error. The overall dimensions of the second stepped design are shown below in Figure 4-10.

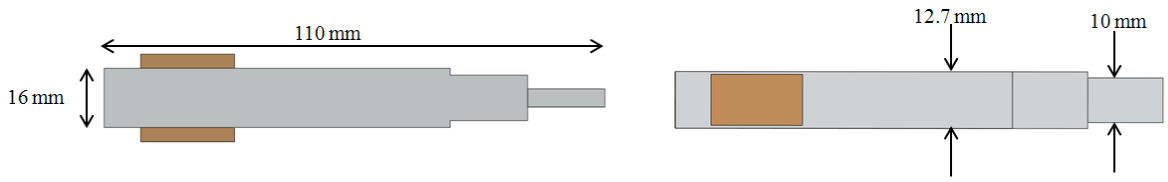


Figure 4-10. Stepped design dimensions to increase bending mode frequency.

Once this revision of the stepped design actuator was constructed, experiments were performed to measure the bending and longitudinal resonant frequencies. Longitudinal measurements were captured by driving the two PZT elements in-phase at with a 100V amplitude sine wave. The longitudinal resonant frequency was measured to be 48,530 Hz with 4 μm pk-pk displacement as shown in Figure 4-11 (a). The bending resonant frequency was measured to be 48,400 Hz with a 2 μm pk-pk amplitude (Figure 4-11(b)).

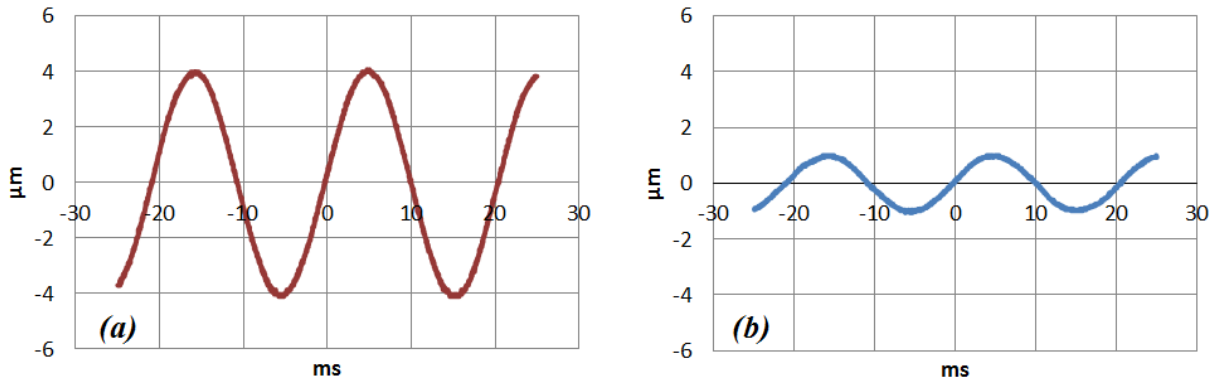


Figure 4-11. Longitudinal response at 48,530 Hz (a) and bending response at 48,400 Hz (b).

4.3 EXPERIMENTAL RESULTS

The revised 50 kHz design from Section 4.2 showed a measured longitudinal and bending mode at 48,530 Hz and 48,400 Hz, respectively. This 130 Hz difference was acceptable to generate ultrasonic, elliptical motion since the frequency can be set between the two resonances and still yield 2-D motion. Figure 4-12 shows two elliptical tip paths measured using MTI-2100 photonic sensors positioned to measure the longitudinal and bending motions simultaneously. The drive signals were two, out-of-phase 200V sine wave voltages.

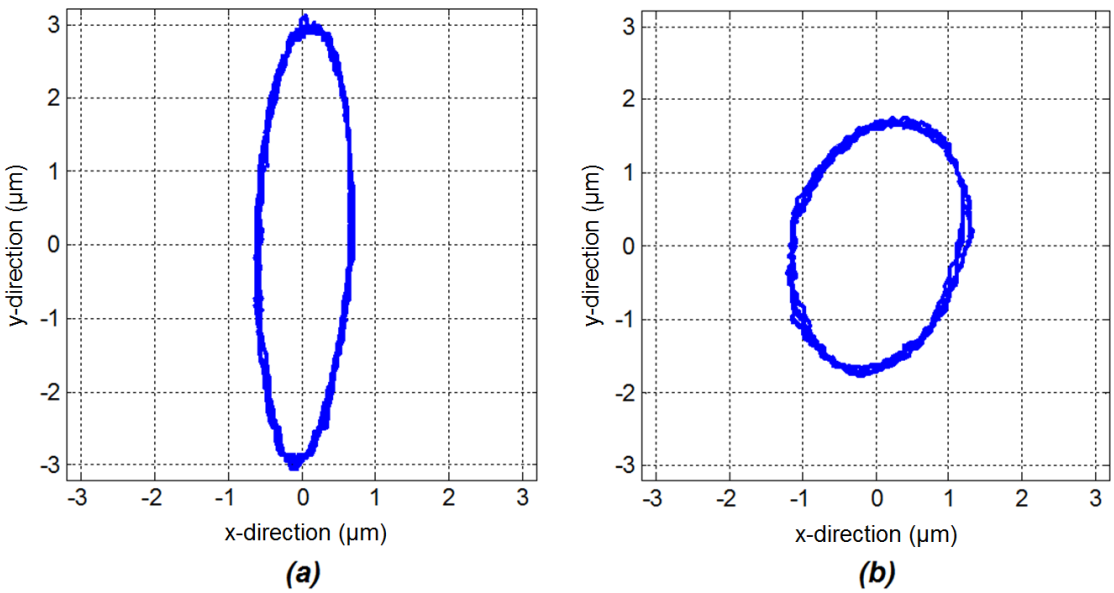


Figure 4-12. Elliptical path for 90° (a) and 180° (b) input phase, where the x-direction is bending motion and the y-direction is longitudinal motion.

For the case where the input signals are two 90° out-of-phase signals, the elliptical path produced by the actuator is shown in Figure 4-12(a). The amplitudes of the ellipse are 3 µm and 0.6 µm in the longitudinal and bending directions, respectively. While these amplitudes are significantly smaller than 3 µm bending and 8 µm longitudinal amplitude requirements, the aspect ratio of this ellipse is approximately the desired shape necessary for creating well-defined features. Figure 4-12(b) shows the tip displacement for 180° out-of-phase input signals, which emphasizes the bending motions more than the longitudinal motion. This tip path is a 1.2 µm amplitude in the x-direction (bending) and 1.8 µm amplitude in the y-direction (longitudinal) which is still less than the required displacement, however; would allow for a faster surface velocity while indenting compared to the ellipse in Figure 4-12(a). This is because the larger x-direction amplitude requires a faster workpiece speed to match the die velocity, resulting in a quicker indent time over a given area.

4.4 SUMMARY

Using the design methodology proposed in Chapter 3 a 50 kHz elliptically-vibrating actuator was designed using analytical and FEA techniques. The prototype was constructed and tested to evaluate its performance and capabilities to produce the required motion for nanocoining. The first prototype constructed showed a large difference between the resonant frequencies predicted by the ANSYS model and the measured response of the actuator. This difference in longitudinal and bending mode frequencies is problematic because it prevents both modes from being excited with a single frequency to produce elliptical motion. Due to this difference, the second prototype geometry was designed to compensate for the measured frequency difference of the first prototype. This resulted in measured longitudinal and bending modes approximately 130 Hz apart, which was sufficient to produce harmonic 2-D motion.

The experimental results demonstrated that this design was capable of producing elliptical paths at around 50 kHz; however, the amplitudes were insufficient for the nanocoining process based on the die size (Equation (1-4)). Future designs will seek to increase the amplitude of the actuator by adjusting the operating frequency and geometry.

5 40 KHZ DESIGN

5.1 DESIGN

Previous 50 kHz resonant actuator designs have successful generated an elliptical path; however, the amplitude of the ellipse is insufficient for the die size. The goal is to design an actuator capable of generating the required amplitudes (Section 1.5) to perform uniform indents with minimal overlap and deformation of the features. This is addressed by considering factors that play a large role in resonant actuator performance including frequency of operation and geometry.

5.1.1 Frequency of Operation

A general correlation between oscillation amplitude of a vibrating body and frequency at which it vibrates can be derived by considering the expression for acoustic power (P) [42].

$$P = u^2 \omega^2 ZA \tag{5-1}$$

where P is acoustic power, u is particle displacement relative to an at-rest state, ω is oscillation frequency, Z is acoustic impedance and A is the cross-sectional area of the body. Considering the acoustic power, acoustic impedance, and cross-sectional area to be constant, Equation (5-1) can be rearranged to result in the following form:

$$u = \frac{K}{\omega} \tag{5-2}$$

where K is a constant determined by power, impedance and cross-sectional area. This relationship supports the idea that fundamentally, oscillation amplitude is inversely proportional to oscillation frequency. This concept supports reducing the actuator's operational frequency in subsequent designs to achieve the necessary amplitude for indentation. Reducing the operating frequency to increase amplitude comes at a cost of

increasing the time to cover a given area with indents. The time it takes to cover a 1 m^2 area with indents using a 20x20 μm die as a function of frequency is illustrated in Figure 5-1. Decreasing the indenting frequency from 50 kHz to 40 kHz will result in an increased process time of approximately 4 hours (from 14 to 18), which is acceptable.

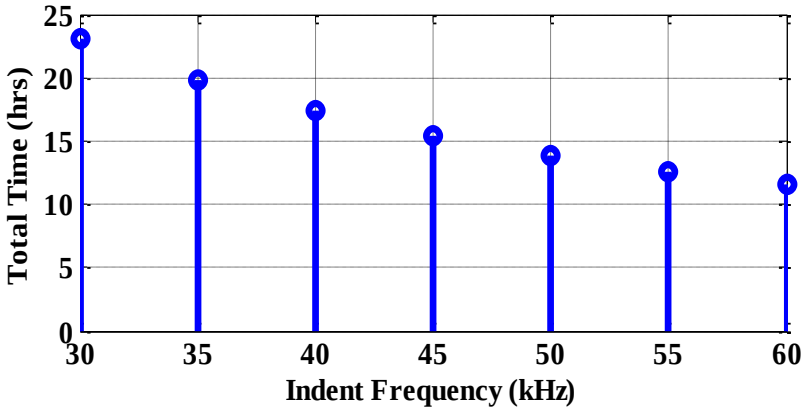


Figure 5-1. Time to indent a 1m^2 surface with a 20x20 μm die.

5.1.2 Overall Dimensions

The approximate beam size necessary to create a 40 kHz resonant device can be calculated using the longitudinal and bending vibration solutions to a free-free uniform beam. Revisiting the analytical expressions for longitudinal and bending natural frequencies given in Equation (3-1), a new beam size is calculated by first solving the longitudinal equation for length l such that $\omega_{n,L}$ equals 40 kHz. This length needed to achieve the desired longitudinal natural frequency can then be plugged into the expression for $\omega_{n,L}$ and a beam height, h, can be calculated assuming the desired 40 kHz natural frequency. The width of the beam transverse to the assumed bending motion is not included in the longitudinal solution and drops out of the bending solution so it is assumed to be on the same order as the height of the beam. The theoretical beam size for a 50 kHz and a 40 kHz resonant beam was calculated using this method. The results are given below in Table 5-1, assuming a uniform aluminum beam with free-free boundary conditions vibrating in its second longitudinal mode and fifth bending mode.

Table 5-1. Analytical solution for uniform beam size (units in mm).

Frequency (kHz)	Length, *l*	Height, *h*	Width, *w*
40	126.4	13.8	12.7
50	101.1	11.0	12.7
% Difference	***20.0***	***20.3***	

The results suggest that increasing the length and height of a 50 kHz resonating beam by 20% would decrease the natural frequency to the desired 40 kHz. This 20% scale factor was used to resize the existing 50 kHz actuator design to operate at the desired 40 kHz. A scale comparison of the 50 kHz actuator and the scaled 40 kHz design is given in Figure 5-2.

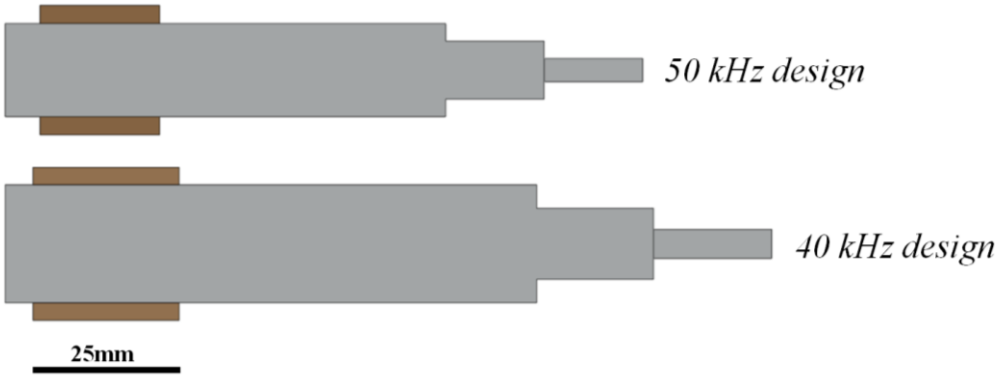


Figure 5-2. Previous 50 kHz design compared with the 40 kHz design using a 20% scale factor for all dimensions.

5.1.3 Tip Geometry

The stepped down horn design was previously implemented to produce strain magnification as the cross-sectional of the resonant beam decreases. Improving the amplitude response for the 40 kHz actuator required that the tip design be refined. ANSYS was used to perform modal analyses and determine the effects that the smaller tip had on the resonant frequencies of both vibration modes and node location. Reducing the tip width to 5 mm (from the previous 10 mm) maintained the resonant properties of the actuator, with respect to the mounting node positions and relative bending and longitudinal mode frequencies. An

illustration of the proposed modification is shown in Figure 5-3, where the cross-sectional area is reduced at the tip of the actuator.

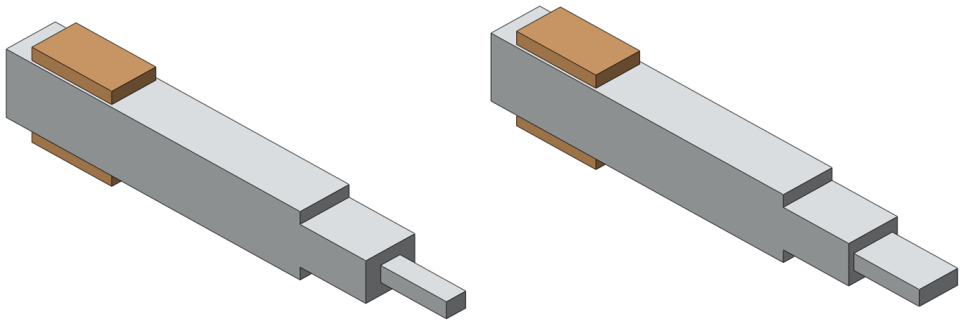


Figure 5-3. Modified tip geometry (right) and existing 50 kHz actuator design (left).

An existing 50 kHz actuator was modified to experimentally test the tip geometry and compare it with the previous design. The tip displacement was measured in the longitudinal and bending directions using the photonic sensors. The two piezo elements were driven with two 90° out-of-phase, 300V sine waves with the drive frequency sweeping from 44 to 50 kHz. The displacement data of both directions was collected as a function of frequency. Figure 5-4 shows the experimental setup used to test the displacement of the actuator in both the longitudinal and bending directions. A comparison of the modified tip to the original tip is also given.

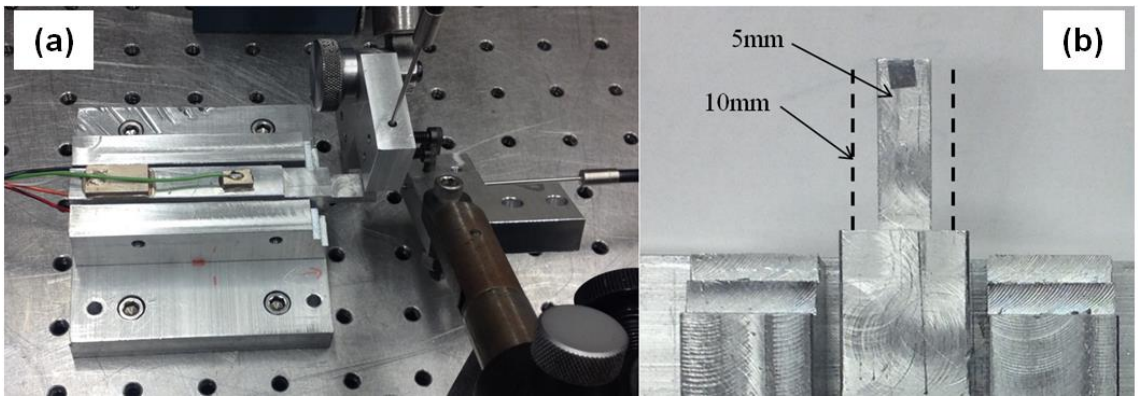


Figure 5-4. Experimental setup to test modified tip geometry (a) and tip comparison (b).

The measurements of the actuator are given in Figure 5-5 for both the original actuator and the modified actuator. Figure 5-5(a) shows the original 10 mm wide tip with the bending resonance occurring at 44.9 kHz and the longitudinal resonance occurring at 47.1 kHz, both with a peak amplitude of 2 μm (4 μm p-p). After the tip was modified, the results in Figure 5-5(b) show a measureable increase in displacement for both modes of vibration, while the resonant frequency of both modes also increased by approximately the same amount relative to each other. These results illustrate that decreasing the cross-sectional area at the tip will increase the amplitude in both directions with a negligible change in the longitudinal and bending resonant frequencies.

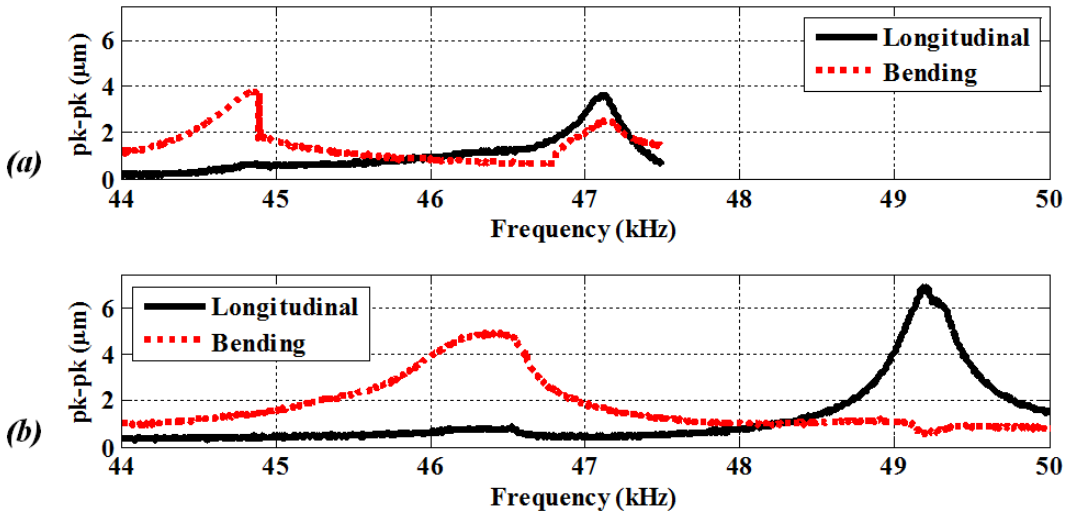


Figure 5-5. Amplitude response of 10 mm wide tip (a) and 5 mm wide tip (b).

5.1.4 FEA Simulations

A 40 kHz resonant actuator was designed to achieve larger amplitudes than previous versions based on the results presented in Section 5.1.1 and Section 5.1.3. After scaling the existing actuator design to resonate at 40 kHz, ANSYS was used to perform modal and harmonic analyses to fine-tune the geometry and piezo positions to maintain node locations and maximize performance characteristics. Results of these simulations are given in Figure 5-6.

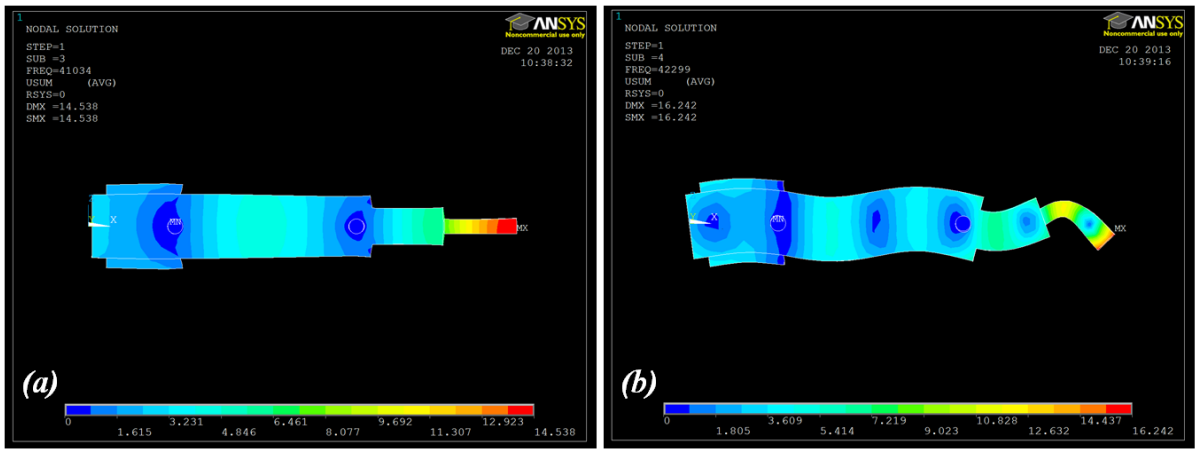


Figure 5-6. Modal analysis of 40 kHz actuator for longitudinal (a) and bending (b) modes.

Previous revisions of the actuator showed that the predicted bending mode in ANSYS is higher than the measured bending mode when constructed. Since this observation has been consistently measured, the actuator geometry is designed such that the bending mode predicted in ANSYS is higher than the longitudinal. This compensates for the mismatch between the modeled and the measured bending frequency and results in longitudinal and bending modes that occur at approximately the same frequency in practice (within 100-200 Hz). For this actuator, the longitudinal mode shown in Figure 5-6(a) occurs at 41,034 Hz and the bending mode in Figure 5-6(b) occurs at 42,299 Hz.

A harmonic analysis was performed to predict the actuator displacement at longitudinal and bending resonance. Since the actuator displacement is small compared to the motion, the set screws were modeled as a fixed connection to the actuator body at the node locations. The ends of the set screws were then constrained with a fixed displacement boundary condition. The piezoelectric elements were modeled using solid226 coupled-field elements, which couple a voltage potential with the resulting force of the element. The actuator body is modeled as aluminum alloy and the set screws are steel. The ANSYS model with applied loads and boundary conditions is shown in Figure 5-7.

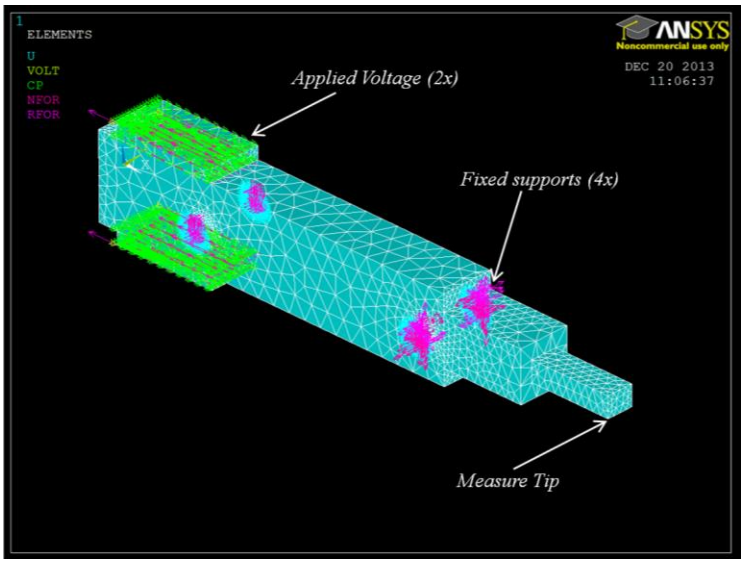


Figure 5-7. Harmonic analysis to predict resonant tip displacement.

The analysis is performed by inputting a voltage sine wave to each piezo element at the resonant frequency of each node and calculating the resulting tip motion. For longitudinal motion, the piezo elements are excited in-phase while the bending motion is simulated by exciting the piezo elements 180° out-of-phase. Simulating the harmonic motion requires a system damping coefficient, which was measured to be approximately $\zeta = 0.001$ in previous prototype designs.

Table 5-2. Simulation results for 200V input.

Mode Shape	Frequency (Hz)	Damping Ratio	Amplitude (μm)
Longitudinal	41034	0.001	6.48
Bending	42299	0.001	3.89

The ANSYS results for a 200V input were encouraging because the predicted bending mode amplitude meets the required 3 μm amplitude. The longitudinal mode was simulated to be smaller than the 9 μm required, but subsequent simulations at higher input voltages returned sufficient amplitude. The final dimensions of the actuator are presented in Figure 5-8.

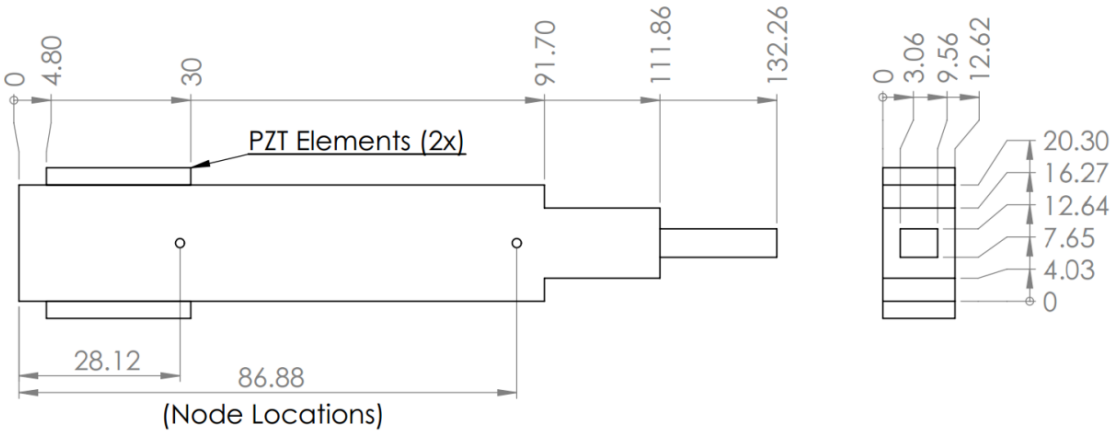


Figure 5-8. Dimensions of final 40 kHz design (all dimensions in mm).

5.2 Experimental Results

The actuator response is characterized by measuring the maximum amplitude in both the longitudinal and bending directions over a range of frequencies. An AD9838 DDS board and microcontroller are used to generate a voltage sine wave that sweeps between 38 and 42 kHz. The signal is split and one channel is phase-shifted with respect to the other using an all-pass filter, which can be varied between 0 and 180° of phase shift. MTI-2100 photonic sensors are used to measure the tip displacement and amplitude data is collected as a function of frequency. Figure 5-9 shows the actuator setup and photonic sensor mount.

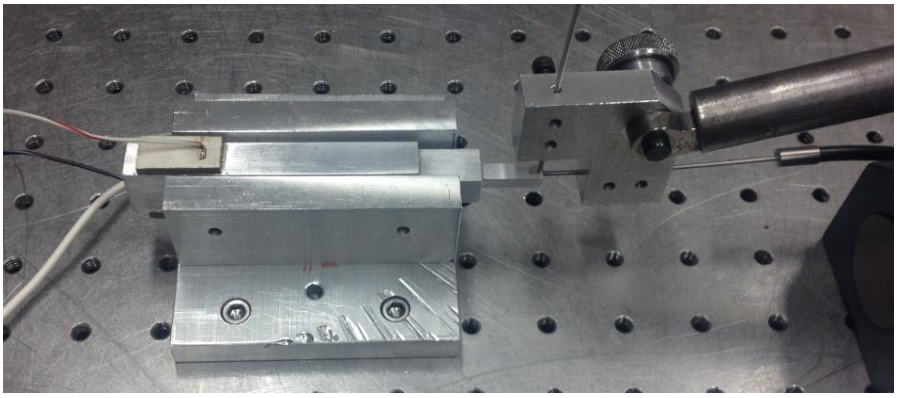


Figure 5-9. 40 kHz actuator experimental setup to measure longitudinal and bending displacement using a dual channel photonic sensor.

5.2.1 Initial Tests

Using the setup shown in Figure 5-9, a sine wave sweep was performed using two signals 90° out-of-phase. The response in the longitudinal and bending direction was recorded for a 200V and 350V amplitude input. The results for the 200V input amplitude demonstrated a 4.5 µm p-p displacement in the bending direction and 6 µm p-p amplitude in longitudinal direction. For the 350V input, the bending amplitude increased to 8.5 µm p-p and the longitudinal increased to 11 µm p-p amplitude. These displacements occurred within about 20 Hz of each other, which demonstrated successful mode matching. The displacements are shown in Figure 5-10.

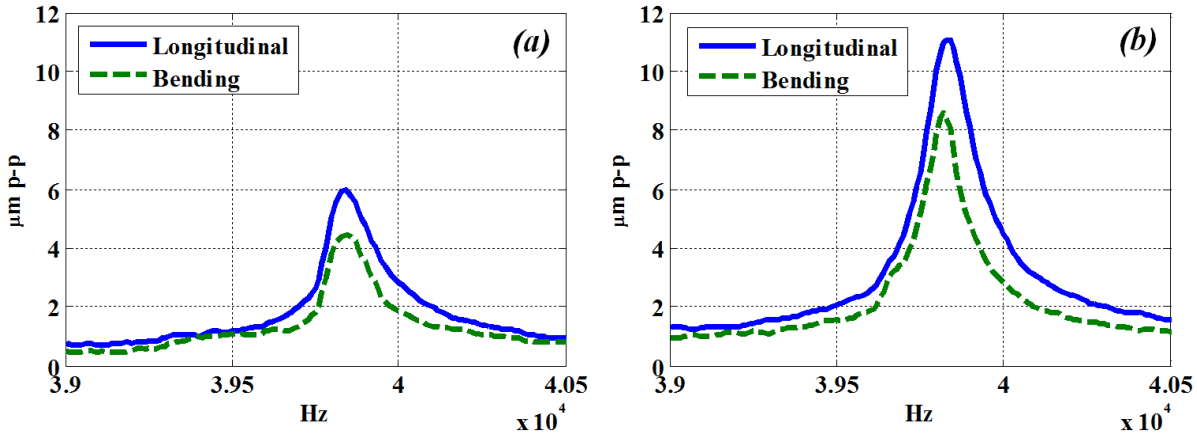


Figure 5-10. Actuator response for 200V (a) and 350V (b) inputs 90° out-of-phase.

It was observed the when the drive frequency approached the resonant frequency of the actuator; an audible noise could be heard (estimated to be around 1-5 kHz). Since the operating frequency is twice that of the human hearing range, it was assumed to be a resonance excited by the actuator motion. This noise, however, has not been observed in previous actuator designs. Another test shown in Figure 5-11 was run where the input signal was two in-phase 350V signals which should predominantly excite the longitudinal mode. When the drive frequency approached resonance, the measured amplitude peaked to 30 µm p-p. At this point, one of the piezo elements detached from the actuator at the epoxy bond. This type of failure has been the first observed instance with both the 50 and 40 kHz actuator

designs. It is possible that the epoxy was not uniformly distributed throughout the piezo surface, causing a portion of the piezo to not be properly bonded to the actuator surface. If an area of the piezo was free to move apart from the actuator, this could explain the epoxy failure as well as a lower frequency resonance that could be heard during the tests.

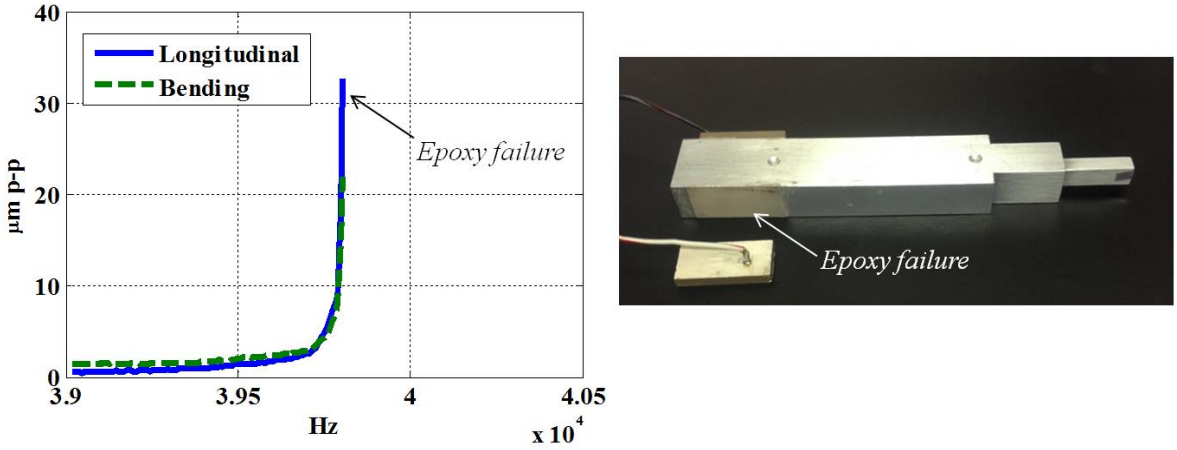


Figure 5-11. 350V in-phase input signal leading to epoxy failure.

5.2.2 Actuator Response Measurements

After the actuator was repaired by reattaching the piezo element, frequency tests were performed for several input signal phases including 7°, 45°, 95°, 150°, and 174° at 200V amplitude using the setup shown in Figure 5-9. The results were collected and the plots are shown in Figure 5-12.

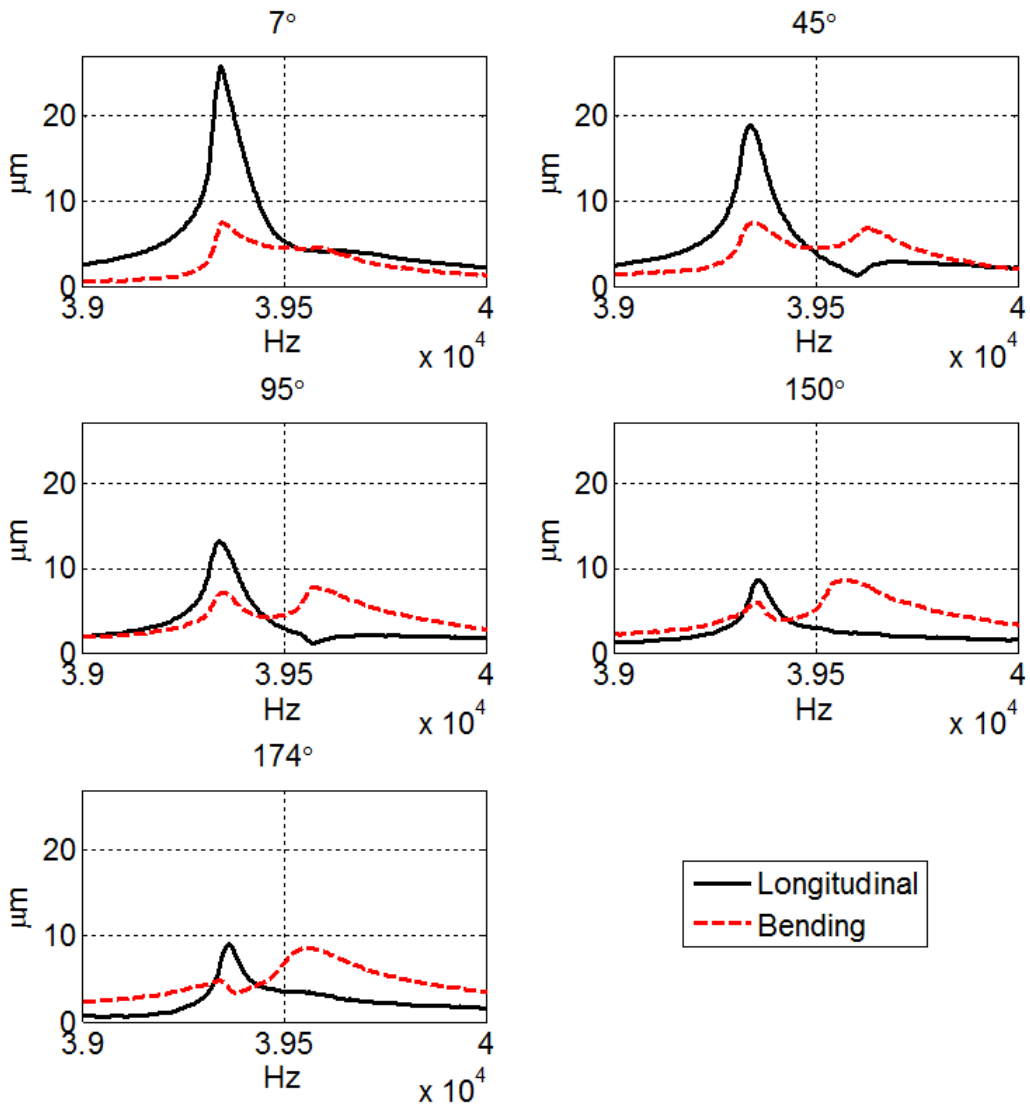


Figure 5-12. Actuator response for 200V input with varying phase (peak-to-peak measurements).

Compared with the results from the previous experiments in Figure 5-10, the actuator appears to show an increase in displacement for the 200V 90° input phase case. The actuator resonance after reapplying the piezo appeared to change, where the longitudinal and bending modes previously occurred at 39.8 kHz, the longitudinal mode in this experiment was measured at about 39.4 kHz and the bending at about 39.6 kHz. Interestingly, there was no audible noise as the actuator approached resonance that occurred in the previous experiment

prior to epoxy failure. The absence of this sound, along with the change in resonant frequencies of the modes, may suggest that the failure at the piezo-actuator interface may have been due to non-uniform adherence of the epoxy which resulted in epoxy failure.. Despite these discrepancies, the 40 kHz actuator seems to be capable of producing the required 3 x 8 μm elliptical path to perform indents with a 20x20 μm die.

5.2.3 Elliptical Path

A two channel MTI-2100 photonic sensor and a Tektronix MSO2014B oscilloscope are used to monitor the actuator motion in real-time. The actuator was driven with a two, 200V AC signals that are 100° degrees out-of-phase to excite both modes simultaneously. The actuator frequency was set to a frequency around 39,500 Hz; however, the actual drive frequency can be varied to change the relative motion between the two modes (i.e. emphasize one mode more than the other). The photonic sensors are positioned normal to the front and top face to capture both the longitudinal and transverse motion of the actuator. The elliptical motion used to perform the indents is shown in Figure 5-13.

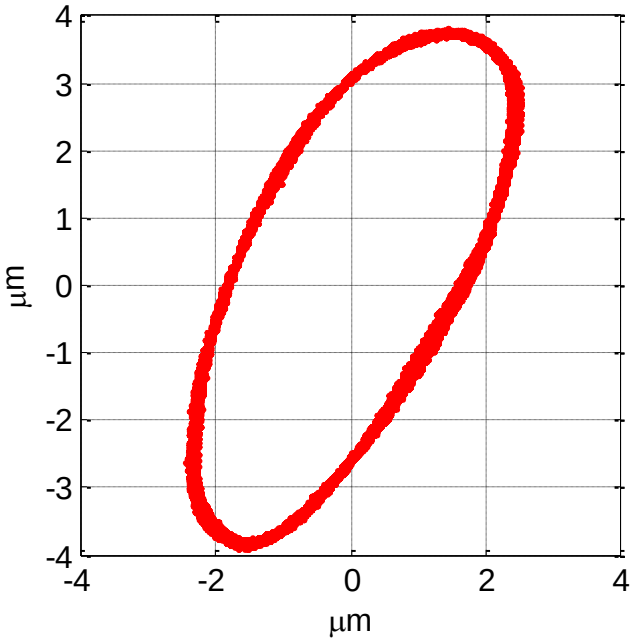


Figure 5-13. Elliptical indenting motion for a 200V AC input signal.

The elliptical motion produced by the 40 kHz design demonstrates a significantly larger amplitude response compared to the 50 kHz design in Figure 4-12. The 2.4 x 3.8 μm amplitude motion is still less than the target 3.2 x 8 μm for nanocoining with a 20 x 20 μm die, however; it will be sufficient to demonstrate the nanocoining process. Since the amplitude generated is about 75% of the target amplitude, about 25% of the die will be overlapped with successive indenting. This small area of overlap is reasonable for creating large areas of sub-micrometer features quickly.

5.3 SUMMARY

A 40 kHz elliptically-vibrating actuator was developed to achieve suitable motion for the nanocoining process. After justification was given for reducing the operating frequency of the actuator from 50 to 40 kHz, a scale factor was introduced to resize the geometry of the 50 kHz design. In efforts to increase the vibrational amplitude, the concentrating tip of the actuator was reshaped using FEA simulations and experimental measurements. Both simulation and experimental results are presented that demonstrate an increase in amplitude response compared to previous designs. The final elliptical path is slightly smaller than the desired response, however; successive indents will only overlap by around 25% which will be acceptable for the indenting experiments.

6 PROCESS CONTROL

To generate consistent indents, a feedback controller is needed to follow small changes in the resonant frequency and maintain the elliptical motion of the vibrating actuator. Since the actuator frequency is changing with respect to time, a method of synchronization is needed to modulate the spindle speed based on the instantaneous drive frequency. Feedback amplitude control was developed to ensure a consistent ellipse and a tracking mechanism to compensate for part and spindle errors in real time to maintain a constant depth of indentation was also constructed and tested.

6.1 RESONANCE TRACKING

During extended operation, the resonant frequency of the system can drift as a result of a well-documented phenomenon, such as piezo self-heating effects and other environmental influences [43, 44]. This change in resonance can be observed in the change in actuator amplitude for a constant-frequency input signal. As the resonance of the system drifts, the output amplitude typically decreases. A method for following the natural frequency to maintain a constant amplitude output and elliptical path shape. Failing to do so would results in uneven depths of the indents.

6.1.1 Butterworth Van Dyke Model

Automatic resonance tracking requires an electrical understanding of how the piezos in Figure 6-1 are behaving at system resonance. To explore these characteristics, a Butterworth Van Dyke (BVD) model is used. This model is a lumped-parameter electrical equivalent model that is used to demonstrate the interaction between the electrical side (piezos) and mechanical (resonating beam) [16, 45] .

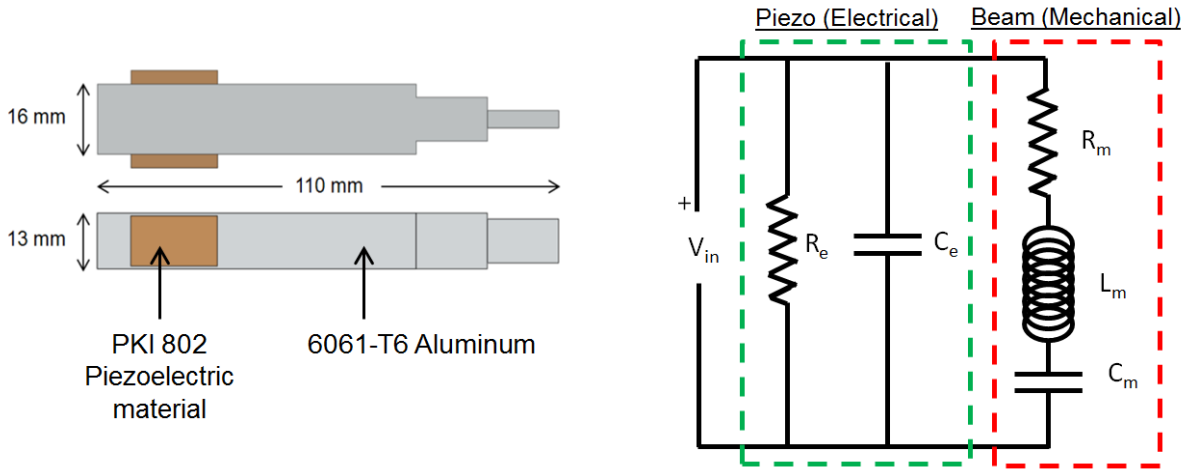


Figure 6-1. Resonant actuator and equivalent electrical model.

Shown in Figure 6-1, the piezoelectric element is modeled as a parallel RC circuit with equivalent resistance and capacitance values denoted as R_e and C_e, respectively. The electrical branch components can be determined by the electrical properties of the piezoelectric elements. The mechanical branch consists of the electrical components R_m, L_m and C_m that represent the analogous mass-spring-damper system. The mechanical electrical components are sized to approximate the frequency response of the actuator.

6.1.1.1 Impedance Analysis

Considering first the mechanical portion from Figure 6-1, combining the Laplace domain representations of the impedances of the resistor, inductor and capacitor in-series yields the impedance equation for this branch. The resulting equation is shown in Equation (6-1).

$$\begin{aligned} Z_m &= Z_{resistor} + Z_{inductor} + Z_{capacitor} \\ Z_m &= R_m + L_m s + \frac{1}{C_m s} \\ Z_m &= \frac{L_m C_m s^2 + R_m C_m s + 1}{C_m s} \end{aligned} \tag{6-1}$$

where Z_m is the impedance of the mechanical system. The electrical branch is composed of a resistor and capacitor in parallel (Piezo in Figure 6-1). Combining the equivalent impedances of these components results in Equation (6-2).

$$\begin{aligned} \frac{1}{Z_e} &= \frac{1}{Z_{resistor}} + \frac{1}{Z_{capacitor}} \\ \frac{1}{Z_e} &= \frac{1}{R_e} + \frac{C_e s}{1} \\ Z_e &= \frac{R_e}{R_e C_e s + 1} \end{aligned} \tag{6-2}$$

where Z_e is the impedance of the electrical system. After formulating the mechanical and electrical branches, the overall impedance of the system is simply a parallel combination of the branch impedances. This calculation is shown below, where Z_t it the total equivalent circuit resistance.

$$\begin{aligned} \frac{1}{Z_t} &= \frac{1}{Z_e} + \frac{1}{Z_m} \\ \frac{1}{Z_t} &= \frac{R_e C_e s + 1}{R_e} + \frac{C_m s}{L_m C_m s^2 + R_m C_m s + 1} \\ Z_t &= \frac{R_e L_m C_m s^2 + R_e R_m C_m s + R_e}{L_m C_m R_e C_e s^3 + \left(R_m C_m R_e C_e + L_m C_m\right) s^2 + \left(R_m C_m + R_e C_e + R_e C_m\right) s + 1} \end{aligned} \tag{6-3}$$

6.1.1.2 Equivalent Electrical Components

The piezoelectric material used in the construction of the actuator is Navy Type III (Piezo Kinetics PKI 802). The specifications provided by the manufacturer state that the capacitance can be calculated using the following equation:

$$C_e = \frac{K_{33}^T \varepsilon_0 L W}{t} \tag{6-4}$$

where K_{33}^{T} is the relative dielectric constant, ε_0 is the vacuum permittivity, and L, W and t is the length, width and thickness of the piezo element, respectively. The resistance R_e is also a published value based on the thickness of the element. The values for this piezoelectric material are given in Table 6-1.

Table 6-1. Published values for PKI 802 and dimensions.

Parameter		Units
K_{33}^{T}	1100	-
ε_0	8.85E-12	F/m
L	21	mm
W	13	mm
t	3	mm

6.1.1.3 Equivalent Mechanical Components

Using the impedance expression derived in Equation (6-3) to model the physical system requires equivalent electrical components for the circuit. The mechanical branch is an RLC circuit and equivalent electrical values can be calculated based on the analogous mass-spring-damper system. Table 6-2 individually compares the mechanical mass-spring-damper system components with the RLC circuit in terms of equivalent impedance where f is force, v is velocity, m is mass, K is the spring constant, D is damping, i is current, V is voltage, L is inductance, C is capacitance and R is electrical resistance.

Table 6-2. Mechanical-Electrical equivalent impedances (*s* = Laplace domain).

Mechanical Impedance			Electrical Impedance		
$Z_m(s) = \frac{force}{velocity}$			$Z_e(s) = \frac{voltage}{current}$		
Mass	$f = m\frac{dv}{dt}$	$Z_m = ms$	Inductor	$V = L\frac{di}{dt}$	$Z_e = Ls$
Spring	$v = \frac{1}{K}\frac{df}{dt}$	$Z_m = \frac{K}{s}$	Capacitor	$i = C\frac{dV}{dt}$	$Z_e = \frac{1}{Cs}$
Damper	$v = Df$	$Z_m = \frac{1}{D}$	Resistor	$V = iR$	$Z_e = R$

To begin the mapping, the spring-capacitance relationship is considered. From Table 6-2, setting $Z_e = Z_m$ for the capacitor and spring yields the following equivalent relationship:

$$C = \frac{1}{K} \tag{6-5}$$

where *K* is the spring constant and *C* is the equivalent capacitance value. To estimate the spring constant of the actuator, the equation for the stiffness of a bar in the longitudinal direction is used such that:

$$K = \frac{EA}{L} \tag{6-6}$$

where *E* is Young's modulus of aluminum, *A* is the cross sectional area of the actuator and *L* is the length of the actuator (dimensions given in Figure 6-1). With Equations (6-5) and (6-6), an equivalent capacitance can be calculated. The inductance and resistance values are then calculated using a second-order system response approximation. The natural frequency of an RLC circuit is given as:

$$\omega_n = \frac{1}{\sqrt{L_m C_m}} \tag{6-7}$$

Likewise, for an RLC circuit, the bandwidth of the resonance peak is defined as:

$$\Delta\omega_n = 2\alpha \tag{6-8}$$

where $\Delta\omega_n$ is the half-power bandwidth, and α is defined as:

$$\alpha = \frac{R_m}{2L_m} \tag{6-9}$$

The values for $\Delta\omega_n$ and ω_n can be measured by performing a frequency sweep around the resonant frequency of the actuator. The resonant frequency ω_n is simply the frequency where the maximum amplitude response is measured, while the half-power bandwidth is the frequency range between the two points where the amplitude is $1/\sqrt{2}$ of the maximum. For this actuator, the resonant frequency was measured to be 48.3 kHz and the half-band width was measured to be 120 Hz. These measurements are illustrated on the plot of the actuator's longitudinal amplitude response for a range of frequencies in Figure 6-2.

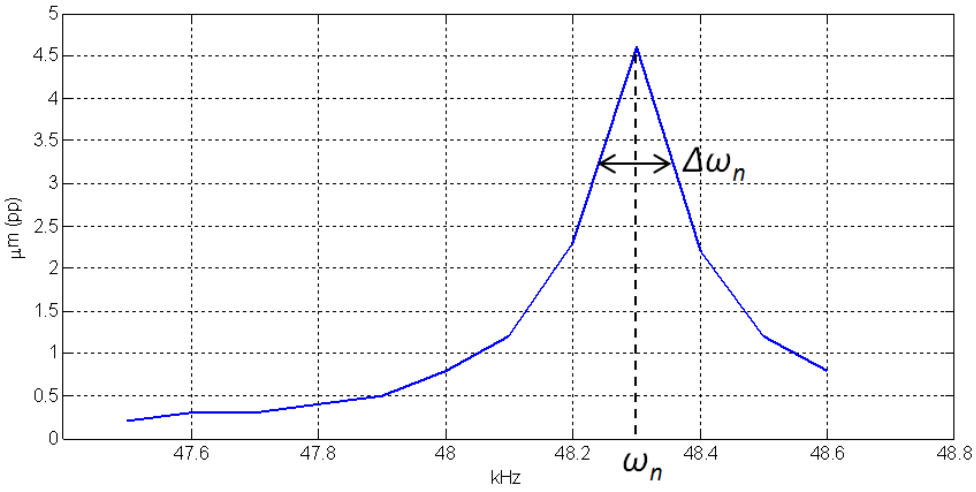


Figure 6-2. Actuator amplitude response to 100V sine wave input.

After these measurements were performed on the actuator, the result from Equation (6-5) can be used to determine equivalent inductance from Equation (6-7) and the equivalent resistance from Equations (6-8) and (6-9).

6.1.1.4 Simulations

The equivalent circuits are simulated by considering the admittance of the mechanical and electrical circuits. In a physical sense, the admittance is a position or velocity response due to a force input. Considering the impedance expressions given in Equations (6-1) and (6-2), the admittance can be expressed as:

$$Y_m = \frac{1}{Z_m} = \frac{C_m s}{L_m C_m s^2 + R_m C_m s + 1}$$
$$Y_e = \frac{1}{Z_e} = \frac{R_e C_e s + 1}{R_e} \qquad (6\text{-}10)$$

where Y_m is the mechanical branch admittance and Y_e is the electrical branch admittance. Likewise, from Equation (6-3) the combined system admittance can be expressed as the following:

$$Y_t = \frac{1}{Z_t} = \frac{L_m C_m R_e C_e s^3 + \left(R_m C_m R_e C_e + L_m C_m\right) s^2 + \left(R_m C_m + R_e C_e + R_e C_m\right) s + 1}{R_e L_m C_m s^2 + R_e R_m C_m s + R_e} \quad (6\text{-}11)$$

where Y_t is the combined system admittance. The above systems are simulated using the '*bode'* command in MATLAB which returns a phase (voltage vs. current) and frequency response for a given system. The calculated equivalent values using the analysis above for the following simulations are given in Table 6-3.

Table 6-3. Equivalent electrical component values.

Parameter		Units
R_e	900	KΩ
C_e	0.8863	nF
R_m	1.1	Ω
L_m	1.4	mH
C_m	7.66	nF

The mechanical and electrical branches are first considered separately to investigate the dynamic behavior of each component as shown in Figure 6-3. Performing an impedance analysis, it can be seen that the mechanical system has a clear resonant peak and a 180° phase shift as the frequency approaches resonance. Examining the electrical system, there appears to be attenuation in the amplitude as the frequency increase, as well as a migration of the phase towards 90°. This behavior implies that the piezoelectric element acts as a low pass filter.

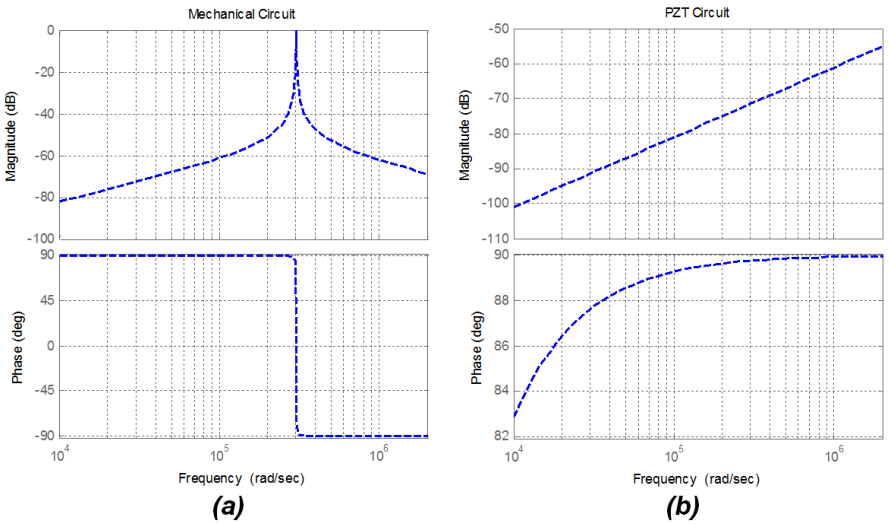


Figure 6-3. Bode diagrams for the mechanical (a) and the electrical (b) equivalent circuit.

Evaluating these two systems in parallel, as shown in Figure 6-1, allows evaluation of the combined system. Figure 6-4 shows the frequency and phase response for the total system, for a range of frequencies around resonance. As the frequency increases towards resonance, the phase decreases from 90° to -90°, with a 0° phase corresponding to the maximum admittance (or resonant frequency). As the frequency increases past the resonant peak, the admittance reaches a minimum (or anti-resonance) where the phase flips back from -90° to 90°. This phenomenon is characteristic of piezoelectric resonant actuators [46, 47].

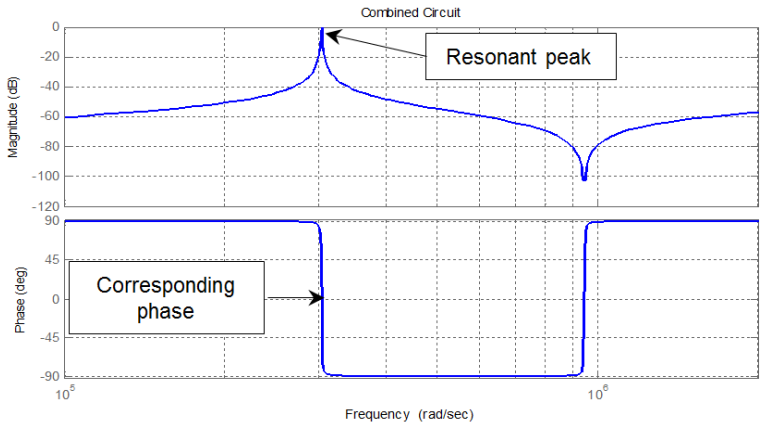


Figure 6-4. Combined electrical model for resonant actuator.

This relationship between resonance and phase is utilized in the resonance tracking control scheme. The physical manifestation of the impedance measurement is the phase between the voltage and current across the piezo elements. In this sense, the piezos act as a self-sensing mechanism where they are excited and measured simultaneously for information about the system.

6.1.2 Resonance-Tracking Control Scheme

Since the actuator must resonate to maintain its elliptical vibration motion, and method of control is needed to automatically generate a drive signal with a frequency that tracks any change in the resonant frequency. To achieve this behavior, a phase-locked loop (PLL) is used to track the phase measurement that corresponds with system resonance shown in Figure 6-4. A basic schematic of traditional PLL system is given in Figure 6-5 [48, 49].

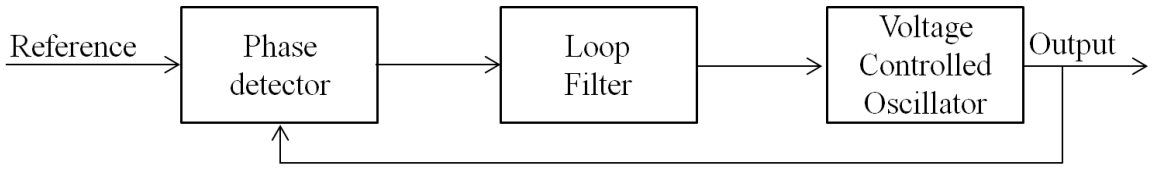


Figure 6-5. Block diagram of a phase-locked loop.

A PLL operates by comparing an AC reference signal to its output signal using a phase detector, which then generates a DC error signal proportional to the phase between the two signals. The error signal is processed by a loop filter or control algorithm and sent to the output oscillator, which changes the output frequency to drive the measured phase error to zero. This strategy is commonly used in resonance tracking of ultrasonic transducers and resonators by phase-locking the frequency of the output drive voltage to a current reference fed back from the piezoelectric element [50-52].

6.1.3 Amplifier Dynamics

Amplifiers, or piezo drivers, are used to amplify (100x) the voltage output from the signal generator up to 350V to drive the piezoelectric elements on the actuator. The piezo drivers used were TREK Model PZD350A M/S. These amplifiers have a monitor signal output port, where voltage signals are proportional to the amplified output voltage and current of the amplifier that would otherwise be too large to measure directly. The current monitor port is used as the reference signal in the PLL loop, as shown in Figure 6-6.

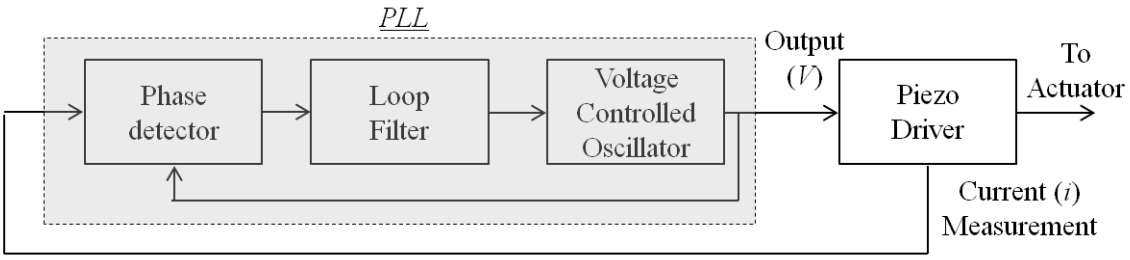


Figure 6-6. PLL with piezo driver.

Since the PLL configuration is measuring the phase between the output voltage and the current measurement, any amplifier dynamics or phase lag could be a potential source of error for the controller. In the case that the amplifier dynamics are varying with frequency, input signal amplitude, or temperature, compensations must be made in the control scheme. An illustration of these varying dynamics is shown in Figure 6-7, which demonstrates phase lag from 19° to 26° over ranges of input amplitudes of 1V, 2V and 3V. This behavior would be problematic in the event that the input amplitude is not constant (i.e. varied based on feedback measurement to maintain actuator amplitude), as the added phase lag from the amplifier would be added to the actual current-to-voltage phase across the piezos.

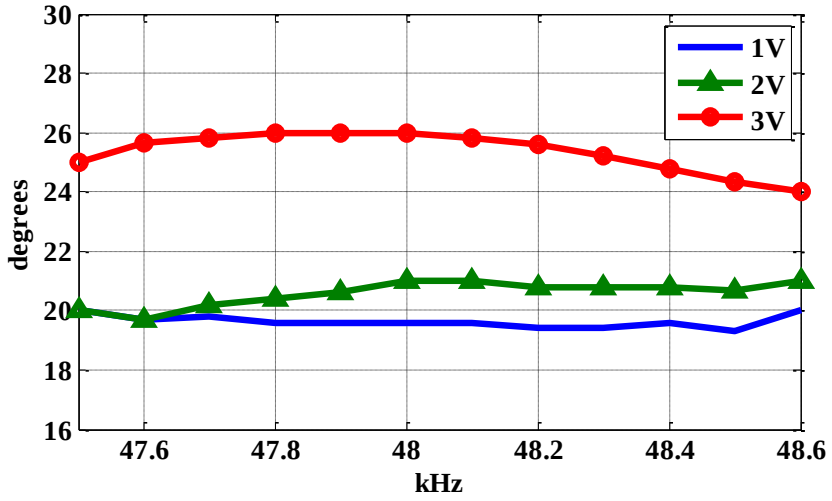


Figure 6-7. Measurement of phase between the input and output voltage signals of the piezo driver for vary frequencies and amplitude.

6.1.4 Signal Conditioning

Since there are two piezoelectric elements on the actuator that require out-of-phase excitation, the PLL output signal must be split such that two drive signals are produced (one for each piezo) with identical frequency and a phase-shift relative to each other. This can be achieved using an all-pass filter, which resembles a low pass filter but uses an op amp to make up for the low pass attenuation at the cutoff frequency. This results in a filter with zero attenuation, but retains the desired phase shift.

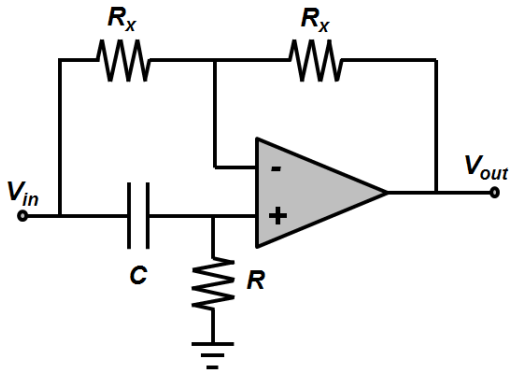


Figure 6-8. All-pass filter schematic.

Figure 6-8 shows a simple, first order all-pass filter schematic where resistor *R* and capacitor *C* is sized to produce the desired cutoff frequency, $f_c = 1/RC$. Resistors R_x are non-critical as long as they possess equivalent resistance. Variable phase can be achieved by using a potentiometer for resistor *R*, which is used to change the cutoff frequency. Assuming a constant input frequency, changing the cutoff frequency will result in changing the phase response of the filter. Simulations were performed of the schematic in Figure 6-8, where *C* is 10nF and *R* varies logarithmically between 100 and 1000Ω. The results are given in Figure 6-9.

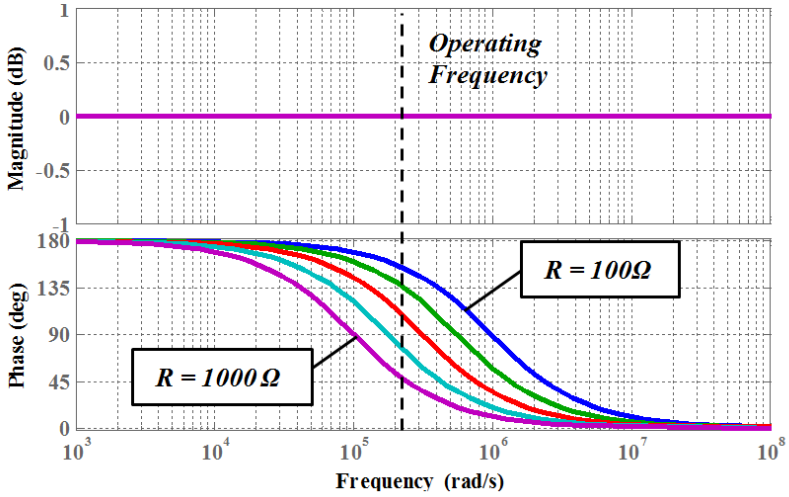


Figure 6-9. All-pass filter simulation for varying resistance, *R*.

Variable resistance can be achieved by using a potentiometer which can be adjusted, and then the output signals can be measured for resulting phase. This method however cannot be used in situ since the two sine wave drive signals must be removed from the amplifier input and measured using an oscilloscope. Selecting phase is also difficult due to the *1/R* phase response as shown in Figure 6-10(a). This plot shows that as resistance approaches zero the phase resolution decreases drastically, making fine adjustments to the input phase near 90° difficult. Since the phase is so sensitive to changes in resistance, small perturbations can also drastically compromise commanded phase which could cause unwanted changes in the actuator behavior.

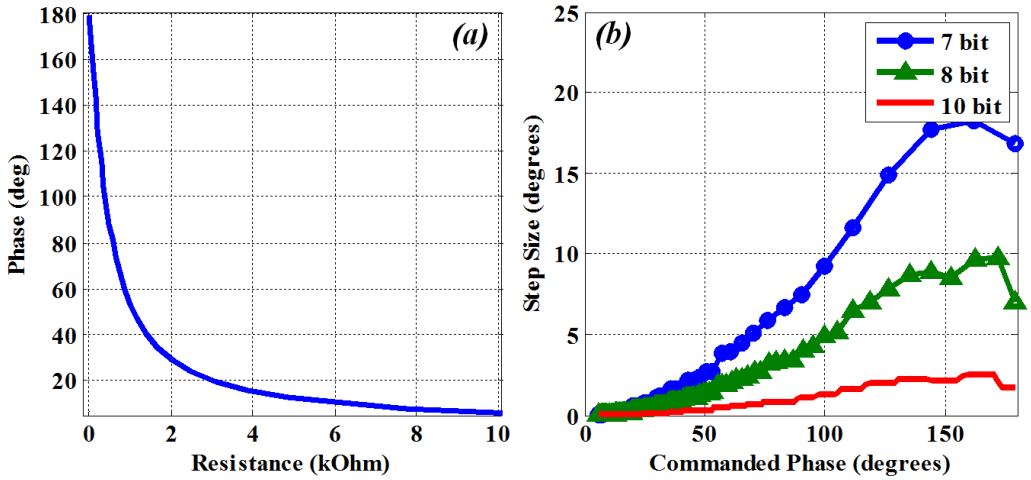


Figure 6-10. Measured phase response (a) and digital potentiometer resolution vs. step size (b).

A proposed solution for real-time monitoring and updating of input phase, as well as increased stability at lower resistances involves using a digital potentiometer (digipot). The devices are controlled using a microcontroller via serial communication (SPI). Unlike potentiometers that are continuously variable, digital potentiometers have discrete resistance increments with resolution typically between 7 and 10 bits. Figure 6-10(b) shows the resolution step size in degrees for a 7, 8, and 10 bit digipot for commanded phase based on the all-pass filter design. The same issue as the manual tuning pot arises where resolution decreases as the commanded phase increases; however, multiple digipots can be connected in-series and programmed in such a way that there is greater phase resolution. The use of a digipot in the all-pass filter system is desirable because with the use of a computer interface, input phase can be measured and set on the fly while the actuator is running. A digitally controlled potentiometer can also be placed in-the-loop for a feedback system to precisely control the phase of the actuator inputs.

6.1.5 Controller 1 – DDS

The first controller design used a phase detector chip, microcontroller and a direct digital synthesis (DDS) board to automatically track the resonant frequency. These components

create a digital implementation of the PLL control scheme shown in Figure 6-5. Here a phase detector measures the phase between the current and voltage across the piezos and a controller drives that error to zero by modulating the output frequency. A complete system schematic is shown in Figure 6-11.

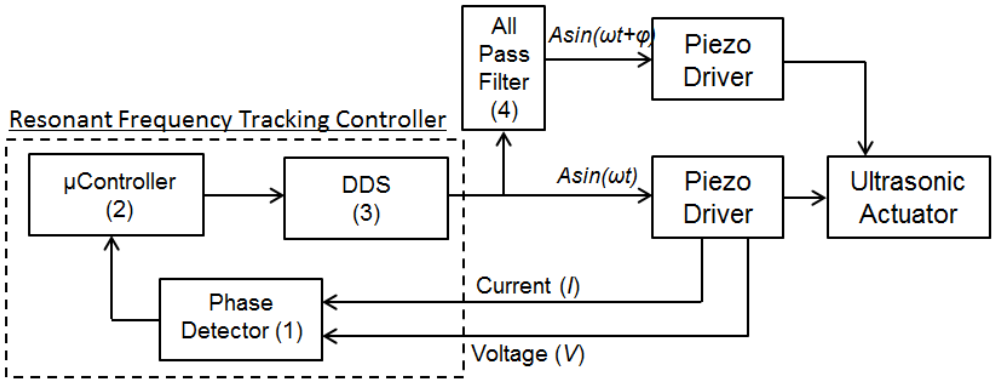


Figure 6-11. DDS board-based resonance tracking controller

6.1.5.1 Phase Detector

The phase detector ((1) in Figure 6-11) is responsible for measuring the phase between the voltage feedback signal and the current feedback signal from the piezo driver. The output of the voltage feedback is 100:1 ratio with the actual amplified output, while the current feedback is a voltage proportional to the current output (1V=40mA). A complete electrical schematic for the phase detecting system is given in Figure 6-12.

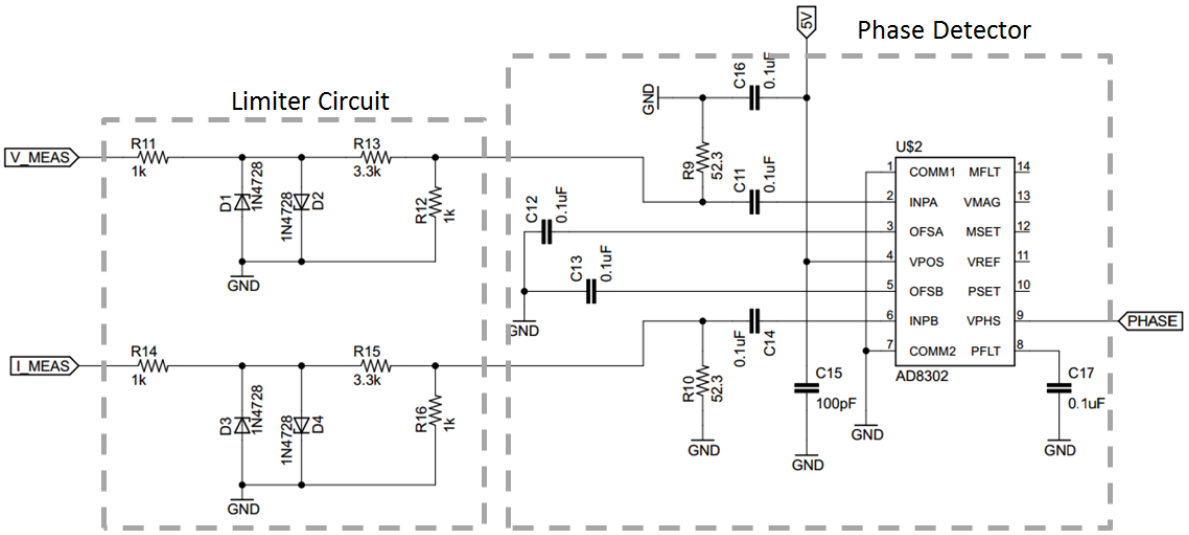


Figure 6-12. Phase detector circuit constructed for resonance tracking system.

Based on the piezo driver specifications, the potential feedback voltages can reach 10V which would damage the phase detector chip. To avoid damage, a limiter circuit is introduced using Zener diodes that will keep the input signals under 1V AC amplitude. The output of the limiter circuit for a 10V 50 kHz voltage signal is shown in Figure 6-13(a). A small time-lag is apparent in the measurement: however, since both feedback signals are subjected to the same limiter circuit the relative phase will be unaffected. Additionally, the output of the limiter resembles a saturated sine wave (close to a square wave), which was found not to affect the phase detector performance.

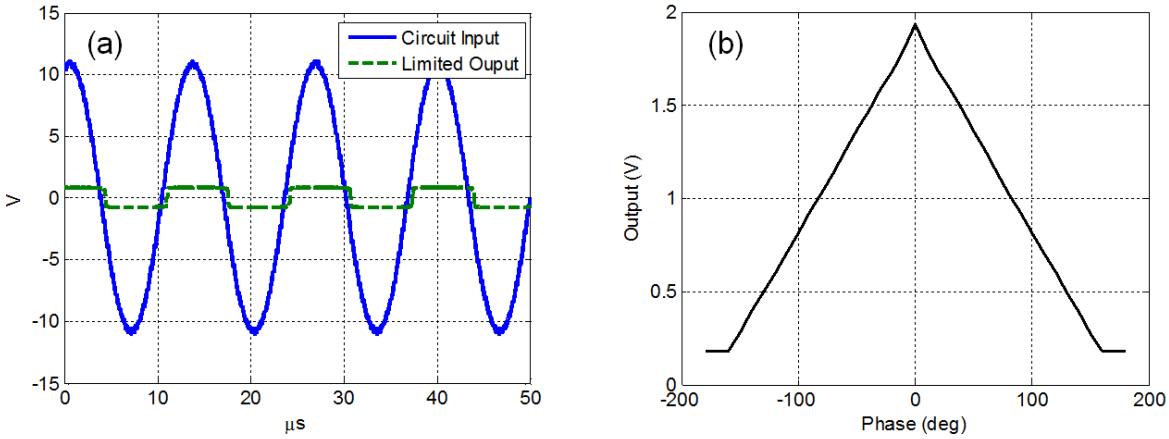


Figure 6-13. Input and resulting output of limiter circuit (a) and voltage output versus phase error of AD8302 phase detector chip.

After the input signals are limited, they are fed into an AD8302 phase detector chip that outputs a voltage proportional to the phase between the input signals (Figure 6-13). This voltage output is symmetrical about 0° phase which means the measurement could be ambiguous if the system is centered around 0°. This can be addressed by offsetting the chip output 90° per the manufacturers' instructions.

6.1.5.2 Microcontroller and DDS Board

The output voltage from the phase detector is the error signal that must be controlled by modulating the drive frequency. This system relies on an Arduino microcontroller (2) and a DDS board (3) (Shown in Figure 6-11) to read the error signal and appropriately adjust the frequency of the output signal. The microcontroller reads the analog phase error voltage and compares it with a previously determined phase set point, if the desired locked phase is not equal to zero. A software implemented proportional-integral (PI) control is used drive the phase error signal to zero at steady state. Once a new output frequency is calculated by the controller, a 28 bit tuning word is sent to the AD9838 DDS board via serial connection (SPI).

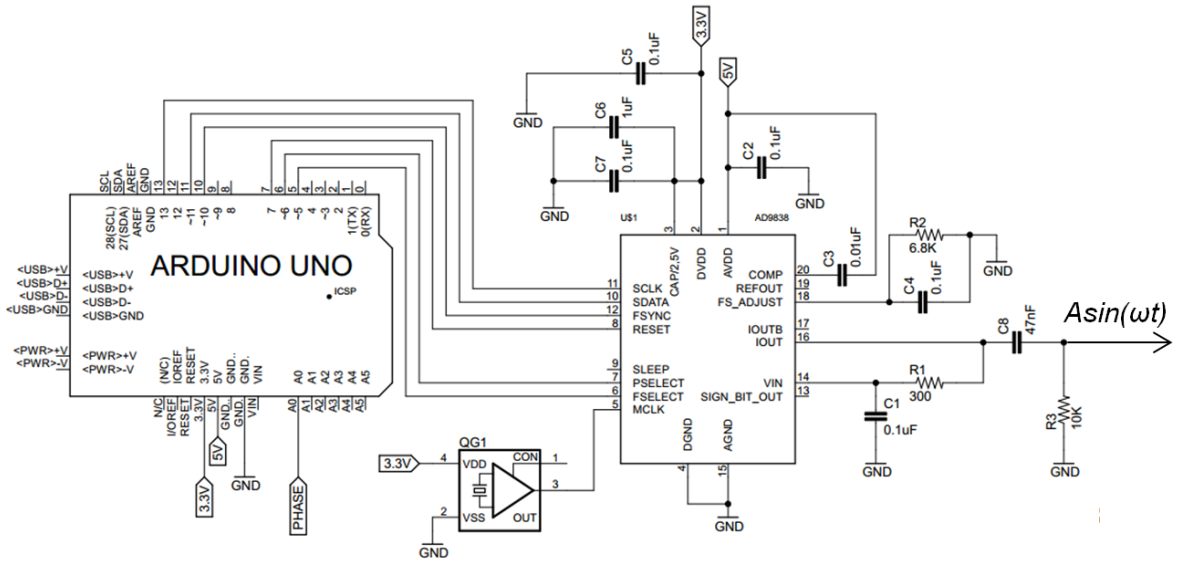


Figure 6-14. Electrical schematic of the microcontroller and signal generating board.

6.1.5.3 Signal Conditioning Circuit

From the signal generating DDS board in Figure 6-14, the updated AC drive signal is amplified using an op amp circuit with a gain of 11:1. The amplitude of the signal entering the amplifying circuit A_1 must be amplified to reach the desired output amplitude of A_2 of around 2.5V. This circuit can also be used to control the amplitude of the signal by varying the resistance of R6 in Figure 6-15.

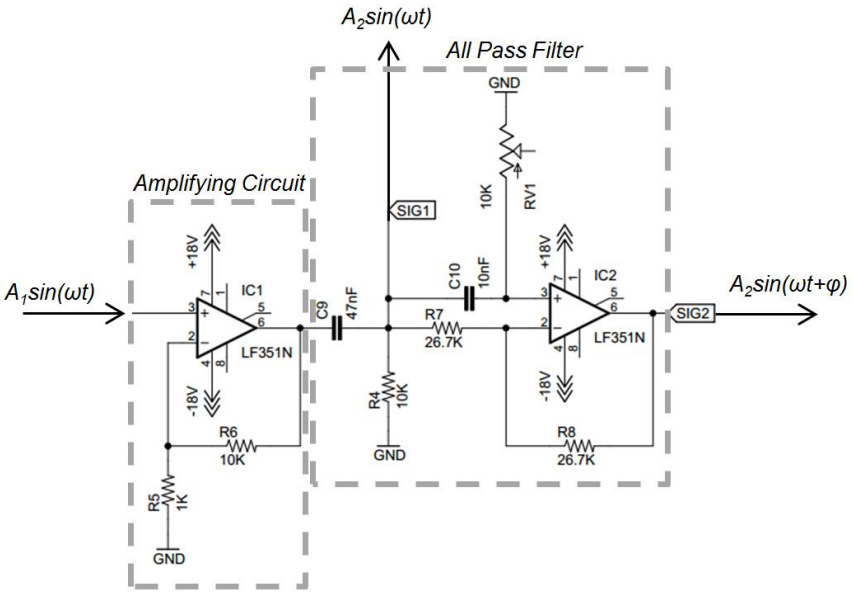


Figure 6-15. Signal conditioning with amplifying circuit and phase-shift circuit.

After the signal is amplified, it is split and one channel is sent through the all-pass filter to phase shift it relative to the unprocessed signal (Section 6.1.4). Variable resistor RV1 in Figure 6-15 is used to vary the cutoff frequency of the all-pass filter and modulate the induced phase delay between φ=10° and φ=180° to drive the piezoelectric elements of the actuator out-of-phase. After this processing, the two out-of-phase signals with identical amplitude and frequency are amplified with the piezo drivers to excite the actuator.

6.1.5.4 Performance

The closed-loop resonant tracking controller was tested on a 50 kHz, elliptically-vibrating resonant actuator and the closed-loop drive frequency and resulting longitudinal amplitude were measured over 13 minutes. Figure 6-16 shows the output amplitude of the actuator for a constant frequency input and with closed-loop control of the resonant frequency.

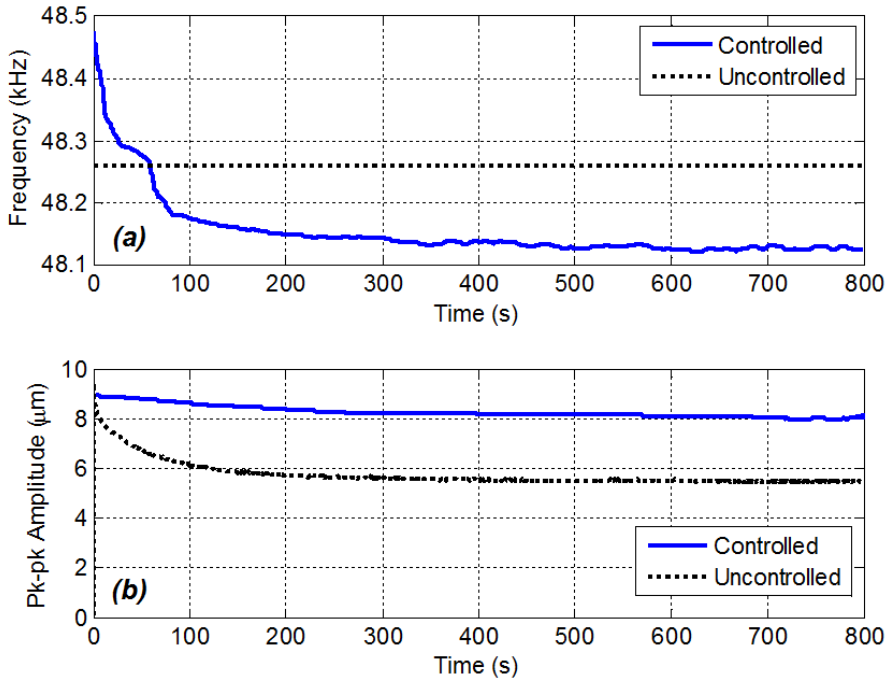


Figure 6-16. Comparison of actuator amplitude with open-loop (constant frequency) control (a) and closed-loop control (b).

It can be observed that for the uncontrolled system, where a constant drive frequency is used, the amplitude of the actuator decreases by about 3 µm over the course of 13 minutes. This implies that the resonance of the actuator is shifting, resulting in a decrease in output amplitude. Conversely, the closed-loop system demonstrates the drive frequency decreasing over time as the resonance shifts and maintaining the output amplitude to within 1µm of the start amplitude. This small decrease in amplitude in the closed-loop system can be attributed to piezo heating effects.

6.1.6 Controller 2 – Lock-In

The first controller design in Section 6.1.5 was constructed to follow the resonant frequency of the actuator by feeding back the current and voltage signals from the amplifier and measuring the phase, which was shown to work on a 50 kHz elliptically-vibrating actuator in Figure 6-16. This strategy was robust to amplifier dynamics which were measured to vary

slightly over time and with changing input levels. While this controller worked, it was limited by the resolution of the hardware components (most notably, the analog inputs on the microcontroller). To improve the performance of the controller, a lock-in amplifier was implemented.

The second resonance tracking controller utilizes an off-the-shelf lock-in amplifier to perform the phase-error measurement as well as the control algorithm and signal generation that was previously perform using a phase-detecting chip, microcontroller, and DDS board (Figure 6-11). In addition, the all-pass filter for phase delaying one channel is now controlled with a digital potentiometer, microcontroller and computer interface. The schematic for the controller using the lock-in amplifier is shown in Figure 6-17.

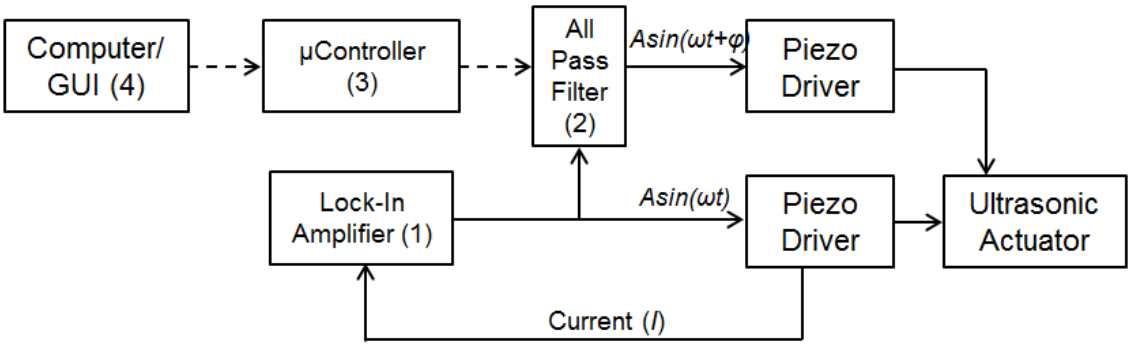


Figure 6-17. Lock-in amplifier-based resonance tracking controller.

6.1.6.1 Lock-In Amplifier

Lock-in amplifiers use a PLL system (Figure 6-5) to synchronize an AC output signal with a reference signal in frequency and in phase. While lock-in amplifiers generally use an external reference signal to set the frequency and phase of the output oscillator, some models have an internal reference which allows the user to choose the frequency of the oscillator directly. Lock-in amplifiers can be used as measuring devices in situations where the signal of interest is only a few nanovolts while the noise is several thousand times larger [53]. In

these cases, a reference is used that is synchronous with the signal that is to be measured while all the noise that is not synchronized with the reference is rejected.

For this work, the Signal Recovery Model 7270 lock-in amplifier was used to synchronize the measured current signal with the output voltage signal ((1) in Figure 6-17). The amplifier also allows the user to specify a locking phase, as well as output amplitude selection up to 5V. The input reference frequency can range from 1 mHz to 250 kHz.

6.1.6.2 Computer Interface for Phase Control

This controller uses an all-pass filter configuration (Figure 6-8) with a digitally controlled potentiometer to allow precise phase control between the two input channels. This is an improvement over the manual potentiometer from the previous design which must be adjusted and measured every time a phase change is required. The circuitry for implementing the digipot is shown in Figure 6-18.

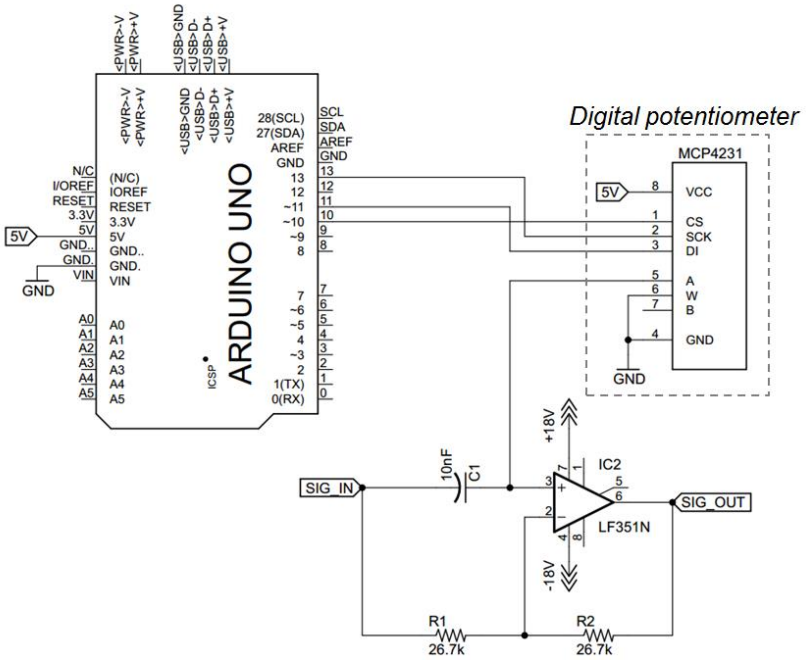


Figure 6-18. Microcontroller and digipot for computer-controlled phase.

An Arduino Uno microcontroller is used to control the resistance of the MCP4231 digital potentiometer, thereby changing the phase induced by the all-pass filter. From Figure 6-10(b), it can be seen that the highly nonlinear response of the phase of the filter versus resistance causes there to be large step changes in resolution of 7 and 8 bit digipots. The digipot used for this work uses a 7 bit register which means that around 90° of phase, the resolution of the filter is about 6° of phase. Higher resolution potentiometers would yield finer resolution in the filter and more precise control of the phase.

To alter the resistance of the digipot on-the-fly, code was written to enable serial communication with the Arduino microcontroller. Matlab GUIDE was used to develop a rudimentary interface to establish a serial communication with a specified COM port. Once communication with the microcontroller is established, the resistance value can be updated while the system is running to change the phase between the two signals. This interface is shown in Figure 6-19.

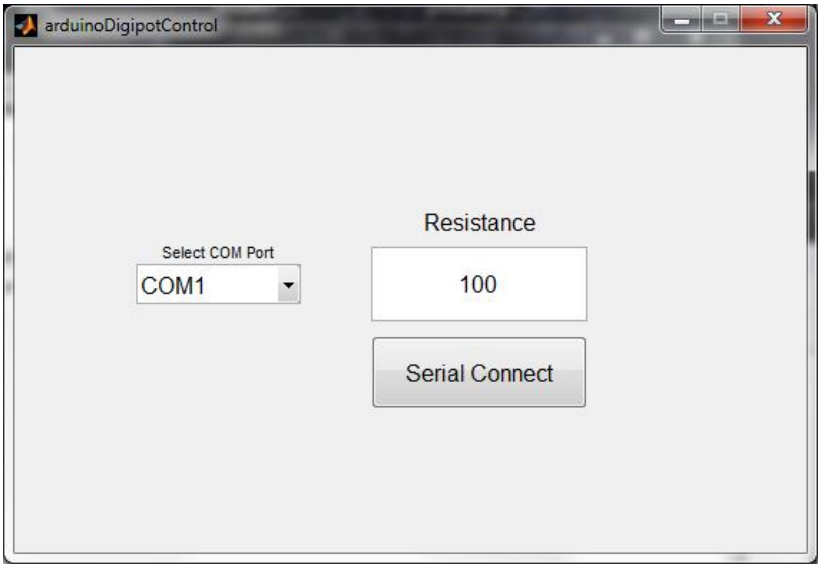


Figure 6-19. GUI for controlling the amount of phase induced by the all-pass filter.

6.1.7 Actuator Controller Summary

Two systems were developed to automatically track the resonance of the elliptical actuator.

1) The first system uses the voltage and current feedback signals from the piezo drivers to measure the phase error across the piezos. A microcontroller is used to update the frequency generated by the DDS board to drive the phase error to a desired setpoint. This system is advantageous because the amplifier dynamics discussed in Section 6.1.3 are 'in-the-loop,' where the voltage at the output of the amplifier is being compared with the output current. This system, however, has limited resolution due to the ADC inputs on the microcontroller for the phase error signal. Improvements could be made by using a higher resolution analog input for the error measurement to resolve smaller changes in the behavior of the actuator.

2) The second system utilizes the lock-in amplifier technology which is beneficial because of its high measurement resolution, as compared to the first system. This configuration is also preferable because of the digitally-controlled all-pass filter. However, this system does not take the amplifier dynamics into consideration where only the current signal is fed-back and compared with the voltage signal going to the input of the piezo drivers. This means that as the piezo driver dynamics vary over time (shown in Figure 6-7), the lock-in amplifier will not compensate accordingly. This system could be improved with an additional phase measurement that could close-the-loop on the output phase of the amplifiers to ensure that the signals going to the two piezo elements on the actuator remain at the desired phase relative to each other.

6.2 SPINDLE SPEED

The elliptical actuator performing the indents operates at its resonant frequency and that operating frequency tends to drift over time. In the previous section, the resonance-tracking controller is used to constantly update the actuator frequency to maintain the resonant

behavior. This time-varying frequency, however; is problematic when trying to create fields of uniform indents since the required spindle speed is a function of actuator frequency (Equation (1-5)). Furthermore, the crossfeed rate of the actuator is a function of spindle speed (Equation (1-7)) which also must be controlled as the actuator frequency changes. The proposed solution is to synchronize the resonance-tracking controller with the spindle speed and machine axes, as shown in Figure 6-20.

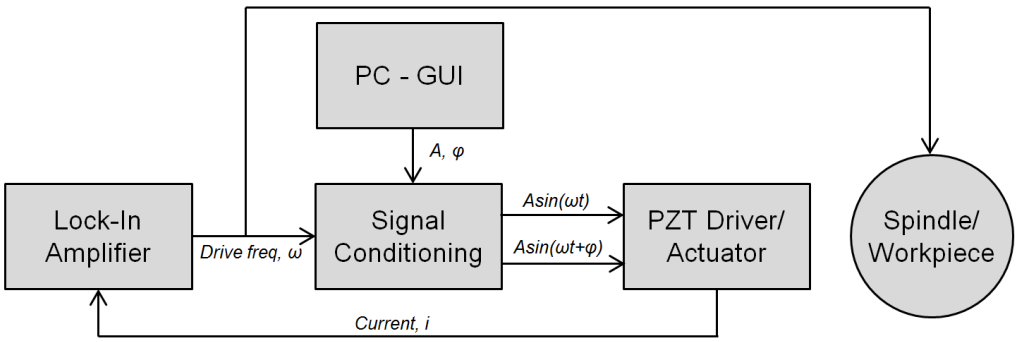


Figure 6-20. Schematic for synchronizing the actuator frequency with spindle speed.

Currently this synchronization is not implemented in the system since with small workpieces (1" diameter) the indenting time to cover the surface is only a couple of minutes. Over this time, the change in resonant frequency of the actuator is typically small enough that the spindle speed does not need to be adjusted (<5 Hz). This technique would be critical for indenting over large amounts of time (hours), where large changes in resonant frequency can occur (100's of Hz).

6.3 SURFACE FOLLOWING[1]

Compensation for roundness of the drum and radial spindle error is performed using a slower-speed piezo actuator that moves a stage containing the indenting actuator based on a cap gauge measurements. Runout error of the drum that is centered along the z-axis is measured ahead of the indenting process and moved to follow the drum surface based on

[1] Work performed by REU summer student Jonas Kessing

distance *d* as shown in Figure 6-21. The surface must be tracked to within 50 nm to maintain uniform indentation coverage.

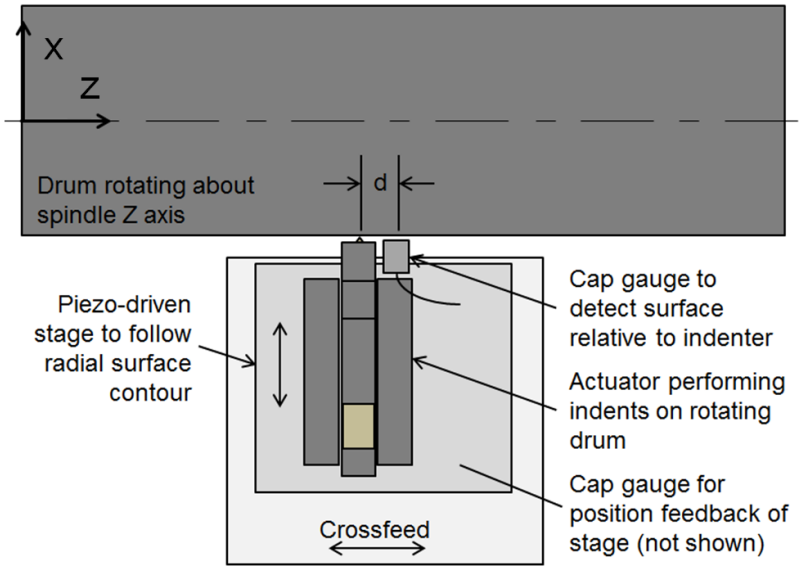


Figure 6-21. System for tracking radial part error while indenting.

Motion of the surface relative to the actuator can be compensated by moving the x-position of actuator in sync with the workpiece surface so a constant indent depth is achieved. The position of the actuator can be adjusted using the x-axis of the DTM; however, its closed-loop bandwidth is limited to about 10 Hz. To achieve a higher bandwidth than the DTM axis can achieve, a linear flexure stage was developed to allow motion in the radial direction relative to the workpiece surface. A piezo stack is used to make position corrections based on an external cap gauge measuring the error in the radial direction. A second cap gauge measures the motion of the flexure stage and forms a closed loop positioning control to track the workpiece measurement based on the external cap gauge. Since the external cap gauge is distance *d* in the z-direction ahead of the indenting point, corrective stage actuation must be delayed based on the crossfeed rate. This ensures that the measured error at the cap gauge is being compensated for along the z-axis where the indenting is actually being performed. The system arrangement is illustrated in Figure 6-22.

The flexures were designed to only allow deflection of the stage in the lateral direction relative to the actuator body and provide sufficient stiffness to preload the piezo stack. To avoid a decrease in actuation range, the preload stiffness of the flexures was designed to be about 10% that of the piezo stack stiffness [54].

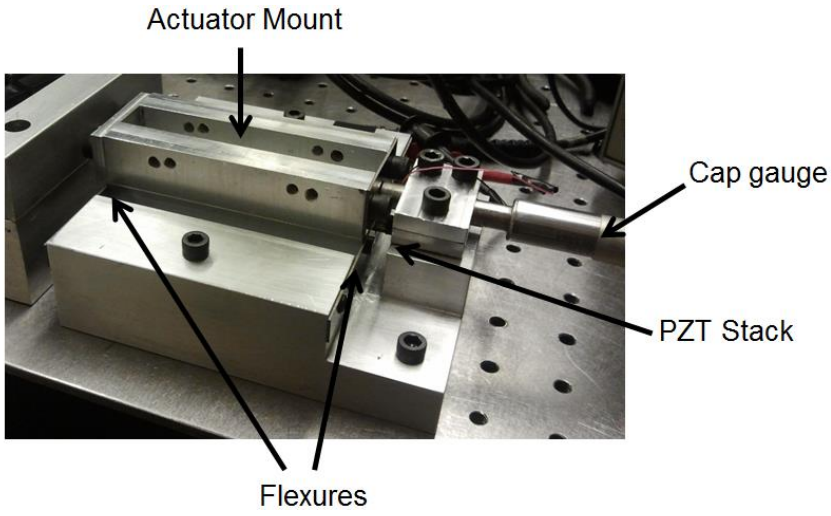


Figure 6-22. Flexure stage setup with actuator mount.

The open-loop natural frequency of the system was investigated by exciting the piezo stack with an AC signal from 0 to 5000 Hz, where the natural frequency of the system was measured to be at 1000 Hz. The resulting output magnitude and phase are shown in Figure 6-23.

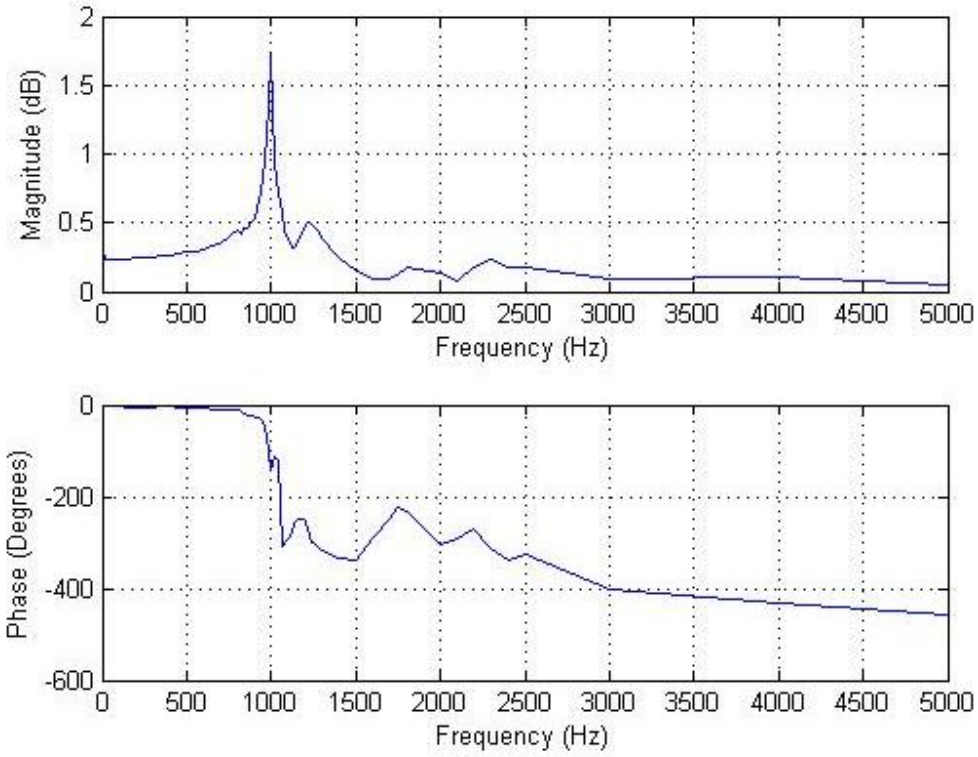


Figure 6-23. Open loop frequency response of piezo-driven flexure stage.

The system was controlled using a National Instruments compact RIO (cRIO) device, which was programmed in LabView. The cRIO used two analog inputs, one for each cap gauge, and one analog output for the signal to the amplifier. The cRIO had a 400 MHz real-time (RT) processor and a Field Programmable Gate Array (FPGA). LabView was used to write virtual instruments (VIs) on the RT processor and FPGA as well as on the desktop computer. The RT processor was much faster than the computer and the FPGA was much faster than the RT processor. However, the FPGA could only perform a limited number of commands per cycle. The FPGA read in the analog input voltages, filtered them with a PID filter, and output the analog voltage that was sent to the piezo stack (Figure 6-24). Based on the 1000 Hz natural frequency of the system measured in Figure 6-23, the bandwidth of the system is filter-limited to 100 Hz.

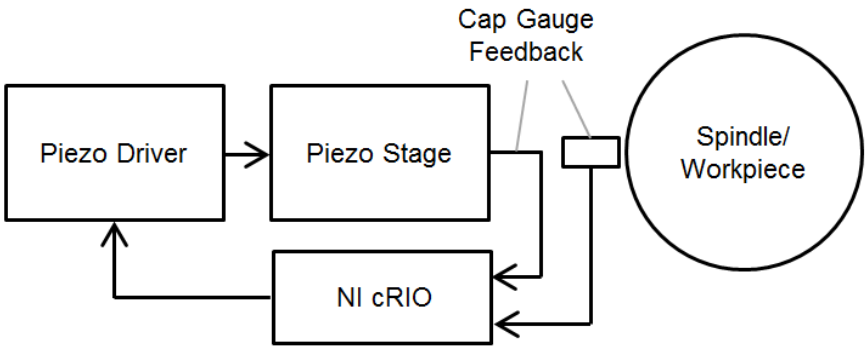


Figure 6-24. Surface error following schematic for feedback.

The system has demonstrated using a 1" diameter diamond turned aluminum cylinder with a 1 μm arbitrary error. Figure 6-25 shows the part measurement and the corresponding stage cap gauge measurements taken in closed-loop operation, as well as the difference between the measurements that represents tracking error. While rotating at 500 rpm, once-per-rev errors are compensated to keep the actuator within 50 nm of the surface.

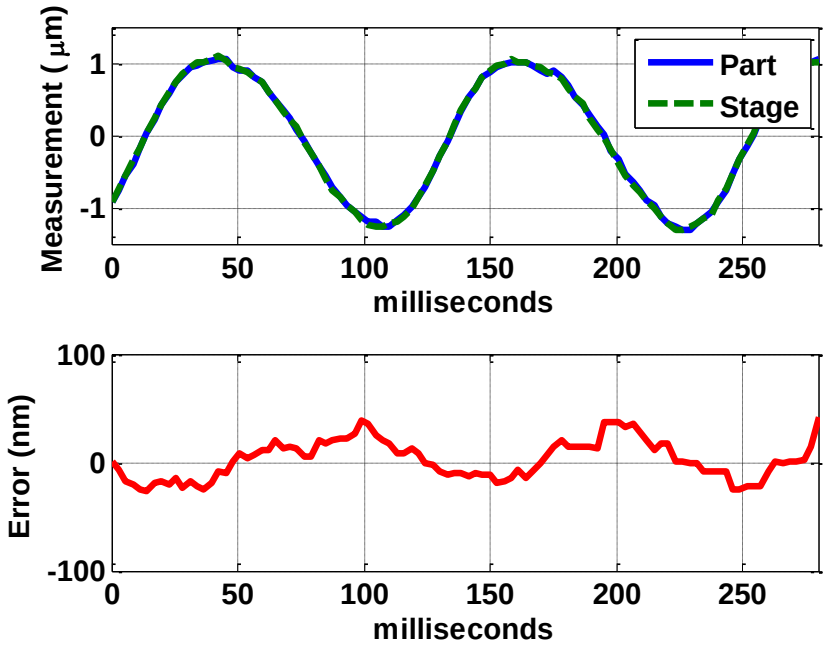


Figure 6-25. Measured following error of 1 inch diameter cylindrical workpiece.

6.4 AMPLITUDE CONTROL

The resonant tracking method used to determine the instantaneous drive frequency of the actuator is useful in maintaining resonant behavior. However, due to heating and other environmental influences, resonance tracking does not explicitly guarantee the amplitude at resonance will remain constant. A method of feedback amplitude control is proposed where piezoelectric strain sensors are used to measure the actuator amplitude and make necessary changes to the drive signal amplitude to ensure constant displacement at the indenting tip. A schematic of the actuator control is given in Figure 6-26.

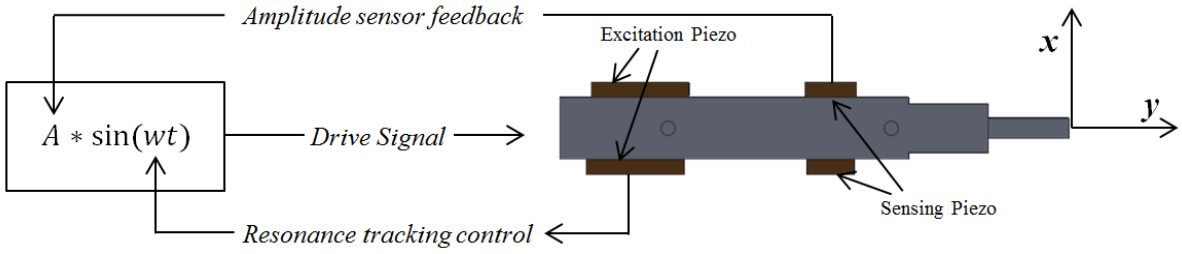


Figure 6-26. Schematic of actuator with excitation and sensing piezo elements.

The piezo amplitude sensors are positioned along opposite sides of the beam in a position such that they do not lie directly on a bending or longitudinal node. FEA was used to predict the mutual position that would induce the maximum strain on the sensors in both the longitudinal and bending directions (Figure 6-27).

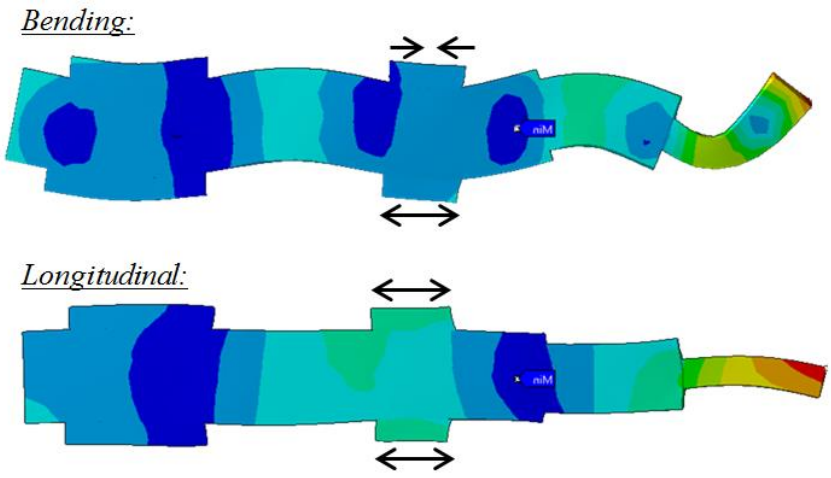


Figure 6-27. Resonant mode shapes and their influence on the sensors.

The two sensors are positioned opposite each other such that the longitudinal component of the two outputs would be in-phase and the bending component would be opposite for the two signals. Equation (6-12) demonstrates how this property can be mathematically implemented to isolate the longitudinal and bending motion measurements.

$$y = \frac{sensor_1 + sensor_2}{2}$$
$$x = \frac{sensor_1 - sensor_2}{2} \tag{6-12}$$

where x and y are the bending and longitudinal tip motion, respectively, and $sensor_1$ and $sensor_2$ are voltage signals from the induced strain on the sensing piezos. Experiments were run to evaluate this sensing approach, where a 2D actuator was first powered with two, in-phase drive signals to emphasize the longitudinal motion. The results of this experiment are shown in Figure 6-28.

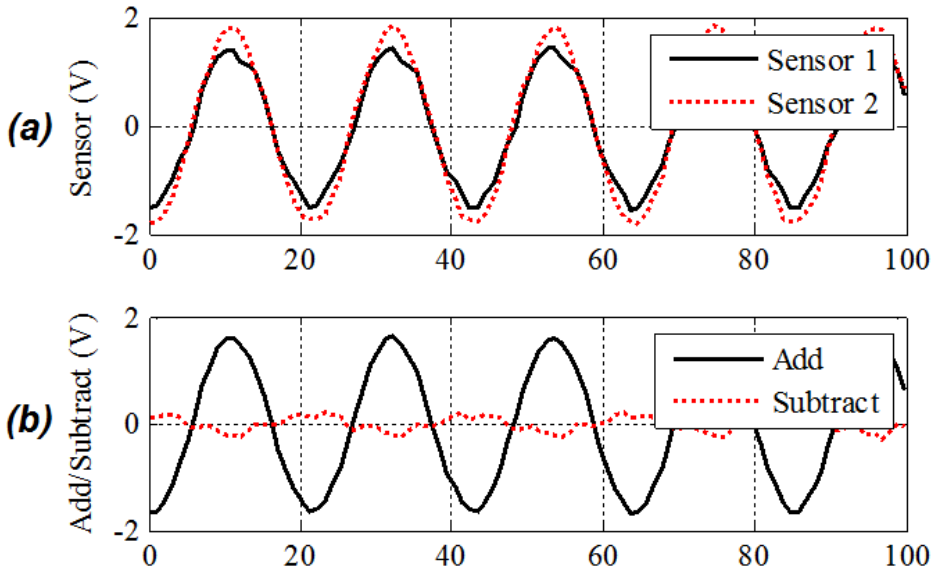


Figure 6-28. Sensor voltage outputs (a) and processing of the signals (b) for the longitudinal excitation case.

Figure 6-28(a) shows in-phase sensor signals of similar magnitude which would be indicative of longitudinal motion since both oppose sensors are being excited in the same manner. Using Equation (6-12) to isolate the longitudinal and bending motion (*y* and *x*, respectively) yields the results in Figure 6-28(b). This plot clearly shows large motion in the y-direction and little motion in the x-direction.

The next experiment examined this technique for piezo drive signals that were 180° out-of-phase, emphasizing the bending mode.

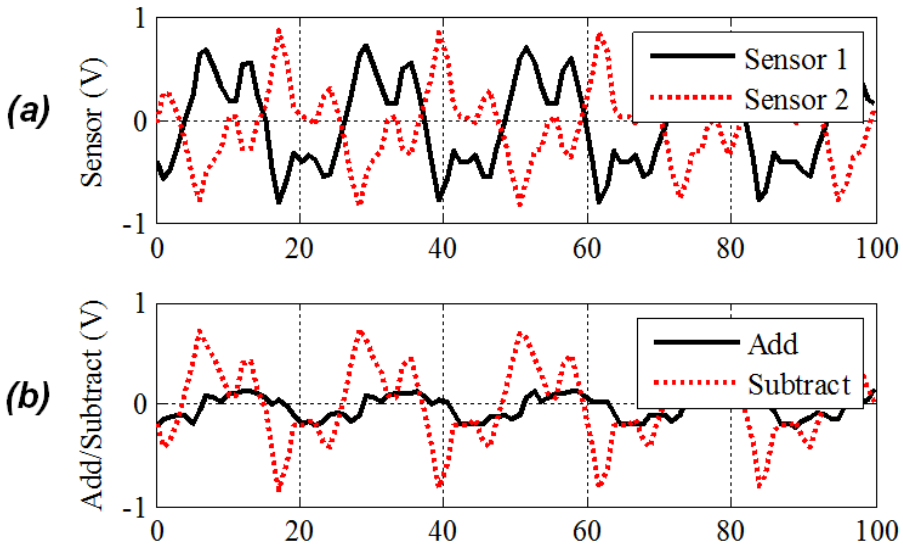


Figure 6-29. Sensor voltage outputs (a) and processing of the signals (b) for the bending excitation case.

The results shown in Figure 6-29(a) seem to show the two sensor signals 180° out-of-phase which would be consistent with the bending direction of actuator. The signal processing using Equation (6-12) shows the y-direction motion relatively small compared to the x-direction motion which would be consistent with primarily bending motion (Figure 6-29(b)). These sensing results also agree with the corresponding tip measurement for the bending motion case shown in Figure 6-29(c). Once notable difference between the sensor measurements for the longitudinal and bending case is that the bending case seems to contain higher order noise whereas the longitudinal motion tends to be cleaner and more closely resemble a sine wave. This could be due to asymmetries in the sensor placement or actuator construction.

6.5 SUMMARY

System control is necessary to produce large areas of tiled die indents due to the drifting resonant frequency and amplitude of the actuator and potential radial part motion of the workpiece. Two resonant tracking control system were designed and implemented based on a traditional PLL design. The first system integrated a phase detector chip, microcontroller

and DDS signal generator board to measure the phase between the voltage and current across the piezos and update the drive frequency to maintain a desired phase setpoint. This system is advantageous because the piezo driver dynamics are 'in-the-loop', such that the controller intrinsically compensates for potential time-varying amplifier dynamics.

The second resonance tracking system utilized an off-the-shelf lock-in amplifier to measure the phase between the piezo current and output voltage drive signal. This system can potentially achieve better performance than the first system due to the high resolution of the phase detector and signal generator (±0.001 degrees and 0.001 Hz, respectively). A digital potentiometer controlled by a microcontroller was also implemented where the phase between the two drive signals could be updated via serial connection to a Matlab-based user interface. A drawback of this setup is that because the current is referenced to the output voltage of the lock-in amplifier, and not the piezo driver, the induced dynamics of the drivers would not be compensated in the control system. This may prove problematic for longer indentation times (hours)

To compensate the radial errors in the part and to maintain consistent indent depths, a surface-tracking, piezo-driven, flexure stage was developed. This apparatus used a capacitance gauge to measure the surface of the part and correct errors by moving the actuator accordinly, achieving much higher bandwidths than using the DTM axis (100 Hz versus 10 Hz). The test setup demonstrated less than 50 nm of tracking error for a 1" diameter workpiece rotating at 500 rpm with a 1 μm off-center error.

Finally, a feedback amplitude control concept was proposed to measure the two directions of motion of the elliptical actuator to ensure the indenting path dimensions remain constant. This design used two, opposing piezo strain sensors whose signals could then be added and subtracted to isolate the longitudinal and bending motions of the actuator. The preliminary results were successful in demonstrating the feasibility of this technique.

7 INDENT EXPERIMENTS

7.1 EXPERIMENTAL SETUPS

Since the indents must be held consistently at a depth of about 500 nm and positioned to less than 1 μm relative to each other, high-precision Diamond Turning Machines are required to orchestrate the workpiece and actuator position. A cylindrical workpiece was mounted on the spindle of the DTM and cut with a single point diamond tool to produce an optical quality finish (Ra < 10 nm). The actuator is mounted on the y-axis and aligned such that the die is square with the periphery of the workpiece. While the spindle is rotating the workpiece, the vibrating actuator is brought into contact with the workpiece using the x-axis, and crossfed in the z-direction to imprint the die pattern around the periphery of the workpiece (Figure 7-1). This work utilized two diamond turning machines which will be described.

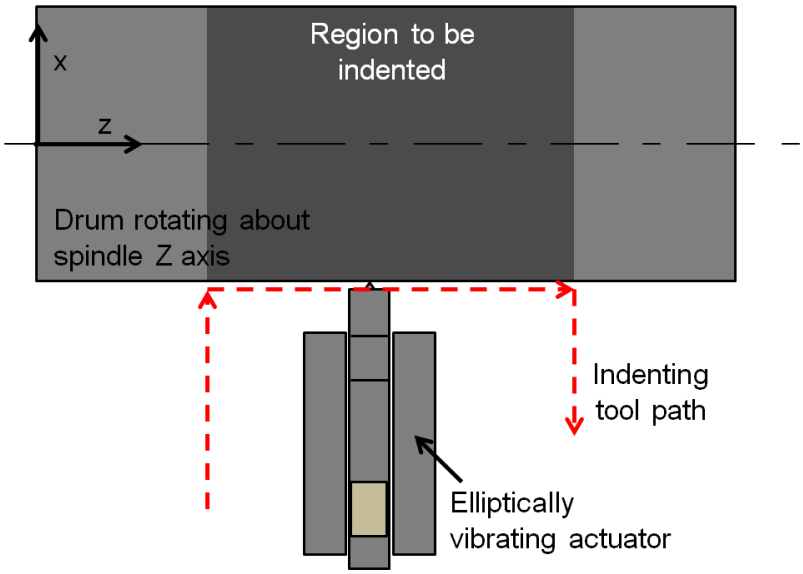


Figure 7-1. Top view of tool motion for ultrasonic indenting.

7.1.1 ASG

The first nanocoining setup uses a Rank Pneumo ASG-2500 2-axis diamond turning machine shown in Figure 7-2. The cylindrical workpiece is mounted on the spindle which moves in

the z-direction and the actuator is mounted on the x-axis. The z-axis is used to crossfeed the actuator along the length of the diamond turned workpiece. Since there is no y-axis, a microheight adjuster is used to manually set the vertical height of the actuator relative to the spindle centerline.

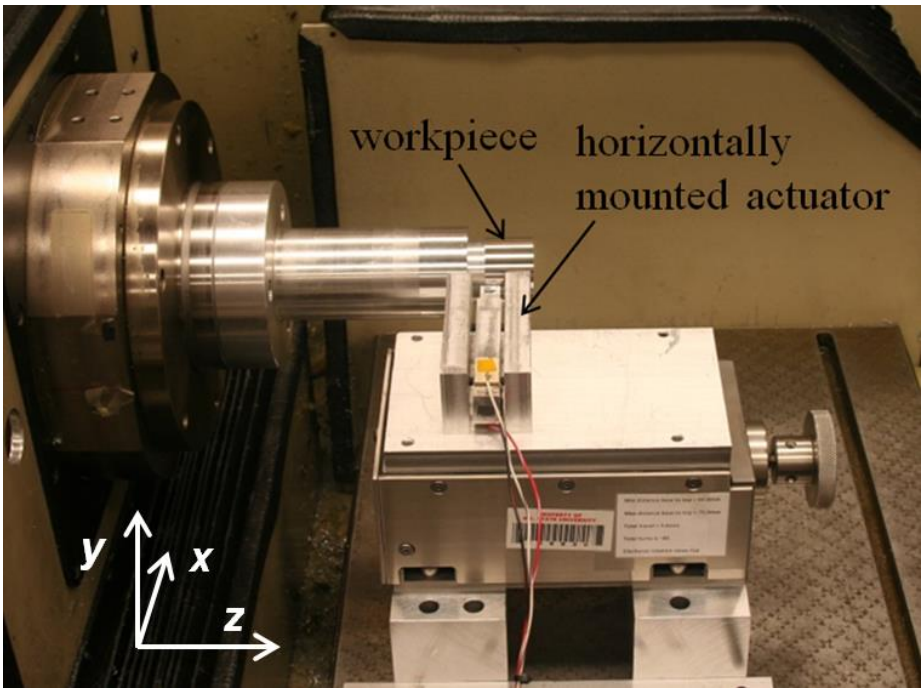


Figure 7-2. Nanocoining setup on ASG

7.1.2 Nanoform

The second nanocoin setup uses a Rank Pneumo Nanoform 600 with a Moore Nanotechnology Systems y-axis, resulting in a 3-axis (+ spindle) DTM shown in Figure 7-3. The workpiece is mounted on the spindle which moves in the x-direction while the actuator is mounted on the y-z axis. The actuator is crossfed along the length of the part in the z-direction to produce areas of indents. The significant difference between the ASG and Nanoform is that the Nanoform has a controllable y-axis which is convenient in adjusting the height of the actuator on the machine. Another difference is the axes arrangement, where the spindle is mounted on the z-axis of the ASG, but the x-axis of the Nanoform. Likewise, the

actuator is mounted on the x-axis of the ASG, but the z-axis of the Nanoform. These differences, however; do not significantly impact the indenting experiments.

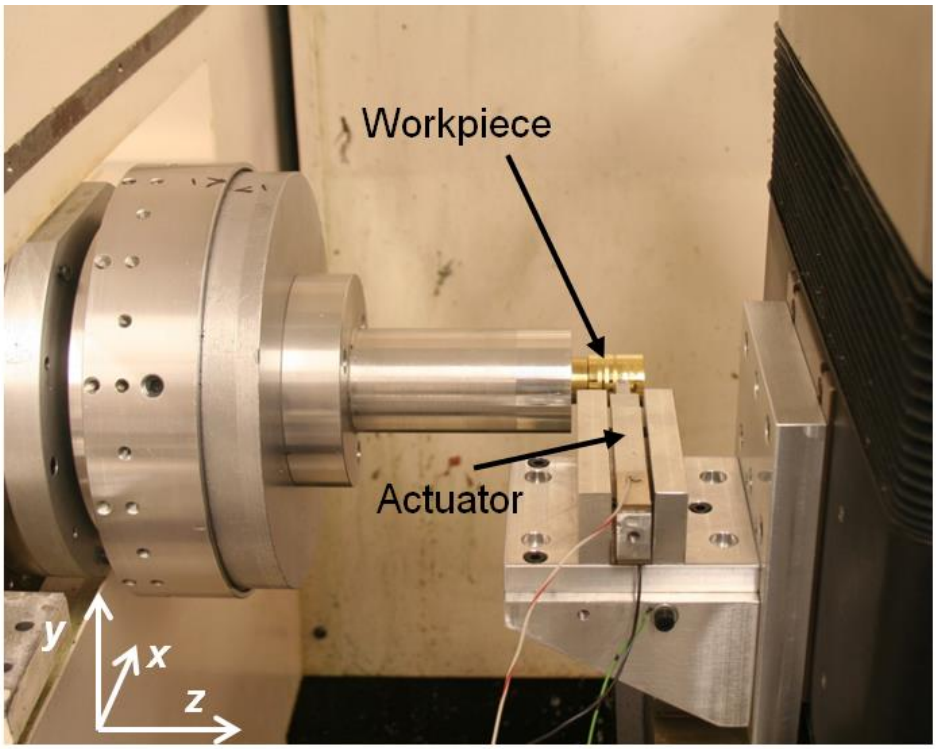


Figure 7-3. Nanocoining setup on the Nanoform.

7.1.3 Alignment

During indentation, the surface of the die must be normal to the surface of the workpiece to ensure the even distribution of features created in a single indent. An example of a misaligned die during indention of a brass workpiece is shown in Figure 7-4. It can be observed that for each indent, only a portion of the features are being replicated which implies that the die is titled and only contacting over part of its area. This error must be compensated to create uniform coverage of features.

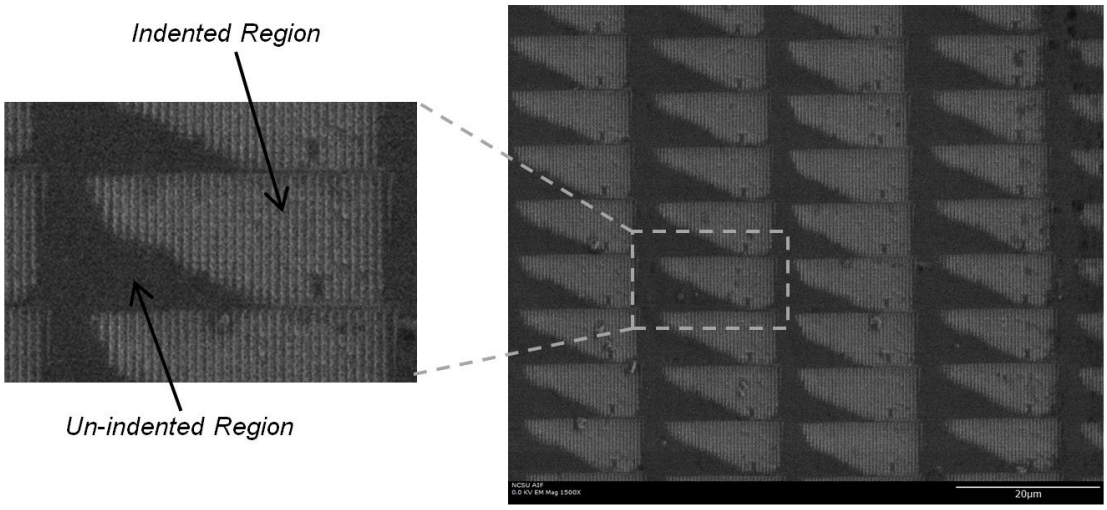


Figure 7-4. SEM of indents with a misaligned die resulting in partial feature creation.

This die tilt can be measured quantitatively using a Zygo NewView white light interferometer. Since this measurement device uses white light, the sub wavelength features created by nanocoining cannot be observed; however, the angle of the individual indents can be measured to estimate the alignment error or the actuator. An example of this NewView measurement for an array of about 6 full indentations is shown Figure 7-5, where it can clearly be seen that each indent is tilted and the result is an uneven surface.

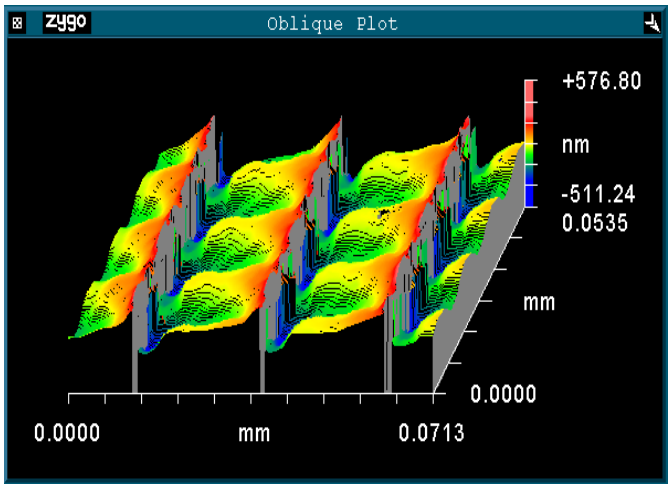


Figure 7-5. Zygo NewView measurement of an indented surface.

7.1.3.1 Correcting Alignment Errors

Die tilt is characterized by the pitch, roll, and yaw of the actuator body with respect to the surface of the drum. The rotational axes of these alignment errors are defined in Figure 7-6.

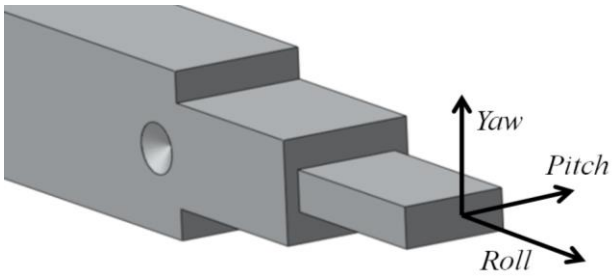


Figure 7-6. Axes of the actuator about which tilt, roll and yaw occur.

Since the face of the die is 20x20 μm and is not feasible to measure in situ, reference surfaces are chosen on the actuator near the location of the die. It is standard operating procedure to perform 'test indents,' where a region of indents is generated and tilt error can be measured, which can then be related to the position of the reference surfaces on the actuator body. A touch probe is used to measure the reference surface by rastering the axis in the direction of the tilt of interest and measuring the slope of the surface. An example of referencing the actuator position using a touch probe is shown in Figure 7-7.

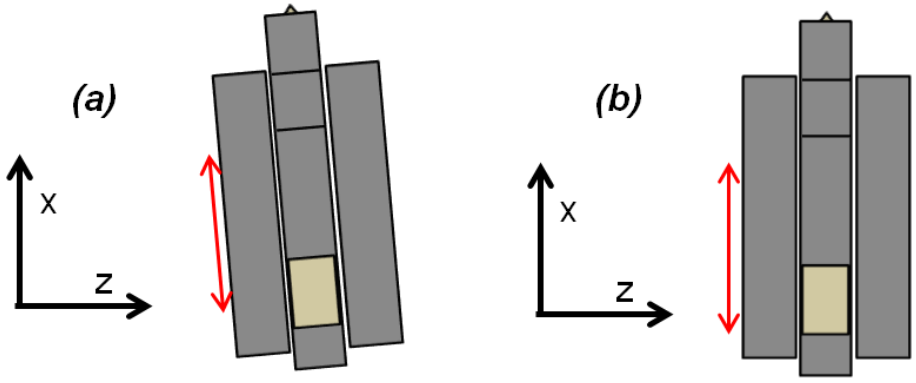


Figure 7-7. Measuring yaw with respect to x-axis where (a) has a yaw error and (b) is corrected assuming the die is orthogonal to the actuator mount.

Fixing an electronic indicator to the x-axis allows measurement of the tilt (deflection in the z-direction) to be made and the actuator's position correlated with test indents. Based on a NewView measurement as in Figure 7-5, the known tilt caused by the yaw error can be corrected for even though the die is not necessarily orthogonal to the reference surface. The slope of the indents can be calculated and that slope can be corrected for by loosening the mounting bolts and tapping the actuator into the appropriate position as in Figure 7-7(b).

Pitch is also critical for generating uniform indents and can also be measured using the Zygo NewView interferometer. As with the yaw correction, the indent profile can be extracted from the NewView measurements and a slope calculated (Figure 7-8(a)). This pitch can be corrected for using the y-axis position of the actuator shown in Figure 7-8(b).

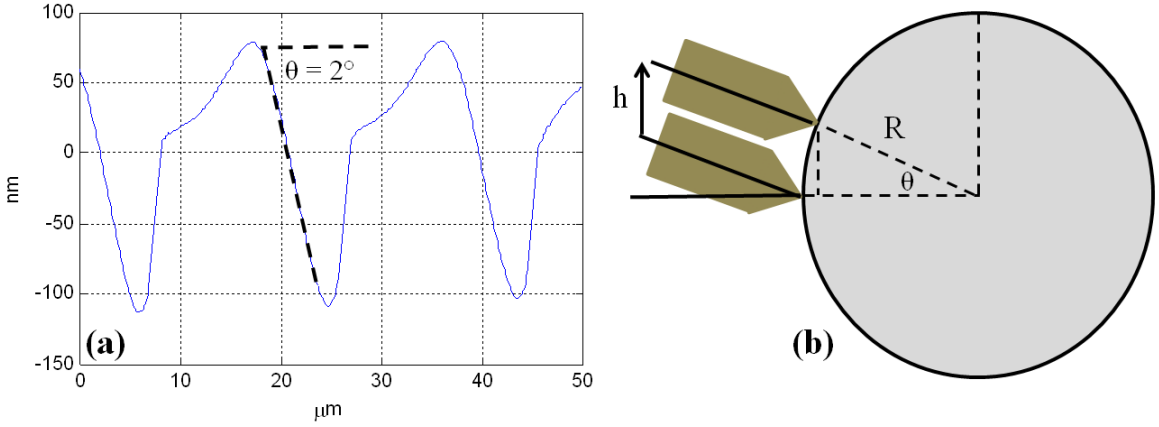


Figure 7-8. Measured tilt error profile of test indents (a) and tilt correction using y-position (b).

Due to the nature of the actuator position and cylindrical workpiece, tilt errors can be corrected by adjusting the vertical y-axis to move the actuator relative to the spindle centerline. This method is preferable to the alternative of shimming the actuator to correct for tilt because it offers much greater adjustment resolution. The actuator would need to be moved a relatively large amounts to make small changes in the relative angle of contact

between the die and workpiece. Equation (7-1) is used to calculate the amount of height adjustment h is required for a tilt error θ and workpiece of radius R.

$$h = R\sin\theta \tag{7-1}$$

Finally, the roll error where the die is rotated about the centerline of the actuator is not explicitly addressed in this work. This is because it is a non-critical dimension, meaning that since there are small amounts of overlap between each indent roll error will not affect the registration of the features since the die can still be flat relative to the surface. The only time this would need to be addressed is if the roll angle was so large that gaps of unindented surface were left between registrations.

7.1.4 Experiment Electronics

During the indent experiments, the actuator is driven by the closed loop control system discussed in Section 6.1.6 using a Signal Recovery Model 7270 lock-in amplifier to automatically track the resonance of the actuator. The drive signal is split and one channel is phase-delayed to produce input signals to excite both of the actuator's operating modes. Two Trek piezo drivers amplify the drive signal as well as measure the current draw of the piezo elements necessary for the feedback system. A photonic sensor and oscilloscope are used to monitor the actuator's motion and set the machine axes and spindle speed to match the die speed. The photonic sensors cannot be used to measure the actuator tip while indenting is occurring; however, the resonance tracking system ensures the elliptical path is consistent throughout the duration of the indentation experiment. This electronics setup is shown in Figure 7-9.

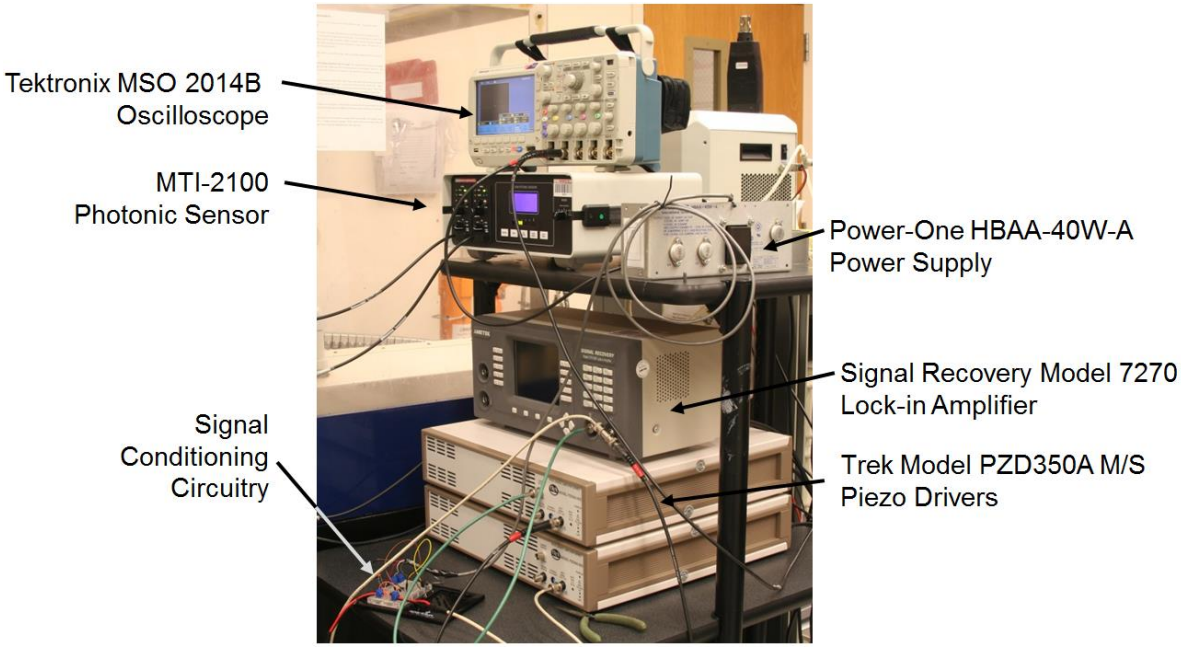


Figure 7-9. Electronics to drive and measure actuator.

7.2 INDENTING PROCEDURE

Performing the indents using either the ASG (Figure 7-2) or Nanoform (Figure 7-3) DTM require the same steps to create well-formed, uniform features. The procedure used for the indentations performed in this work is as follows:

1. Center a cylindrical workpiece one the axis of the DTM. Workpiece diameters of 1" or less were used due to space limitations in the chamber of the Scanning Electron Microscope (SEM).
2. Use a single-point diamond tool to produce an optical quality finish on the workpiece (Ra<10 nm). After the workpiece is cut, it is not removed for the duration of the experiment since the runout error must be less than 50 nm for even indents at every position on the workpiece.
3. Mount the actuator on the DTM axis and set the die height to the nominal spindle centerline height. Choose a reference surface on the side of the actuator mounting assembly and adjust it to be square to the axis of the DTM for a position reference.

4. Run the actuator using the closed-loop controller and adjust the phase setpoint to produce the desired elliptical path. Using the photonic sensor and oscilloscope, measure the dimensions of the ellipse and set the corresponding spindle speed to match the horizontal velocity of the die path (Equation (1-5)).
5. The actuator is moved into contact with the rotating workpiece until a visible line of indents is observed. The zero point is set at the surface of the part.
6. A part program is run that jogs the actuator into contact with the surface (based on the zero point set in step 5), and crossfed along the surface of the part to create an array of indents as shown in Figure 7-1.
7. After the indents are created, the part is removed from the spindle and a white light interferometer (Zygo NewView) is used to measure the slope of each indent, as shown in Figure 7-5.
8. Another part is placed back on the DTM spindle and steps 1 and 2 are repeated.
9. Using the slope measurements from step 7, the actuator can be adjusted to correct for pitch and yaw errors using the techniques discussed in Section 7.1.3.1.
10. After the actuator tilt is corrected, steps 4-6 are performed again to create indents around the periphery of the workpiece.

7.3 50 kHz Indent Results

Indents are performed on a 6061 aluminum workpiece using the 50 kHz actuator design. The experimental setup shown in Figure 7-2 on the ASG DTM was used. Table 7-1 outlines the indent parameters used in the experiment.

Table 7-1. Specifications for indent experiment with 50 kHz actuator.

Process Parameter		Value	Units
Indenting frequency	f	47.8	kHz
Length of die	L_{die}	19	mm
Width of die	w_{die}	19	mm
Horizontal ellipse dimension	a	0.00065	mm
Radius of drum	r_{drum}	12.2	mm
Spindle speed	ω_{speed}	152.8	rpm
Feedrate	fr	2.9	mm/min
Features/second	-	15.5	million

The procedure used is outline in Section 7.2 and the indented workpiece is imaged using an SEM to examine the final indent results. Images of a selected region of indents are shown in Figure 7-10 for magnifications of 400x, 1100x, 400x, and 13000x.

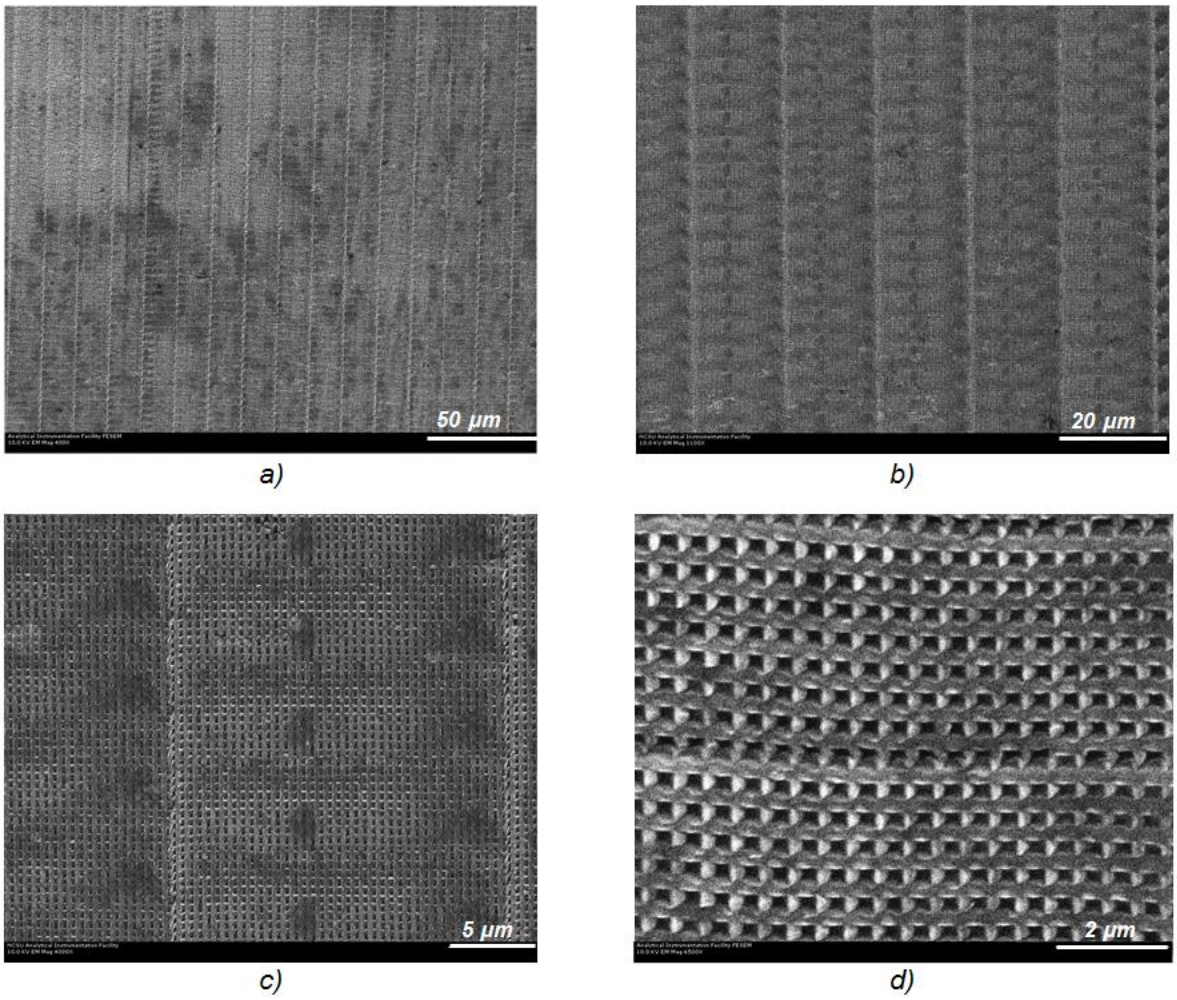


Figure 7-10. SEM images of indented aluminum workpiece at 400x (a), 1100x (b), 4000x (c), and 13000x (d).

The indents in Figure 7-10 were generated with the actuator crossfeeding from left to right, and the worksurface moving from bottom to top. Figure 7-10(d) clearly shows that the square post die features have indented into the surface while matching the speed of the workpiece. This is apparent by lack of elongated features that would occur if there was a large mismatch in velocity. Figure 7-10(c) shows that the tilt of the actuator was matched well since there is not discernable border marks at the top and bottom of each indent. There is a visible line vertically up the image suggesting that the edge of the indents overlapped; however, features are still visible so the effect of this on the functionality of the surface

should be negligible. The images in both Figure 7-10(b) and (c) show periodic regions of missing features, which is most likely due to material pickup. This occurs when material gets pressed into the die and sticks to the features causing a slight distortion of the surface. Previous work shows a correlation between material pick-up, feature shape, and worksurface material [15]. Finally, Figure 7-10(a) shows a large region of indents, were areas are varying between light and dark in appearance. This variation is not due to indent inconstancies, but rather the surface of the workpiece was not properly cleaned before indenting or imaging.

7.4 40 kHz Indents

The 40 kHz actuator design was used to perform indents on a brass workpiece. Two different featured dies are used in the following experiments where one has an array of square posts, and the other has an array of tapered posts. Both of the feature types are 500 nm tall with a 500 nm pitch. Depictions of the die profiles for the square and tapered post are shown in Figure 7-11(a) and (b), respectively.

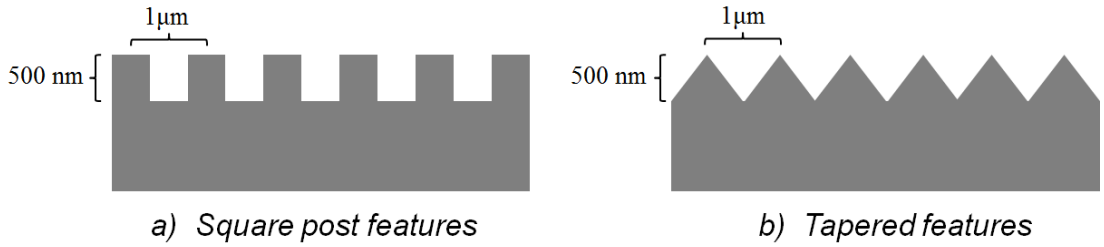


Figure 7-11. Sketch of the die feature shapes

7.4.1 Square Post Die

Indents are performed on a 360 brass workpiece using the 40 kHz actuator design. The experimental setup is shown in Figure 7-3 using the Nanoform DTM. Table 7-2 outlines the indent parameters used in the experiments.

Table 7-2. Indent parameters for square post die.

Process Parameter		Value	Units
Indenting Frequency	f	38.91	kHz
Length of die	L_{die}	0.020	mm
Width of die	W_{die}	0.020	mm
Horizontal ellipse dimension	a	0.0016	mm
Radius of drum	r	12.7	mm
Spindle speed	$\omega_{spindle}$	294.1	rpm
Feedrate	fr	5.8	mm/min
Features/second	-	31.1	million

The procedure used is outlined in Section 7.2 and the indented workpiece was imaged using an FESEM to examine the final indent results. Images of a selected region of indents are shown in Figure 7-12 for magnifications of 250x, 500x, 4000x, and 20000x.

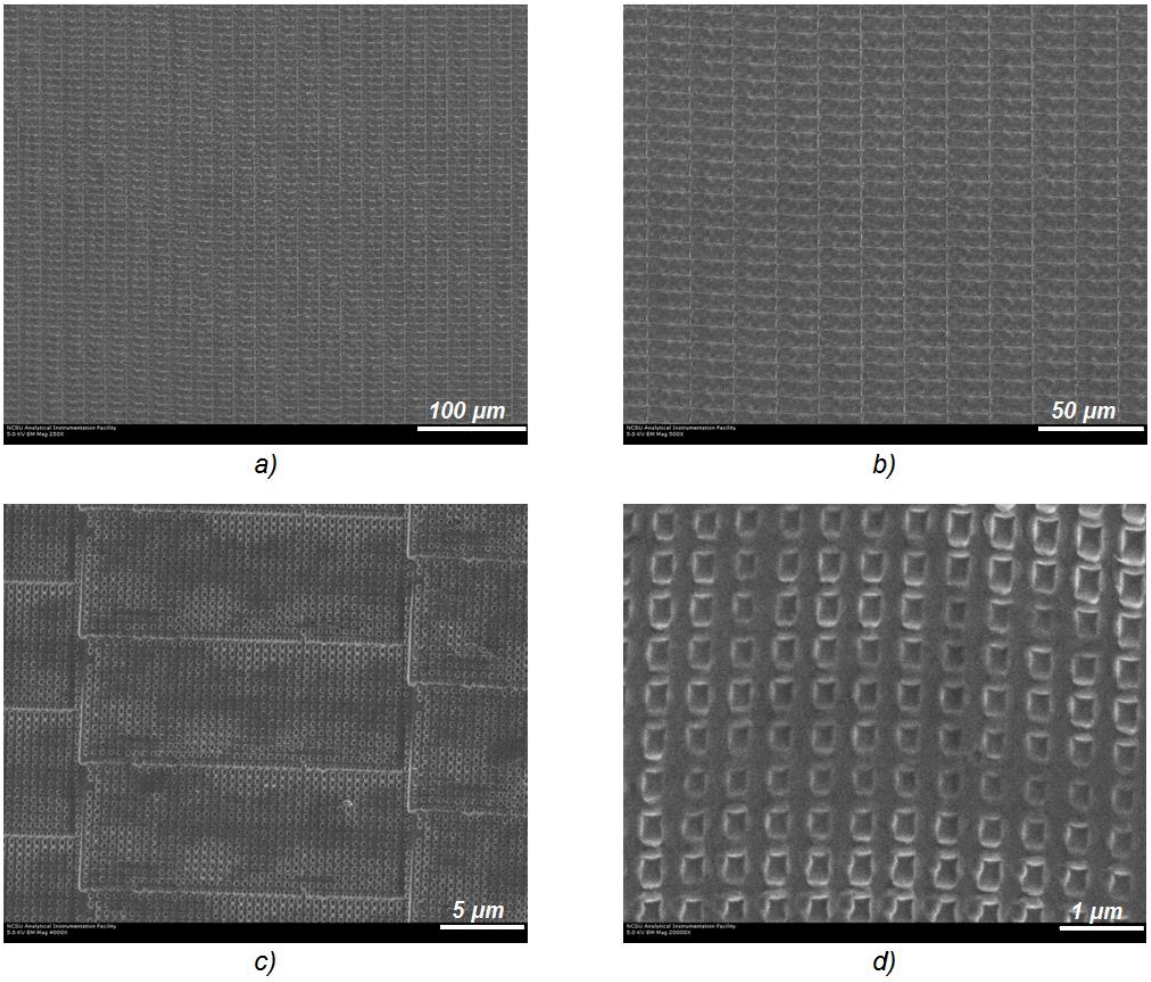


Figure 7-12. SEM images of indented brass workpiece at 250x (a), 500x (b), 4000x (c), and 20000x (d).

The indents in Figure 7-12 were generated with the worksurface moving from bottom to top, and the actuator crossfeeding from left to right. Each die impression appears rectangular (in spite of using a square die) due to the fact that overlapping was required to match the worksurface speed with the attainable transverse velocity of the actuator. As transverse amplitude of the actuator is increased, larger surface velocities can be used to match the speed.

The lower magnification images in Figure 7-12(a) and (b) demonstrate the ability to achieve uniform areas of indents using a resonant actuator and appropriate feedback controller. The

higher magnification images in Figure 7-12(c) and (d) clearly show the impressions left by the square-post die. The indents appear well-formed with minimal distortion, suggesting that the worksurface speed matched the velocity of the die at the time of contact. The square borders around each indent are artifacts of the die, where a slight tilt in the orientation of the die relative to the workpiece causes one side of the die to indent slightly deeper. This alignment can be fine-tuned in successive experiments by measuring the tilt of each indent and adjusting the actuator attitude appropriately.

7.4.2 Tapered Die

In efforts to reduce material pickup and increase the quality of the feature replications, a tapered-feature die was used. The indents are performed on the Nanoform DTM shown in Figure 7-3, using a 360 brass workpiece and the 40 kHz actuator design.

Table 7-3 outlines the indent parameters used in the experiments.

Table 7-3. Indent parameters for tapered die.

Process Parameter		Value	Units
Indenting Frequency	f	38.84	kHz
Length of die	L_{die}	0.020	mm
Width of die	w_{die}	0.020	mm
Horizontal ellipse dimension	a	0.0014	mm
Radius of drum	r	12.6	mm
Spindle speed	ω_{speed}	249.6	rpm
Feedrate	fr	4.8	mm/min
Features/second	-	27.2	million

The procedure used is outlined in Section 7.2 and the indented workpiece is imaged using an FESEM to examine the final indent results. Images of a selected region of indents are shown in Figure 7-13 for magnifications of 400x, 1000x, 6500x, and 20000x.

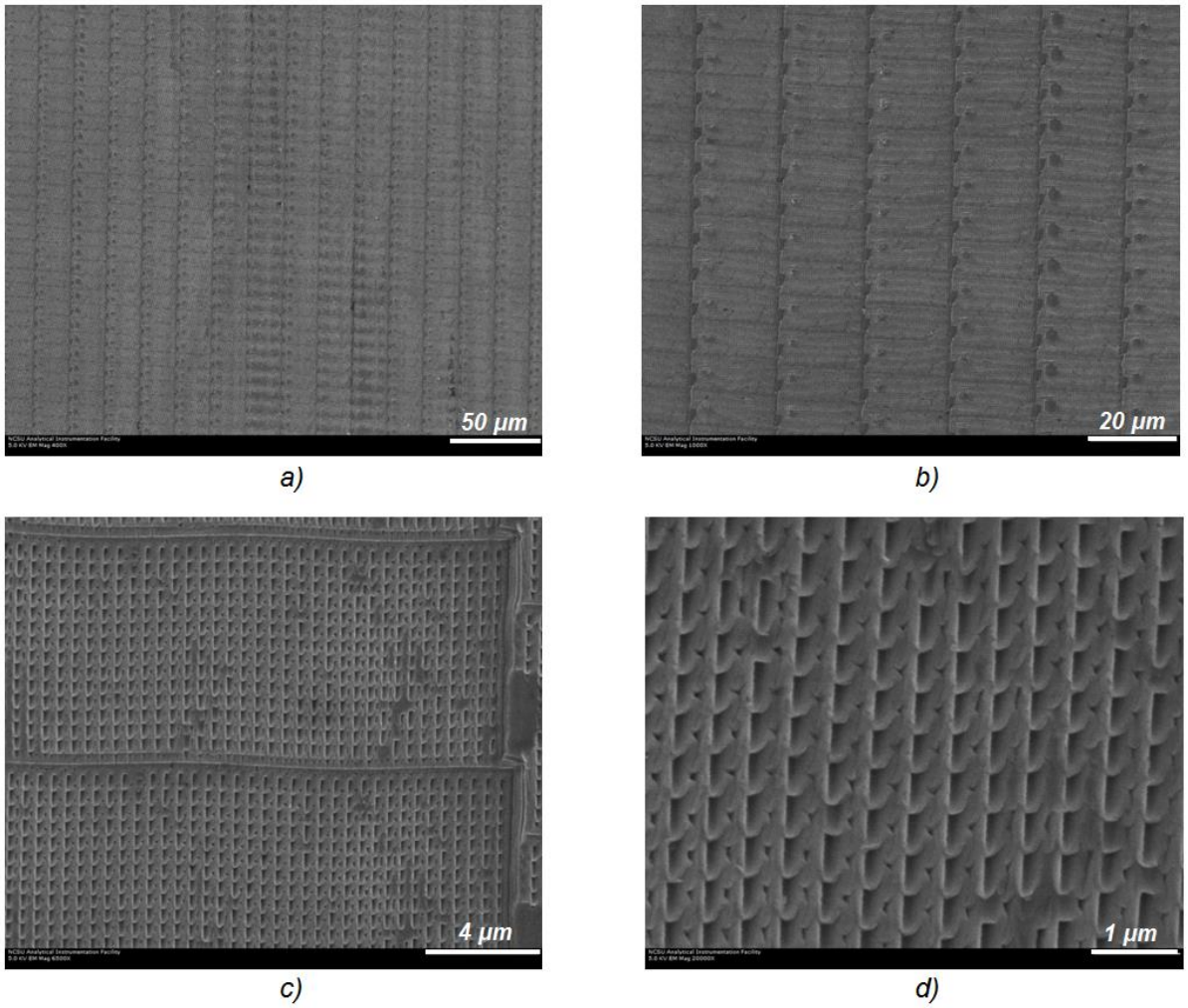


Figure 7-13. SEM images of brass workpiece at 400x (a), 1000x (b), 6500x (c), and 20000x (d).

Figure 7-13(a) and (b) demonstrate successful replication of features over several revolutions of the workpiece, where the indents appear uniform. Figure 7-13(b) shows small, repeated areas of no feature formation which is due to material pickup. The pickup with the tapered die, however, appears to be less than in previous experiments using the square post die. Figure 7-13(c) shows uniform features registration over the face of the die, but there appears to be border marks created when the die is pressed too far into the surface. This could be avoided in subsequent experiments by reducing the depth of indentation using the DTM axis.

Figure 7-13(d) show the features within one die indent, where there is slight distortion caused by a velocity mismatch between the die and workpiece. This suggests that the surface velocity was slightly mismatched with the horizontal die velocity, resulting in approximately 100 nm of distortion.

7.5 SUMMARY

The nanocoining actuator and controller were validated by performing several indent experiments using 1" diameter, cylindrical aluminum and brass workpieces. Since the die must impact the worksurface uniformly, alignment of the actuator is vital to producing well-formed indents. An alignment process to calibrate the pitch, roll, and yaw of the actuator relative to the machine axes was described. These tilt angles are adjusted based on test indents, where the die impression tilt is measured using a white light interferometer. Due to the geometry of indenting a cylindrical workpiece, the tilt of the actuator can be adjusted by moving it in the vertical direction because it allows finer angle resolution than other tilt-adjustment methods.

Indents were performed using two, 20x20 μm, 500 nm pitch diamond dies; one with square post features and the other with tapered features. Due to the previously investigated influence of feature shape and workpiece material on material pickup, it was determined that the brass workpiece and tapered die provided better fidelity than the aluminum workpiece and square post die. SEM imaging was used to examine the resulting indented workpiece, which demonstrated well-formed indents created at both 50 kHz and 40 kHz. Since the 40 kHz actuator is capable of larger horizontal amplitudes, areas could be indented quicker than with the 40 kHz actuator. Ultimately, the indent experiments demonstrated that a 2-D, resonant actuator could be used to replicate features and ultrasonic frequencies.

8 CONCLUSIONS AND FUTURE WORK

8.1 SUMMARY OF RESEARCH

8.1.1 Design of Elliptically-vibrating Actuator

Inspired by existing ultrasonic EVAM tools, an elliptically-vibrating, resonant actuator for nanocoining was designed where two orthogonal modes could be excited simultaneously. Triggering these modes out-of-phase resulted in elliptical paths at over 40 kHz. The body of the actuator was designed to focus the vibrational motion using an ultrasonic concentrator to amplify the motion and produce the desired path with several micrometers of displacement in both directions. A mount was designed to hold the actuator at node points to avoid restriction of the resonant motion, while being sufficiently stiff in the lateral direction to reject any disturbances that may occur when the die impacts the workpiece. A method of tuning the geometry to adjust the mode frequencies was also outlined. This design was shown to be capable of elliptical motion at ultrasonic frequencies.

8.1.2 Construction and Testing of Prototypes

After the geometry was designed using FEA simulations, the actuator body was CNC machined from a single piece of 6061 aluminum and PZT elements were fixed to opposing sides of the body. The relatively simple construction allowed for several actuator iterations to be built and tuned quickly, which was useful in refining the design based on physical measurements and models. The systematic method of designing and building the actuators proved successful in creating a mechanism for the of sub-micrometer features.

8.1.3 Resonance-tracking Controller

An automatic resonance tracking controller was designed and implemented to automatically update the drive frequency of the actuator based on resonant frequency shifts that occur in the actuator due to self-heating and environmental effects. The design was based on a classic PLL design, which involves measuring the phase between the input voltage and current draw

across the piezos. Electrically-equivalent impedance models of the system were used to demonstrate how this phase measurement corresponds to mechanical resonance. Two resonance-tracking systems were developed; one involved using discrete phase measuring and signal generating electronics, driven by an Arduino microcontroller and the second utilized an off-the-shelf lock-in amplifier that encloses a PLL loop to track the resonance. Both systems demonstrated the feasibility of using a PLL strategy to track the actuator resonance and maintain the desired elliptical path despite the time-varying resonant frequency of the actuator.

8.1.4 System Synchronization

To generate uniformly distributed indents over large areas, the actuator frequency, workpiece speed, and crossfeed rate must be synchronized. For example, when the actuator resonant frequency decreases the resonance tracking controller will lower the frequency of the drive signal. As the frequency is reduced, the workpiece speed must also decrease because the indents are being performed at a slower rate. This reduction in workpiece speed also requires a reduction in crossfeed rate. Since the synchronization of the process is driven by the actuator speed, a central controller is proposed that makes the necessary adjustments to the machine axes based on the instantaneous actuator frequency. Another system introduced dealt with compensating for runout errors in the cylindrical workpiece that would affect the indent depth of the features. This system used a low-speed, auxiliary actuator to follow the surface based on a noncontact sensor positioned slightly ahead of the indenter. This system was able to track a 1 μm runout error at around 10 Hz to within 50 nm, which is sufficient to produce uniform indentations.

8.1.5 Ultrasonic Indentation

Indents were performed at 40 kHz and 50 kHz to demonstrate the feasibility of nanocoining surfaces with ultrasonic devices. Both aluminum and brass workpieces were used to create arrays of submicrometer features at a rate of about 6 mm^2/s, or approximately 23.7 million 500 nm pitch features per second (1600 per die indent). An alignment procedure was

developed where test indents were used to calibrate the alignment of the actuator on the machine axes. The test indents were measured using a white light interferometer and the uniformity of the features over the area of the die improved by adjusting the tilt and yaw of the actuator. SEM imaging was used to examine the indents, and indents were uniformly created over the surface of the workpiece.

8.1.6 Conclusion

The use of an ultrasonic, elliptically-vibrating actuator for nanocoining sub-micrometer features has been proven successful. The design methodology of a bi-modal resonant device was demonstrated to be useful in producing prototypes that can generate the desired elliptical motion. It has also been shown that the use of a feedback control to track the resonance of the actuator can generate areas of uniform features despite the varying resonant frequency. Finally, SEM images of experimental indent results showed uniform features coverage after indenting at ultrasonic frequencies.

8.2 Recommendations for Future Work

8.2.1 Lower-order Modes for Actuator Design

Much of the actuator design methodology was centered on increasing the amplitude of the actuator to achieve the desired elliptical path. While an elliptical path was generated, the smaller-than-ideal amplitude resulted in overlapping indents which results in a slower area-rate of indents. A future design goal that may yield inherently larger amplitudes may be to use lower-order modes, such as the 4^{th} bending mode as opposed to the 5^{th} bending mode. This would require physically larger actuator footprint to increase the stiffness in the bending direction to utilize this lower order mode; however, previous designs in the literature suggest that this may yield more desirable amplitude characteristics [29].

8.2.2 Independent Longitudinal and Bending Piezo Elements

One issue that restricted the size of the ellipse that could be used for indenting was an issue of a tilted elliptical path, whereby the maximum horizontal velocity did not occur at the minimum vertical position (where indenting occurs). This is due to the longitudinal and bending modes not resonating at 90° out-of-phase. The current design relies on a coupled excitation method, where the two modes are excited using two opposing piezos. A design improvement may be to have two sets of piezo elements, such that two elements are excited in-phase to actuate the longitudinal mode and two elements are excited out-of-phase to actuate the bending mode. Improved control of the elliptical shape could then be found by varying the phase between the longitudinal piezo elements and the bending piezo elements.

8.2.3 Further Develop Touch-off Procedure

Performing the indents required that a touch-off procedure to find the surface of the workpiece and set the depth of indenting. Since this process involved visual inspection of the 'touch-off ring,' or a small indented area that appears when the actuator is slowly brought into contact with the worksurface, it was difficult to determine the actuator indent depth. A method of force feedback may be useful in calibrating the indent depth for future experiments. This force feedback may also be a useful feedback mechanism for a depth-tracking system, such as the one constructed for tracking part errors in Section 6.3.

8.2.4 Investigate Functionality of Nanocoined Features

While this work focused on a system to create sub-micrometer features, further research is needed to determine the functionality of large areas of indented nanofeatures. Issues such as measuring large areas (meters) of features that would not be feasible or possible to image in an SEM chamber, or the effect of residual features of indenting such as border marks would be of great interest prior to the commercialization of this technology.

REFERENCES

[1] G. Xie, G. Zhang, F. Lin, J. Zhang, Z. Liu and S. Mu. The fabrication of subwavelength anti-reflective nanostructures using a bio-template. Nanotechnology 19(9), pp. 095605. 2008.

[2] A. Yoshida, M. Motoyama, A. Kosaku and K. Miyamoto. Antireflective nanoprotuberance array in the transparent wing of a hawkmoth, cephonodes hylas. Zool. Sci. 14(5), pp. 737-741. 1997.

[3] K. Forberich, G. Dennler, M. C. Scharber, K. Hingerl, T. Fromherz and C. J. Brabec. Performance improvement of organic solar cells with moth eye anti-reflection coating. Thin Solid Films 516(20), pp. 7167-7170. 2008.

[4] A. R. Parker and H. E. Townley. Biomimetics of photonic nanostructures. Nature Nanotechnology 2(6), pp. 347-353. 2007.

[5] Jingyun Huang and Xudong Wang and Zhong,Lin Wang. Bio-inspired fabrication of antireflection nanostructures by replicating fly eyes. Nanotechnology 19(2), pp. 025602. 2008.

[6] A. Gombert, W. Glaubitt, K. Rose, J. Dreibholz, B. Bläsi, A. Heinzel, D. Sporn, W. Döll and V. Wittwer. Subwavelength-structured antireflective surfaces on glass. Thin Solid Films 351(1–2), pp. 73-78. 1999.

[7] S. Walheim, E. Schaffer, J. Mlynek and U. Steiner. Nanophase-separated polymer films as high-performance antireflection coatings. Science 283(5401), pp. 520-522. 1999.

[8] D. Stavenga, S. Foletti, G. Palasantzas and K. Arikawa. Light on the moth-eye corneal nipple array of butterflies. Proceedings of the Royal Society B: Biological Sciences 273(1587), pp. 661-667. 2006.

[9] J. Drelich, E. 'Chibowski, D. Meng and K. Terpilowski, "Hydrophilic and superhydrophilic surfaces and materials," Soft Matter, vol. 7, 2011.

[10] J. Bico, U. Thiele and D. Quéré. Wetting of textured surfaces. Colloids Surf. Physicochem. Eng. Aspects 206(1–3), pp. 41-46. 2002.

[11] M. Sun, C. Luo, L. Xu, H. Ji, Q. Ouyang, D. Yu and Y. Chen. Artificial lotus leaf by nanocasting. Langmuir 21(19), pp. 8978-8981. 2005.

[12] Q. Chen, G. Hubbard, P. A. Shields, C. Liu, D. W. E. Allsopp, W. N. Wang and S. Abbott. Broadband moth-eye antireflection coatings fabricated by low-cost nanoimprinting. Appl. Phys. Lett. 94(26), 2009.

[13] C. Sun, P. Jiang and B. Jiang. Broadband moth-eye antireflection coatings on silicon. Appl. Phys. Lett. 92(6), pp. 061112. 2008.

[14] K. Park, H. J. Choi, C. Chang, R. E. Cohen, G. H. McKinley and G. Barbastathis. Nanotextured silica surfaces with robust superhydrophobicity and omnidirectional broadband supertransmissivity. ACS Nano 6(5), pp. 3789-3799. 2012.

[15] E. Zdanowicz, T. Dow and R. Scattergood, "Rapid fabrication of nanostructured surfaces using nanocoining," Nanotechnology, vol. 23, 2012.

[16] A. Abdullah, M. Shahini and A. Pak. An approach to design a high power piezoelectric ultrasonic transducer. Journal of Electroceramics 22(4), pp. 369-382. 2009.

[17] K. Adachi, T. Takahashi and H. Hasegawa. Analysis of screw pitch effects on the performance of bolt-clamped langevin-type transducers. J. Acoust. Soc. Am. 116(3), pp. 1544-1548. 2004.

[18] G. A. Piezoelectric ceramics and ultrasonic transducers. 22(10), pp. 13. 1989.

[19] C. Doumanidis and Y. Gao. Mechanical modeling of ultrasonic welding. WELDING JOURNAL-NEW YORK- 83(4), pp. 140-S. 2004.

[20] H. Daniels. Ultrasonic welding. Ultrasonics 3(4), pp. 190-196. 1965.

[21] S. Matsuoka. Ultrasonic welding of ceramics/metals using inserts. J. Mater. Process. Technol. 75(1), pp. 259-265. 1998.

[22] M. Tolunay, P. Dawson and K. Wang. Heating and bonding mechanisms in ultrasonic welding of thermoplastics. Polymer Engineering & Science 23(13), pp. 726-733. 1983.

[23] E. de Vries. Mechanics and mechanisms of ultrasonic metal welding. 2004.

[24] M. Wiercigroch, R. Neilson and M. Player. Material removal rate prediction for ultrasonic drilling of hard materials using an impact oscillator approach. Physics Letters A 259(2), pp. 91-96. 1999.

[25] B. Azarhoushang and J. Akbari. Ultrasonic-assisted drilling of inconel 738-LC. Int. J. Mach. Tools Manuf. 47(7), pp. 1027-1033. 2007.

[26] Q. Zhang, C. Wu, J. Sun and Z. Jia. The mechanism of material removal in ultrasonic drilling of engineering ceramics. Proc. Inst. Mech. Eng. Pt. B: J. Eng. Manuf. 214(9), pp. 805-810. 2000.

[27] D. Brehl and T. Dow. Review of vibration-assisted machining. Precis Eng 32(3), pp. 153-172. 2008.

[28] M. Jin and M. Murakawa. Development of a practical ultrasonic vibration cutting tool system. J. Mater. Process. Technol. 113(1), pp. 342-347. 2001.

[29] E. Shamoto, N. Suzuki, E. Tsuchiya, Y. Hori, H. Inagaki and K. Yoshino. Development of 3 DOF ultrasonic vibration tool for elliptical vibration cutting of sculptured surfaces. CIRP Ann. Manuf. Technol. 54(1), pp. 321-324. 2005.

[30] T. Moriwaki and E. Shamoto. Ultrasonic elliptical vibration cutting. CIRP Annals-Manufacturing Technology 44(1), pp. 31-34. 1995.

[31] E. Shamoto, N. Suzuki, T. Moriwaki and Y. Naoi. Development of ultrasonic elliptical vibration controller for elliptical vibration cutting. CIRP Annals-Manufacturing Technology 51(1), pp. 327-330. 2002.

[32] X. Li and D. Zhang. Ultrasonic elliptical vibration transducer driven by single actuator and its application in precision cutting. J. Mater. Process. Technol. 180(1), pp. 91-95. 2006.

[33] J. Tsujino, T. Ueoka and T. Sano. Welding characteristics of 27 kHz and 40 kHz complex vibration ultrasonic metal welding systems. Presented at Ultrasonics Symposium, 1999. Proceedings. 1999 IEEE. 1999,

[34] J. Tsujino, T. Ueoka, T. Kashino and F. Sugahara. Transverse and torsional complex vibration systems for ultrasonic seam welding of metal plates. Ultrasonics 38(1), pp. 67-71. 2000.

[35] J. Tsujino, H. Yoshihara, T. Sano and S. Ihara. High-frequency ultrasonic wire bonding systems. Ultrasonics 38(1), pp. 77-80. 2000.

[36] M. Bauer and T. Dow. Design of a Linear High Precision Ultrasonic Piezoelectric Motor 2001.

[37] M. Bauer. Design of a linear high-precision ultrasonic piezoelectric motor. ProQuest Dissertations and Theses 2001.

[38] S. Roa, Mechanicals Vibrations. New Jersey: Pearson Prentice Hall, 2004.

[39] S. Sherrit, B. P. Dolgin, Y. Bar-Cohen, D. Pal, J. Kroh and T. Peterson. Modeling of horns for sonic/ultrasonic applications. Presented at Ultrasonics Symposium, 1999. Proceedings. 1999 IEEE. 1999.

[40] F. Wang, X. Zhao, D. Zhang and Y. Wu. Development of novel ultrasonic transducers for microelectronics packaging. J. Mater. Process. Technol. 209(3), pp. 1291-1301. 2009.

[41] S. Sherrit, S. Askins, M. Gradziol, B. Dolgin, X. Bao, Z. Chang and Y. Bar-Cohen. Novel horn designs for ultrasonic/sonic cleaning welding, soldering, cutting and drilling. Presented at Proceedings of SPIE. 2002.

[42] L. Beranek and T. Mellow. Acoustics: Sound Fields and Transducers 2012.

[43] J. Pons, P. Ochoa, M. Villegas, J. Fernández, E. Rocon and J. Moreno. Self-tuned driving of piezoelectric actuators: The case of ultrasonic motors. Journal of the European Ceramic Society 27(13–15), pp. 4163-4167. 2007.

[44] B. Mortimer, T. du Bruyn, J. Davies and J. Tapson. High power resonant tracking amplifier using admittance locking. Ultrasonics 39(4), pp. 257-261. 2001.

[45] R. Martin and R. Sigelmann, "force and electrical Thevenin equivalent circuits and simulations for thickness mode piezoelectric transducers," J. Acoust. Soc. Am., vol. 58, 1975.

[46] Y. Mizutani, T. Suzuki, I. Hiroaki and H. Yoshida, "Power Maximizing of Ultrasonic Transducer Driven by MOS-FET Inverter Operating at 1 MHz," vol. 6, 1996.

[47] A. Ramos-Fernandez, J. Gallego-Juarez and F. Montoya-Vitini. Automatic system for dynamic control of resonance in high power and high Q ultrasonic transducers. Ultrasonics 23(4), pp. 151-156. 1985.

[48] H. Motegi, H. Muroga and S. Suzuki. VCO Controlled by Separate Phase Locked Loop 1990.

[49] R. Suzuki. "Phase-Locked Loop Circuit" U.S. Patent 6 094 078, July 25, 2000.

[50] W. Chang, K. Ma and K. Yarn, "New frequency-tracking control method for ultrasound welding system by the FPGA Chip," International Journal of the Physical Sciences, vol. 5, pp. 2014-2019, 27 September, 2012, 2010.

[51] H. Dong, J. Wu, G. Zhang and H. Wu. An improved phase-locked loop method for automatic resonance frequency tracing based on static capacitance broadband compensation for a high-power ultrasonic transducer. Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions On 59(2), pp. 205-210. 2012.

[52] Y. Maruyama, M. Takasaki and T. Mizuno, "Resonance frequency tracing system for langevin type ultrasonic transducers," in Mechatronic Systems Simulation Modeling and Control, D. Di Paola, M. Annalisa and G. Cicirelli, Eds. InTech, 2010.

[53] J. H. Scofield. Frequency-domain description of a lock-in amplifier. American Journal of Physics 62(2), pp. 129-132. 1994.

[54] Piezo Tutorial: Piezoelectrics in Nanopositioning, Designing with Piezoelectric Actuators," 2012.

[55] E. H. Jordan and M. R. Urban. An approximate analytical expression for elastic stresses in flat punch problems. Wear 236(1), pp. 134-143. 1999.

[56] K. L. Johnson and K. K. L. Johnson. Contact Mechanics 1987.

[57] J. Jiang, G. Sinclair and W. Meng. Quasi-static normal indentation of an elasto-plastic substrate by a periodic array of elastic strip punches. Int. J. Solids Structures 46(20), pp. 3677-3693. 2009.

[58] M. Mehregany and S. Roy. Introduction to MEMS. Microengineering Aerospace Systems, 1999.

[59] M. Kline. Calculus: An Intuitive and Physical Approach 1998.

APPENDICES

Appendix A INDENT MECHANICS

Jordan and Urban examine fretting fatigue [55], where two bodies are in contact and the relative motion between them is not small enough that the contact effect is not dominated by wear. Considering a classical flat-on-flat punch problem, a closed-form approximation to evaluate surface tractions and stress is formulated for the plane strain case. Results show a maximum error of less than 4% for the included material pairs (punch/substrate) when the contact body is five times as thick as the half width. Errors increased as contact body thickness decreased. The model used in this formulation is given in Figure A-1.

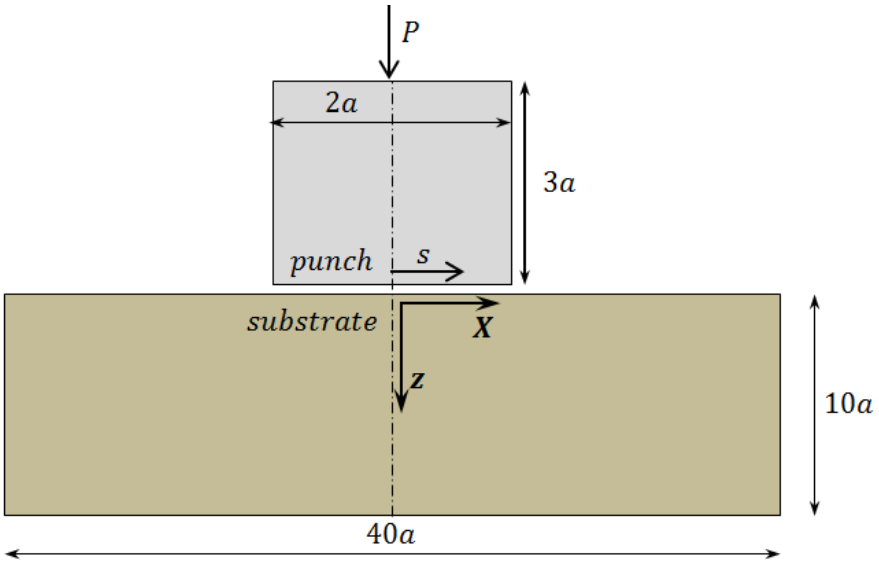


Figure A-1. 2D flat punch model and assumed dimensions.

The significance of this paper lies, in part, in the formulation of the tractive force expression. For a rigid, flat punch considered in 2D half space, the classic analytical solution for the normal tractive force per unit length *p(s)* is given as [56]:

$$p(s) = P \Big/ \left(\pi \left(a^2 - s^2 \right)^{1/2} \right) \tag{A-1}$$

Where a is the half-width of the punch, s is the position along the punch half-width and P is the applied normal force. It is clear that near the edges of the punch, the solution approaches a singularity characterized by an exponent of ½. This also implies that no matter how small the load P, plastic deformation will occur close to the edges of the punch due to the exponentially increasing tractive force. However since the punch will never be infinitely rigid, this paper proposes an alternative tractive force expression where the elastic singularity will actually be approached by an exponent $0 < \lambda < 0.5$. This equation is given as:

$$p(s) = P * M(\lambda, a) \Big/ \left(a^2 - s^2\right)^{\lambda} \tag{A-2}$$

where

$$\tan\{(1-\lambda)\pi\}\sin\{(1-\lambda)\pi\} + e\left[1 - \cos\{(1-\lambda)\pi\} - 2(1-\lambda)^2\right] = 0 \tag{A-3}$$

$$M(a,\lambda) \cong \frac{3.8^{\lambda(2\lambda-1)}}{2} \frac{a^{(2\lambda-1)}(1-2\lambda)}{\sin\left[(1-2\lambda)\frac{\pi}{2}\right]} \tag{A-4}$$

M is chosen to satisfy the force balance, and λ is a function of a ratio of the punch modulus over the substrate modulus ($e = E_p/E_s$). Previous work used the weighted sum of the two bounding conditions (constant tractive effort and infinitely rigid punch) which were then used to estimate the tractive force, along with a parameter derived from parametric finite element studies to incorporate the punch and half space elastic constants. Using the known surface equations from Equation (A-2), stresses in the half space can be found using a superposition integral and are given in Equation (A-5).

$$\sigma_x(X,z) = -\frac{-2z}{\pi}\int_{-a}^{a}\frac{p(s)(X-s)^2}{\left\{(X-s)^2+z^2\right\}^2}ds$$

$$\sigma_z(X,z) = -\frac{-2z^3}{\pi}\int_{-a}^{a}\frac{p(s)}{\left\{(X-s)^2+z^2\right\}^2}ds \qquad \text{(A-5)}$$

$$\sigma_{zx}(X,z) = -\frac{-2z^2}{\pi}\int_{-a}^{a}\frac{p(s)(X-s)}{\left\{(X-s)^2+z^2\right\}^2}ds$$

where *X* is the position in the half plane along the free surface, *s* is the location on the punch in the *X* direction, and *z* is the position into the substrate where *z=0* is the substrate free surface (shown in Figure A-1). The analytical solutions are verified using comprehensive FEA modeling. Closed-form solutions previously only existed for infinitely rigid punches and uniform pressure situations. When considering a non-rigid punch, this work is proposed as an alternative to lengthy finite element analyses or computationally intensive numerical solutions to find tractive forces and calculate crack growth.

Appendix A.1 Applications to Nanocoining

This application to crack growth calculations is a natural extension of the analytical results obtained from the previously mentioned formulation to estimate stress intensity factors, as applied to the phenomenon of fretting. The authors stated that by using the integral form of the tractive forces (Equation (A-2)) with the included mode I stress intensity factor expressions for a crack normal to the contact surface, the solution can be obtained quickly using a PC. The results of this method yielded reasonable predictions compared to existing fatigue life data.

Since nanocoining involves transferring features via deformation from a die to a substrate, understanding the resulting stresses is vital in designing the process. Nanocoining using a diamond (E = 1220 GPa) indenter has been performed on a variety of materials including 6061 aluminum, brass, copper, electroless nickel, silicon and germanium which have moduli

ranging from 70 GPa (aluminum) to around 160 GPa (silicon). The closed-form approximation proposed by Jordan and Urban that take into account the punch and substrate moduli would be useful in predicting the material effects of various substrates. Namely, the results suggest a method of predicting with reasonable accuracy the elastic singularity that occurs at the edges of the punch and the resulting stresses. To extend this method to indenting a surface using a diamond die, Equation (A-3) was first used to estimate the exponent λ based on the die and substrate moduli.

Table A-1. Material properties and numerical results for a = 10 μm and P = 1 N.

	Material	Modulus (Es)	e=Ep/Es	λ
Punch	Diamond	1660 GPa	--	--
Substrate	Aluminum	70 GPa	23.714	0.4750
	Brass	100 GPa	16.600	0.4647
	Silicon	160 GPa	10.375	0.4460
		Infinitely Rigid Punch		0.5000

Shown in Table A-1, three substrate materials were considered including aluminum, brass and silicon. Since the calculation of λ is a function of the ratio of the punch modulus to the substrate modulus, there is little variation in the λ parameter between the three substrate materials. This estimation suggests that since the diamond punch has an orders-of-magnitude larger modulus, the range of indenting substrates have little impact on the force and stress distribution of the indenting process. Using the results from Table A-1, Equation (A-2) and Equation (A-4), the force distributions are calculated.

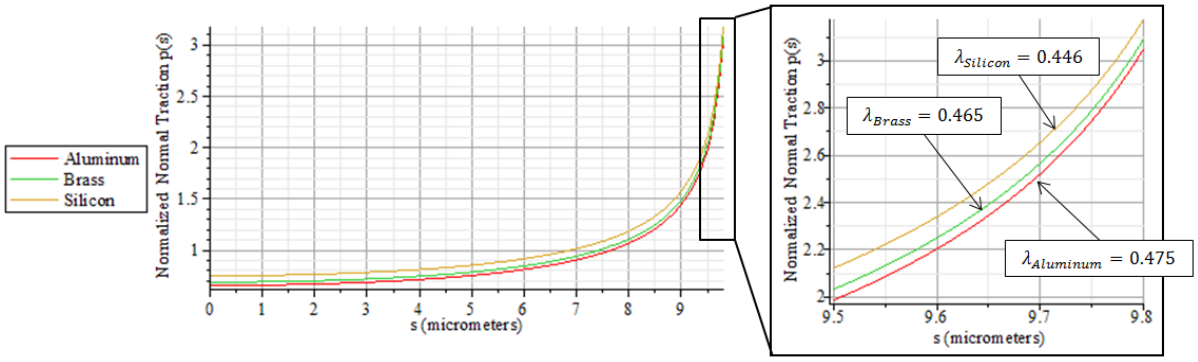


Figure A-2. Tractive force along half-length of die for aluminum, brass and silicon.

Figure A-2 demonstrates the normal tractive force approaching infinity at the edge of the die (a=10 μm). The substrate material dictates the exponential behavior at which the tractive forces approach a singularity. Since the diamond die is orders-of-magnitude larger than the substrate materials, there is relatively little variation in the tractive distributions between substrates. This is also reflected in the stress calculations for σ_x, σ_z, and σ_{zx} using Equation (A-5) in

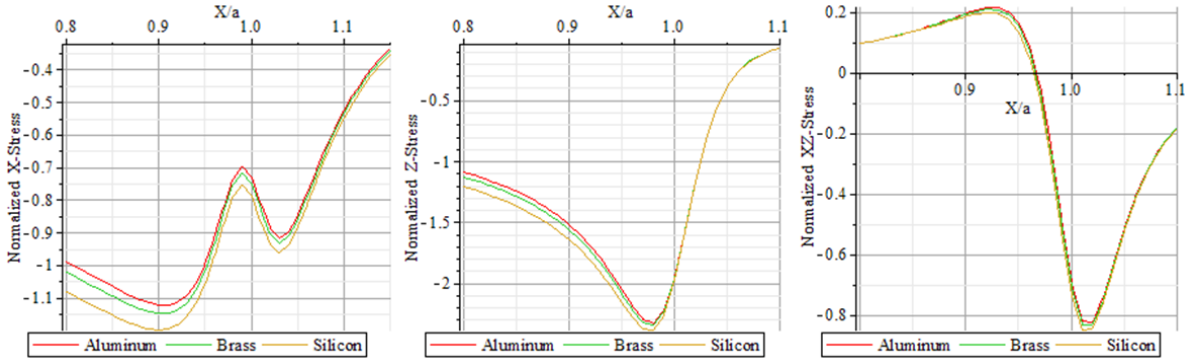


Figure A-3. Stress calculations for a diamond punch with a = 9.99 μm, X = 8..11.5μm and z = 500 nm.

The integral over ±a in Equation (A-5) is performed to ±9.9999 μm since the results approach a singularity at a = 10μm. The results from free-surface position X=8 to 11.5 μm since this is the highest area of stress and of most interest for this case. The stress

distribution is also a function of depth, z, which was chosen to be 500 nm for these results. Several depths are considered in the yielding analysis.

Appendix A.2 Discussion of Results

The stress distributions over the region in contact with the die are all approximately the same due to the large difference between the diamond die modulus and the substrate modulus. Based on the proposed contact model, the stresses in the substrate due to contact with the punch are a function of substrate position *X* and depth *z*. It can be observed from the given stress integrals in Equation (A-5) that the *x, z* and *zx* stresses will approach infinity at the free surface (*z=0*) when *s=a* (edge of the die). This is consistent with the presence of a force singularity in the solution at the edge of the punch. The stress results given in Figure A-3 appear to vary little between the aluminum, brass and silicon substrate for a constant depth, z=500nm. While the stress distributions are similar, the material response to the distributions may vary significantly. The following section will use the Von Mises yield criterion to address deformation behavior based on the stress profiles in Figure A-3 and material properties.

Appendix A.3 Yielding

Considering the deformation response of a substrate due to an indenter is typically done using numerical (primarily FEA) analysis. A significant amount of work has been done considering indenter geometries such as spherical, conical and pyramidal. Jiang states that there are relatively few investigations for plane-strain indenters because there are no companion hardness tests [57]. To address the issue of deformation for this case, the Von Mises yield criterion will be considered.

$$\sigma_{eff} = \frac{1}{\sqrt{2}}\left[\left(\sigma_x - \sigma_y\right)^2 + \left(\sigma_x - \sigma_z\right)^2 + \left(\sigma_y - \sigma_z\right)^2 + 6\left(\sigma_{xy}^2 + \sigma_{xz}^2 + \sigma_{yz}^2\right)\right]^{1/2} \qquad \text{(A-6)}$$

$$\sigma_{eff} \geq \sigma_0 \rightarrow \textit{yielding}$$

where σ_0 is the yield stress of the material. Since the plane strain assumption is used, σ_y, σ_{xy}, and σ_{yz} are considered to be zero. Since according to the model stress increases closer to the free surface (z=0), the normalized *x*, *z* and *zx* stress distributions are plotted for multiple depths. Figure A-4 illustrates the predicted stressed based on Equation (A-5) .

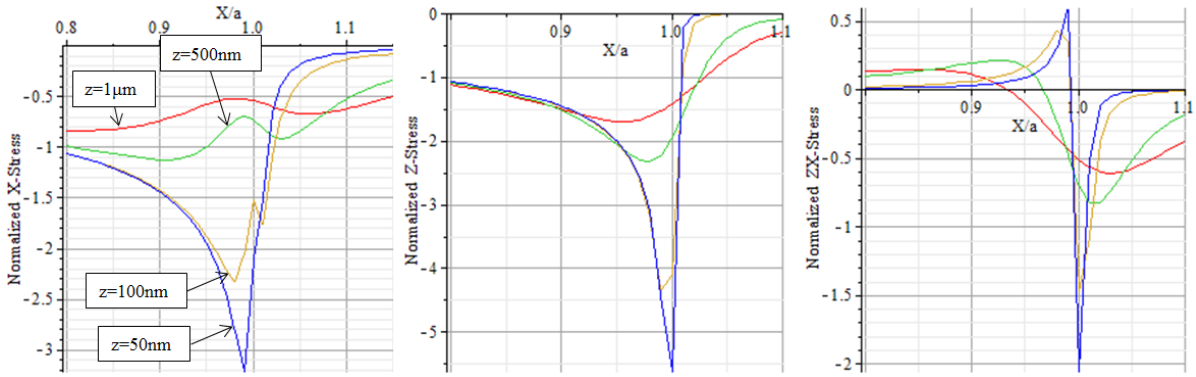


Figure A-4. Stresses in aluminum substrate at various depths where $\sigma_{nominal}$=50 MPa.

Using the stress results at various depths, yielding can be examined for aluminum, brass and silicon using the Von Mises yield criterion (Equation (A-6)). The maximum stress for *x*, *z*, and *zx* were used in the calculation of the yield criterion. The depth at which yielding occurs was determined iteratively for each material. An example calculation of the Von Mises stress criterion based on the plotted results is given below for aluminum where σ_{nom} is assumed to be 50 MPa and z = 80 nm.

$$\sigma_{eff} = \frac{1}{\sqrt{2}}\left[\left(2.5\sigma_{nom}\right)^2 + \left(2.5\sigma_{nom} - 4.5\sigma_{nom}\right)^2 + \left(-4.5\sigma_{nom}\right)^2 + 6\left(1.65\sigma_{nom}\right)^2\right]^{1/2} \quad \text{(A-7)}$$

$$\sigma_{eff} = 241.95MPa > 241MPa$$

This result shows that at the point of maximum stress (i.e. the edges of the die) the aluminum substrate will yield from 80nm depth to the surface leaving border marks. For the case of brass, the yield region will be smaller due to its high modulus. Likewise, the silicon with a significantly higher yield stress, will only yield at the surface of the substrate. It should be

noted that since silicon is so brittle, its ultimate strength is often considered to be equivalent to its yield strength [58]. A summary of the results are given in Table A-2.

Table A-2. Calculated yield zones based on Von Mises criterion.

	Material	Yield Stress	Depth where yielding initiates
Substrate	Aluminum	241 MPa	z = 80 nm
	Brass	310 MPa	z = 50 nm
	Silicon	1 Gpa*	z = 2.5 nm

*approximated based on ultimate strength

A sketch of the results given Table A-2 is given below where the approximate yield stress fields are super imposed on a schematic of the punch half-space. This illustration demonstrates the responses due to the different materials for the given stress distribution (Figure A-5).

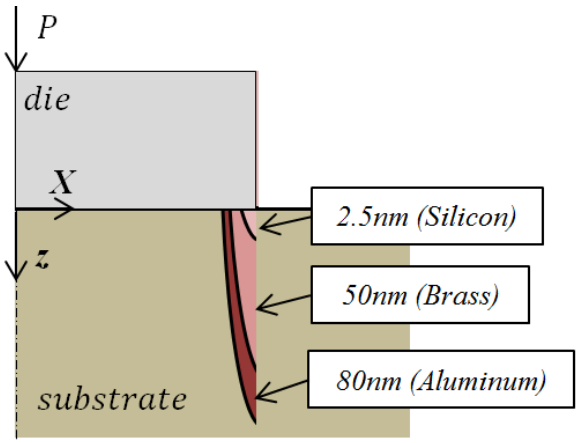


Figure A-5. Sketch of the yield zones based on the Von Mises criterion.

Appendix A.4 Conclusions

Due to the large difference in moduli between the diamond punch and the aluminum, brass, and silicon substrates, the estimated tractive forces and stress distributions across the area of the indenter where similar for all three materials. However, the onset of yield was drastically different due to the various material properties of the substrate. The Von Mises yield

criterion was used to estimate the onset of yield in each material based on the calculated stress profiles generated from Equation (A-5). Since the stress distributions in the substrate are a function of depth (z-direction) the Von Mises yield criterion was used to find at what point yield is present in the substrate. The aluminum was subject to the largest yield region for a given nominal stress. In contrast, the silicon experienced a relatively small yield zone due to its high modulus. The results suggest that the softer materials such as aluminum will be more susceptible to border marks during indentation. The harder materials such as silicon may be more resilient to boarder marks, however, it may be more susceptible to cracking due to its brittleness. This work may be useful in minimizing the effects of border marks and increasing the quality of the indentation in nanocoining.

Appendix B SMEAR CALCULATIONS

```
%%% calculate smear for varying indent depths
%%% and ellipse aspect ratios

%%% John Nowak

clear all
clc
clf

f = 40e3;               % actuator frequency, Hz
a = 3.18e-6;            % horizontal amplitude, m
phs = pi/2;             % ellipise phase, rad
w = f*2*pi;             % actuator freq, rad
per = 1/f;              % indent period
vdrum = a*w;            % workpiece surface velocity


b = (.5:0.1:12)*10^-6;
indep = (100:100:700)*10^-9;
t = 0:per/1000:per;

for i = 1:length(b);
    for j = 1:length(indep)
    x = a*sin(w*t);
    y = b(i)*sin(w*t+phs);
    vx = a*w*cos(w*t);
    surface = min(y)+indep(j);
    surfplot = surface(ones(length(t)),1);
    contact = find(y<surface);
    ycon = y(contact);
    vcon = vx(contact)';
    tcon = t(contact);
    surfcon = -vdrum*ones(length(contact),1);
    vsmear = vdrum - abs(vcon);
    xmin = min(tcon);  xmax = max(tcon);
    relcon = surfcon - vcon;
    smear(i,j) = abs(trapz(tcon,relcon))*10^9;  %nm
    end
end

for k = 1:size(smear,2)
    smear(:,k) = smooth(smear(:,k));
end

ar = b./a;
plot(ar,smear),axis([0 3.5 0 250]),xlabel('b/a'),ylabel('nm of
smear'),grid,...
    legend('100 nm','200 nm','300 nm','400 nm','500 nm','600 nm','700
nm');
```

Appendix C ANALYTICAL FREE-FREE BEAM SOLUTIONS

```
%%% John Nowak
%%% Calculate free-free beam vibration modes
%%% for varying lengths

clear all
clc

% % % Properties of Aluminum % % %
E = 70e9;    % Young's [Pa]
p = 2700;    % density [kg/m3]

c = sqrt(E/p);

nL = 2;      % Longitudinal mode number
nT = 6;      % Transvers mode number

b = 0.013;
h = 0.016;
A = b*h;
I = (h*h*h*b)/12;

BnL = 3.137*nT-1.583

fn=30000:1000:50000;
for i = 1:1:length(fn);
Ll(i) = (nL*pi*c)./(fn(i));
Lb(i) = (((BnL^4)*E*I)/(p*A*(2*pi*fn(i))*(2*pi*fn(i))))^(1/4);
end
for i = 1:1:length(fn);
Ll(i) = ((2*nL+1)*c)/(4*fn(i));
Lb(i) = (((BnL^4)*E*I)/(p*A*(2*pi*fn(i))*(2*pi*fn(i))))^(1/4);
end



Ll_plot = Ll*10^3;
Lb_plot = Lb*10^3;

plot(Ll_plot,fn/1000,'k',Lb_plot,fn/1000,'r--'),...
    xlabel('mm'),ylabel('kHz'),...
    legend('Longitudinal','Transverse');
repPlotArial;
```

Appendix D ANSYS CODE

Appendix D.1 Actuator Model

```
/filnam,FF2Dbeam
/prep7

fac = 1.2

!actuator body dimensions
!h = 0.01608*fac+0.001
h = 0.01608*fac+0.002
w = 0.01262
l = 0.11022*fac


!PZT element dimensions
tpzt = 0.00312
!lpzt = 0.02098*fac
lpzt = 0.0238
wpzt = w

!distance back edge of piezo is from back end of actuator
dend = 0.004*fac


!Front end Step dimensions
ht1 = 0.0102*fac
lt1 = 0.0168*fac
ht2 = 0.00416*fac
lt2 = 0.017*fac
w2 = 0.0065

!location of mounting set screws
ls1 = 0.0251*fac-.002
ls2 = 0.0724*fac

!radius of mounting set screw
rss = 0.004826/2
dss = w/2+0.00194

et,1,SOLID226
KEYOPT,1,1,1001
et,2,solid185
et,3,solid185

! compliance matrix
TB,ANEL,1,1,,0
TBDATA, 1, 13.327e10 , 5.988e10 , 5.988e10
TBDATA, 7, 13.328e10 , 5.988e10
TBDATA, 12, 10.267e10
```

```
TBDATA, 16, 3.670e10
TBDATA, 19, 3.0988e10
TBDATA, 21, 3.0988e10

! Permittivity
EMUNIT, EPZRO, 8.85E-12
MP, PERX, 1 , 1000
MP, PERY, 1 , 1000
MP, PERZ, 1 , 1100
! Density
MP, DENS, 1 , 7700


/com Piezo matrix
TB,PIEZ,1
TBDATA, 3, -6.1424
TBDATA, 6, -6.1424
TBDATA, 9, 13.3625
TBDATA, 14, 11.74
TBDATA, 16, 9.9163

!! material props for 6061
mp,ex,2,68.9e9
mp,dens,2,2700
mp,prxy,2,0.33

!! material props for steel
mp,ex,3,210e9
mp,dens,3,7850
mp,prxy,3,0.3


!!! keypoints !!!

k,1,0,w/2,h/2
k,2,0,-w/2,h/2
k,3,0,-w/2,-h/2
k,4,0,w/2,-h/2
k,5,0,0,0
k,6,l-lt1-lt2,0,0

k,7,dend,wpzt/2,h/2
k,8,dend,-wpzt/2,h/2
k,9,dend,-wpzt/2,h/2+tpzt
k,10,dend,wpzt/2,h/2+tpzt
k,11,dend,0,h/2
k,12,dend+lpzt,0,h/2

k,13,dend,wpzt/2,-h/2
k,14,dend,-wpzt/2,-h/2
k,15,dend,-wpzt/2,-h/2-tpzt
k,16,dend,wpzt/2,-h/2-tpzt
k,17,dend,0,-h/2
```

```
k,18,dend+lpzt,0,-h/2

k,19,l-lt1-lt2,w/2,ht1/2
k,20,l-lt1-lt2,-w/2,ht1/2
k,21,l-lt1-lt2,-w/2,-ht1/2
k,22,l-lt1-lt2,w/2,-ht1/2
k,23,l-lt1-lt2,0,0
k,24,l-lt1,0,0

k,25,l-lt1,w2/2,ht2/2
k,26,l-lt1,-w2/2,ht2/2
k,27,l-lt1,-w2/2,-ht2/2
k,28,l-lt1,w2/2,-ht2/2
k,29,l-lt1,0,0
k,30,l,0,0

k,31,ls1,w/2,0
k,32,ls1+rss,w/2,0
k,33,ls1,dss,0

k,34,ls2,w/2,0
k,35,ls2+rss,w/2,0
k,36,ls2,dss,0

k,37,ls1,-w/2,0
k,38,ls1+rss,-w/2,0
k,39,ls1,-dss,0

k,40,ls2,-w/2,0
k,41,ls2+rss,-w/2,0
k,42,ls2,-dss,0

l,5,6
l,11,12
l,17,18
l,23,24
l,29,30

a,1,2,3,4
a,7,8,9,10
a,13,14,15,16
a,19,20,21,22
a,25,26,27,28

lesize,1,,,20
lesize,2,,,5
lesize,3,,,5
lesize,4,,,10
lesize,5,,,10
lesize,6,,,10
lesize,7,,,10

circ,31,,33,32
```

```
circ,34,,36,35
circ,37,,39,38
circ,40,,42,41

l,31,33
l,34,36
l,37,39
l,40,42

al,26,27,28,29
al,30,31,32,33
al,34,35,36,37
al,38,39,40,41


vdrag,1,,,,,,1
vdrag,2,,,,,,2
vdrag,3,,,,,,3
vdrag,4,,,,,,4
vdrag,5,,,,,,5
vdrag,6,,,,,,42
vdrag,7,,,,,,43
vdrag,8,,,,,,44
vdrag,9,,,,,,45


vglue,all
!!!!!!!!!!!!  set mp's and mesh  !!!!!!!!!!!!!
allsel,all

vsel,s,volu,6,9
vatt,3,,3
allsel,all

vsel,s,volu,,2,3
vatt,1,,1
allsel,all

vsel,s,volu,,5
vatt,2,,2
allsel,all

vsel,s,volu,,10,11
vatt,2,,2
allsel,all

MSMID,1
MSHAPE,1,3D
!MSHKEY,1
vmesh,all

finish
```

Appendix D.2 Modal Analysis Solution

```
/SOLUTION          !ENTERS ANSYS SOLUTION PHASE
ANTYPE,2
MODOPT,LANB,20,35e3,,OFF,,

asel,s,area,,39
nsla,s,1
d,all,ux
d,all,uy
d,all,uz
allsel,all

asel,s,area,,44
nsla,s,1
d,all,ux
d,all,uy
d,all,uz
allsel,all

asel,s,area,,49
nsla,s,1
d,all,ux
d,all,uy
d,all,uz
allsel,all

asel,s,area,,54
nsla,s,1
d,all,ux
d,all,uy
d,all,uz
allsel,all

SOLVE
FINI
```

Appendix D.3 Longitudinal harmonic analysis

```
/SOLU              !ENTER SOLUTION PROCESSOR
SOLCONTROL,0             !TURN OFF SOLUTION CONTROLS
ANTYPE,HARMIC            !SPECIFY ANALYSIS TYPE AS MODAL
HROPT,FULL                 ! Full harmonic response
!NSUBST,10                 ! 30 Intervals within freq. range
HARFRQ, 41381


!HARFRQ,39570.3
!harfrq,37000,40000

asel,s,area,,39
```

```
nsla,s,1
d,all,ux
d,all,uy
d,all,uz
allsel,all

asel,s,area,,44
nsla,s,1
d,all,ux
d,all,uy
d,all,uz
allsel,all

asel,s,area,,49
nsla,s,1
d,all,ux
d,all,uy
d,all,uz
allsel,all

asel,s,area,,54
nsla,s,1
d,all,ux
d,all,uy
d,all,uz
allsel,all

dmprat,0.001

Vin =300

asel,s,area,,20
nsla,s,1
cp,1,volt,all
*get,N1,node,0,num,min
d,N1,volt,0
allsel,all

asel,s,area,,22
nsla,s,1
cp,2,volt,all
*get,N2,node,0,num,min
d,N2,volt,Vin
allsel,all


asel,s,area,,15
nsla,s,1
cp,3,volt,all
*get,N3,node,0,num,min
d,N3,volt,Vin
allsel,all
```

```
asel,s,area,,17
nsla,s,1
cp,4,volt,all
*get,N4,node,0,num,min
d,N4,volt,0
allsel,all

SOLVE
FINI
```

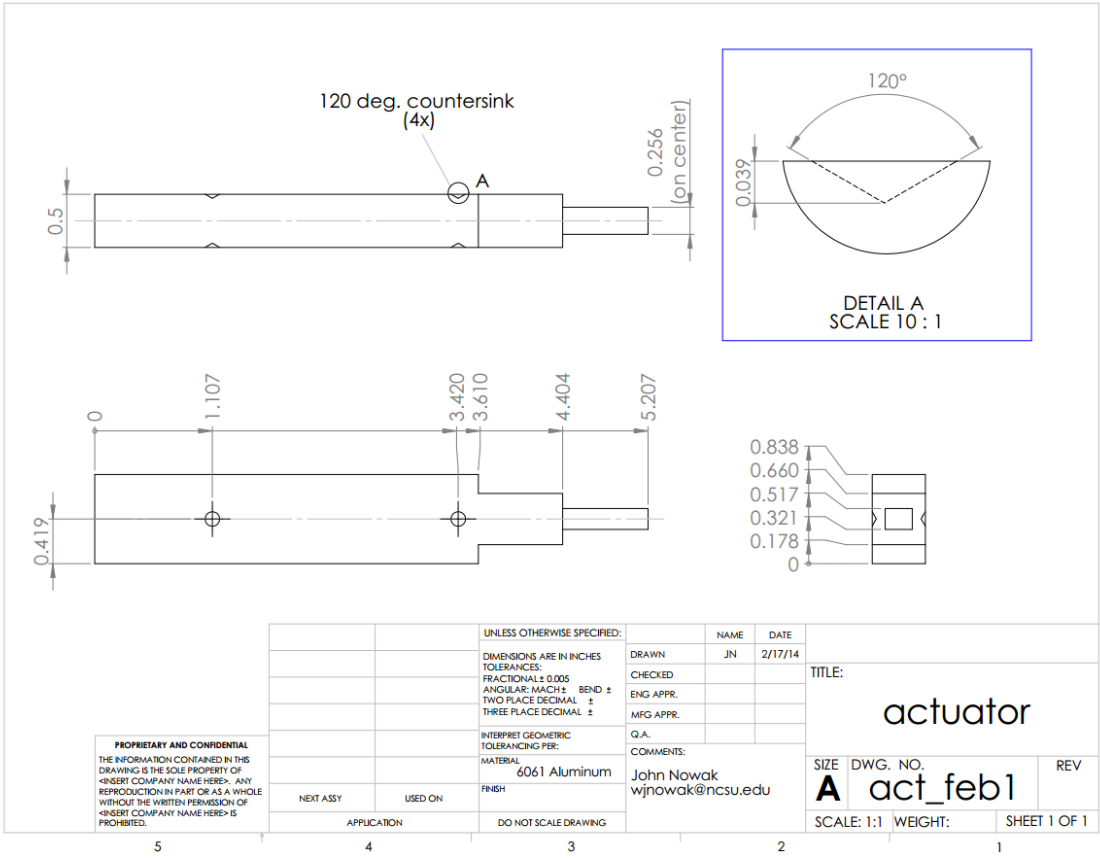
Appendix D.4 Bending Harmonic Analysis

```
/SOLU              !ENTER SOLUTION PROCESSOR
SOLCONTROL,0             !TURN OFF SOLUTION CONTROLS
ANTYPE,HARMIC            !SPECIFY ANALYSIS TYPE AS MODAL
HROPT,FULL                 ! Full harmonic response
!NSUBST,10                 ! 30 Intervals within freq. range
HARFRQ, 42742

!HARFRQ,39570.3
!harfrq,37000,40000

asel,s,area,,39
nsla,s,1
d,all,ux
d,all,uy
d,all,uz
allsel,all

asel,s,area,,44
nsla,s,1
d,all,ux
d,all,uy
d,all,uz
allsel,all

asel,s,area,,49
nsla,s,1
d,all,ux
d,all,uy
d,all,uz
allsel,all

asel,s,area,,54
nsla,s,1
d,all,ux
d,all,uy
d,all,uz
allsel,all

dmprat,0.001
```

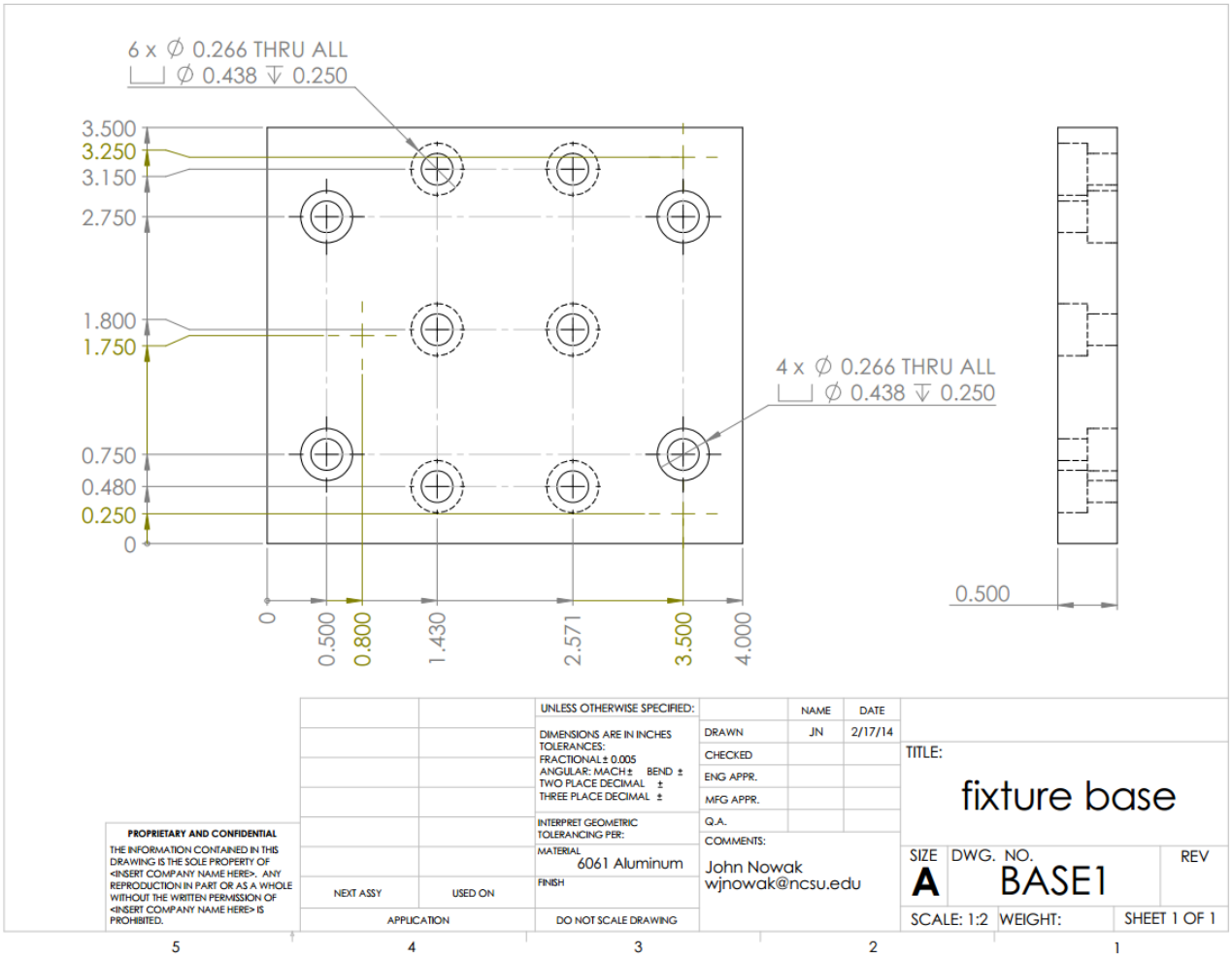
```
Vin =300

asel,s,area,,22
nsla,s,1
cp,1,volt,all
*get,N1,node,0,num,min
d,N1,volt,0
allsel,all

asel,s,area,,20
nsla,s,1
cp,2,volt,all
*get,N2,node,0,num,min
d,N2,volt,Vin
allsel,all


asel,s,area,,15
nsla,s,1
cp,3,volt,all
*get,N3,node,0,num,min
d,N3,volt,Vin
allsel,all

asel,s,area,,17
nsla,s,1
cp,4,volt,all
*get,N4,node,0,num,min
d,N4,volt,0
allsel,all

SOLVE
FIN
```

Appendix E 40 kHz Actuator Drawing

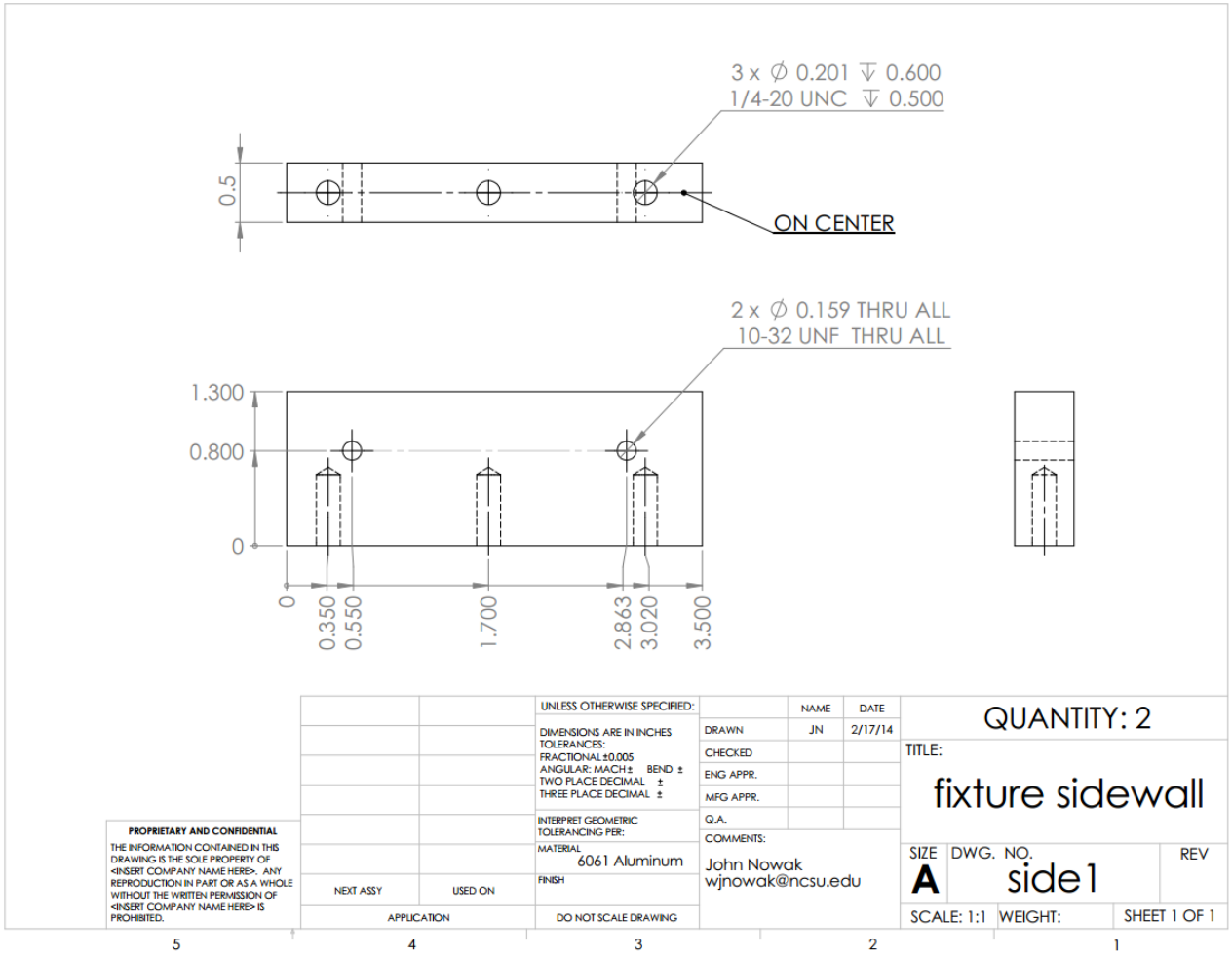
Appendix E.1 Actuator Body



Appendix E.2 Mount Base



Appendix E.3 Mount Sides



Appendix F PKI802 MATERIAL PROPERTIES[2]

Navy Type III		
ELECTROMECHANICAL PROPERTIES	802	809
PHYSICAL PROPERTIES		
Density (x 10^3 Kg/m^3)	7.6	7.7
Curie Temperature (°C)	350	310
Mechanical QM (-)	900	1000
Maximum Operating Temperature (°C)	150	(na)
ELECTRICAL PROPERTIES @ 25oC		
Dielectric Constant @ 1 KHz (-)	1000	1050
Dissipation Factor @ 1 KHz (%)	0.4	0.25
Planar Coupling Factor kp (-)	0.51	0.51
Transverse Coupling factor k31 (-)	0.30	0.30
Longitudinal Coupling Factor k33 (-)	0.61	0.63
Shear Coupling Factor k15 (-)	0.54	0.55
Transverse Charge Coefficient d31 (x 10^{-12} m/V)	-100	-92
Longitudinal Charge Coefficient d33 (x 10^{-12} m/V)	220	230
Shear Charge Coefficient d15 (x 10^{-12} m/V)	320	330
Transverse Voltage Coefficient g31 (x 10^{-3} V m/N)	-11.3	-9.0
Longitudinal Charge Coefficient g33 (x 10^{-3} V m/N)	24.9	27.0
Shear Charge Coefficient g15 (x 10^{-3} V m/N)	36.2	28.0

[2] http://www.piezo-kinetics.com/navy_type_III.php

MECHANICAL PROPERTIES @ 25oC		
Young's Modulus (x 10^{10} N/m2)	7.2	6.8
Poisson's Ratio (-)	0.31	0.3
Elastic Compliance $s11^E$ (x 10^{-12} m^2/N)	10.4	10.9
Elastic Compliance $s33^E$ (x 10^{-12} m^2/N)	13.5	15.2
FREQUENCY CONSTANTS @ 25oC		
Planar Frequency Constant Np (kHz inches)	92.9	91.3
Transverse Frequency Constant N1 (kHz inches)	68.9	66.9
Longitudinal Frequency Constant N3 (kHz inches)	78.7	78.7
Thickness Frequency Constant Nt (Hz-m)	2100	(na)
Shear Frequency Constant N5 (Hz-m)	1460	(na)
AGEING RATES		
Dielectric Constant (% per time decade)	-3.0	(na)
Resonant Frequency (% per time decade)	0.8	(na)
Coupling Constant (% per time decade)	-2.0	(na)
ELECTRIC FIELD DEPENDENCE @ 200KV/m [5 V/mil] and 25oC		
Dielectric Constant (% increase)	2.0	(na)
Dissipation Factor (% increase)	0.5	(na)
MAXIMUM OPERATING FIELD		
AC (KV/m [V/mil])	400 [10]	(na)
DC - forward (KV/m [V/mil])	800 [20]	(na)
DC - reverse (KV/m [V/mil])	400 [10]	(na)

Appendix G INDENT CALCULATOR

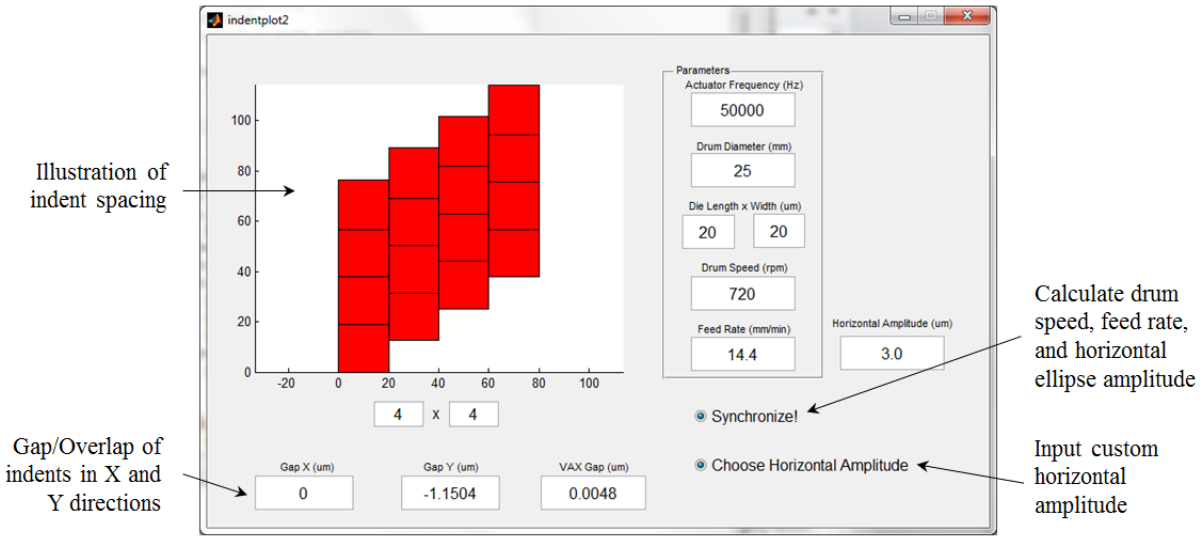


Figure G-6. Indent calucator using Matlab GUIDE.

Appendix G.1 Matlab Code

```
function varargout = indentplot2(varargin)
% INDENTPLOT2 MATLAB code for indentplot2.fig
%      INDENTPLOT2, by itself, creates a new INDENTPLOT2 or raises the
existing
%      singleton*.
%
%      H = INDENTPLOT2 returns the handle to a new INDENTPLOT2 or the
handle to
%      the existing singleton*.
%
%      INDENTPLOT2('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in INDENTPLOT2.M with the given input
arguments.
%
%      INDENTPLOT2('Property','Value',...) creates a new INDENTPLOT2 or
raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before indentplot2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to indentplot2_OpeningFcn via
varargin.
%
```

```
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help indentplot2

% Last Modified by GUIDE v2.5 19-Aug-2013 16:16:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @indentplot2_OpeningFcn, ...
                   'gui_OutputFcn',  @indentplot2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before indentplot2 is made visible.

function indentplot2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to indentplot2 (see VARARGIN)

% Choose default command line output for indentplot2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes indentplot2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

set(handles.radiobutton3,'Value',0);
set(handles.radiobutton4,'Value',0);
set(handles.feedRate,'Enable','on');
set(handles.wDrum,'Enable','on');
set(handles.ampA,'Enable','off');
set(handles.radiobutton4,'Enable','off');
updateAxes(handles);
```

```
% --- Outputs from this function are returned to the command line.

function varargout = indentplot2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Create Functions

function freqAct_CreateFcn(hObject, eventdata, handles)
% hObject    handle to freqAct (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function rDrum_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rDrum (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function wDrum_CreateFcn(hObject, eventdata, handles)
% hObject    handle to wDrum (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function Ldie_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ldie (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function feedRate_CreateFcn(hObject, eventdata, handles)
% hObject    handle to feedRate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function NoTall_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NoTall (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function NoWide_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NoWide (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function ampA_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ampA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function gapX_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gapX (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function gapY_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gapY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function rowSpace_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rowSpace (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function wdie_CreateFcn(hObject, eventdata, handles)
% hObject    handle to wdie (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%--- Callback functions
function freqAct_Callback(hObject, eventdata, handles)
% hObject    handle to freqAct (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of freqAct as text
%         str2double(get(hObject,'String')) returns contents of freqAct as
a double

pickCase(handles);

function rDrum_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to rDrum (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rDrum as text
%        str2double(get(hObject,'String')) returns contents of rDrum as a
double

pickCase(handles);

function wDrum_Callback(hObject, eventdata, handles)
% hObject    handle to wDrum (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of wDrum as text
%        str2double(get(hObject,'String')) returns contents of wDrum as a
double

pickCase(handles);

function Ldie_Callback(hObject, eventdata, handles)
% hObject    handle to Ldie (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Ldie as text
%         str2double(get(hObject,'String')) returns contents of Ldie as a
double

pickCase(handles);

function feedRate_Callback(hObject, eventdata, handles)
% hObject    handle to feedRate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of feedRate as text
%        str2double(get(hObject,'String')) returns contents of feedRate as
a double

pickCase(handles);

function NoTall_Callback(hObject, eventdata, handles)
% hObject    handle to NoTall (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of NoTall as text
%        str2double(get(hObject,'String')) returns contents of NoTall as a
double

pickCase(handles);
```

```
function NoWide_Callback(hObject, eventdata, handles)
% hObject    handle to NoWide (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of NoWide as text
%        str2double(get(hObject,'String')) returns contents of NoWide as a
double

pickCase(handles);

function ampA_Callback(hObject, eventdata, handles)
% hObject    handle to ampA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ampA as text
%          str2double(get(hObject,'String')) returns contents of ampA as a
double

pickCase(handles);

function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1

a = get(hObject,'Value');

if a==1
    set(handles.feedRate,'Enable','inactive');
    set(handles.wDrum,'Enable','inactive');
    set(handles.radiobutton4,'Enable','on');
    updateAxesSync(handles);
else
    set(handles.feedRate,'Enable','on');
    set(handles.wDrum,'Enable','on');
    set(handles.radiobutton4,'Enable','off');
    set(handles.ampA,'Enable','off');
end

function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton2
a = get(hObject,'Value');

if a==1
    set(handles.ampA,'Enable','on');
```

```
else
    set(handles.ampA,'Enable','off');
    updateAxesSync(handles);
end

function gapX_Callback(hObject, eventdata, handles)
function gapY_Callback(hObject, eventdata, handles)
function rowSpace_Callback(hObject, eventdata, handles)

function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton3
a = get(hObject,'Value');

if a==1
    set(handles.feedRate,'Enable','inactive');
    set(handles.wDrum,'Enable','inactive');
    set(handles.radiobutton4,'Enable','on');
    updateAxesSync(handles);
else
    set(handles.feedRate,'Enable','on');
    set(handles.wDrum,'Enable','on');
    set(handles.radiobutton4,'Value',0);
    set(handles.radiobutton4,'Enable','off');
    set(handles.ampA,'Enable','off');
end

function radiobutton4_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton4
a = get(hObject,'Value');

if a==1
    set(handles.ampA,'Enable','on');
else
    set(handles.ampA,'Enable','off');
    updateAxesSync(handles);
end

function wdie_Callback(hObject, eventdata, handles)
% hObject    handle to wdie (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of wdie as text
%        str2double(get(hObject,'String')) returns contents of wdie as a
double
```

```
pickCase(handles);

%--- My functions
function pickCase(handles)
a = get(handles.radiobutton3,'Value');

if a==1
    b = get(handles.radiobutton4,'Value');
    if b==1
        updateAxesSyncA(handles);
    else
        updateAxesSync(handles);
    end

else
    updateAxes(handles);
end
function updateAxes(handles)
% gets all input data
freqAct = get(handles.freqAct, 'string');  %kHz
freqAct = str2double(freqAct);

rDrum = get(handles.rDrum, 'string');  %kHz
rDrum = str2double(rDrum);
rDrum = rDrum/2;

wDrum = get(handles.wDrum, 'string');  %kHz
wDrum = str2double(wDrum);

Ldie = get(handles.Ldie, 'string');  %kHz
Ldie = str2double(Ldie);

wdie = get(handles.wdie, 'string');  %kHz
wdie = str2double(wdie);

feedRate = get(handles.feedRate, 'string');  %kHz
feedRate = str2double(feedRate);

NoTall = get(handles.NoTall, 'string');  %kHz
NoTall = str2double(NoTall);

NoWide = get(handles.NoWide, 'string');  %kHz
NoWide = str2double(NoWide);

% calculates indent position
TAct = 1/freqAct;  %sec
vDrum = rDrum * wDrum *pi/30;  %mm/s
spaceY = vDrum * TAct *1000; %um
spaceX = TAct * feedRate *1000/60;
rowX = (60 / wDrum)*feedRate*1000/60;  %sec
cDrum = 2*rDrum*pi;
noInd = cDrum*1000/spaceY;
newY = rem(cDrum*1000,spaceY);
```

```
gapY = spaceY - Ldie;
gapX = rowX - wdie;
spaceX;  %between indents

set(handles.gapX,'string',num2str(gapX));
set(handles.gapY,'string',num2str(gapY));
set(handles.rowSpace,'string',num2str(spaceX));

ampA = Ldie/2/pi;
set(handles.ampA,'string',num2str(ampA));  %rpm

% plots indents
x0 = [0;wdie;wdie;0;0]; y0 = [0;0;Ldie;Ldie;0];
cla;
hold on;
axis equal;
for j = 1:NoWide;
    for i = 1:NoTall;
        x = (x0 + (i-1)*spaceX) + (rowX*(j-1));
        startY = (j-1)*newY;
%         if startY > Ldie
%             startY = startY - (j-2)*Ldie;
%         end
        y = y0 + ((i-1)*spaceY) + startY;
        fill(x,y,'r-');
    end
end
hold off;

function updateAxesSync(handles)
% gets all input data
freqAct = get(handles.freqAct, 'string');  %kHz
freqAct = str2double(freqAct);

rDrum = get(handles.rDrum, 'string');  %mm
rDrum = str2double(rDrum);
rDrum = rDrum/2;

Ldie = get(handles.Ldie, 'string');  %um
Ldie = str2double(Ldie);

wdie = get(handles.wdie, 'string');  %kHz
wdie = str2double(wdie);

NoTall = get(handles.NoTall, 'string');  %#
NoTall = str2double(NoTall);

NoWide = get(handles.NoWide, 'string');  %#
NoWide = str2double(NoWide);

% calculates indent position
TAct = 1/freqAct;  %sec
```

```
wDrum = freqAct*Ldie*60/(rDrum*1000*2*pi);
feedRate = wDrum*wdie/1000;

vDrum = rDrum * wDrum *pi/30;  %mm/s
spaceY = vDrum * TAct *1000; %um
spaceX = TAct * feedRate *1000/60;
rowX = (60 / wDrum)*feedRate*1000/60;  %sec
cDrum = 2*rDrum*pi;
noInd = cDrum*1000/spaceY;
newY = rem(cDrum*1000,spaceY);
gapY = spaceY - Ldie;
gapX = rowX - wdie;
spaceX;  %between indents

set(handles.gapX,'string',num2str(gapX));
set(handles.gapY,'string',num2str(gapY));
set(handles.rowSpace,'string',num2str(spaceX));

ampA = Ldie/2/pi;
set(handles.ampA,'string',num2str(ampA));  %um

%wDrum = double2str(wDrum);
set(handles.wDrum,'string',num2str(wDrum));  %rpm

%feedRate = str2double(feedRate);
set(handles.feedRate, 'string',num2str(feedRate));  %mm/min

% plots indents
x0 = [0;wdie;wdie;0;0]; y0 = [0;0;Ldie;Ldie;0];
cla;
hold on;
axis equal;
for j = 1:NoWide;
    for i = 1:NoTall;
        x = (x0 + (i-1)*spaceX) + (rowX*(j-1));
        startY = (j-1)*newY;
%         if startY > Ldie
%             startY = startY - (j-2)*Ldie;
%         end
        y = y0 + ((i-1)*spaceY) + startY;
        fill(x,y,'r-');
    end
end
hold off;

function updateAxesSyncA(handles)
% gets all input data
freqAct = get(handles.freqAct, 'string');  %kHz
freqAct = str2double(freqAct);

rDrum = get(handles.rDrum, 'string');  %mm
rDrum = str2double(rDrum);
rDrum = rDrum/2;
```

```
Ldie = get(handles.Ldie, 'string');  %um
Ldie = str2double(Ldie);

wdie = get(handles.wdie, 'string');  %kHz
wdie = str2double(wdie);

NoTall = get(handles.NoTall, 'string');  %#
NoTall = str2double(NoTall);

NoWide = get(handles.NoWide, 'string');  %#
NoWide = str2double(NoWide);

ampA = get(handles.ampA, 'string');  %#
ampA = str2double(ampA);

% calculates indent position
TAct = 1/freqAct;  %sec

wDrum = freqAct*ampA*60/rDrum/1000;
feedRate = wDrum*wdie/1000;

vDrum = rDrum * wDrum *pi/30;  %mm/s
spaceY = vDrum * TAct *1000; %um
spaceX = TAct * feedRate *1000/60;
rowX = (60 / wDrum)*feedRate*1000/60;  %sec
cDrum = 2*rDrum*pi;
noInd = cDrum*1000/spaceY;
newY = rem(cDrum*1000,spaceY);
gapY = spaceY - Ldie ;
gapX = rowX - wdie;
spaceX;  %between indents

set(handles.gapX,'string',num2str(gapX));
set(handles.gapY,'string',num2str(gapY));
set(handles.rowSpace,'string',num2str(spaceX));


%wDrum = double2str(wDrum);
set(handles.wDrum,'string',num2str(wDrum));  %rpm

%feedRate = str2double(feedRate);
set(handles.feedRate, 'string',num2str(feedRate));  %mm/min

% plots indents
x0 = [0;wdie;wdie;0;0]; y0 = [0;0;Ldie;Ldie;0];
cla;
hold on;
axis equal;
for j = 1:NoWide;
    for i = 1:NoTall;
        x = (x0 + (i-1)*spaceX) + (rowX*(j-1));
        startY = (j-1)*newY;
%         if startY > Ldie
```

```
%             startY = startY - (j-2)*Ldie;
%         end
        y = y0 + ((i-1)*spaceY) + startY;
        fill(x,y,'r-');
    end
end
hold off;
```

Appendix H DIGITIAL POTENTIOMETER CONTROL

Appendix H.1 Matlab GUI for Controlling Digipot

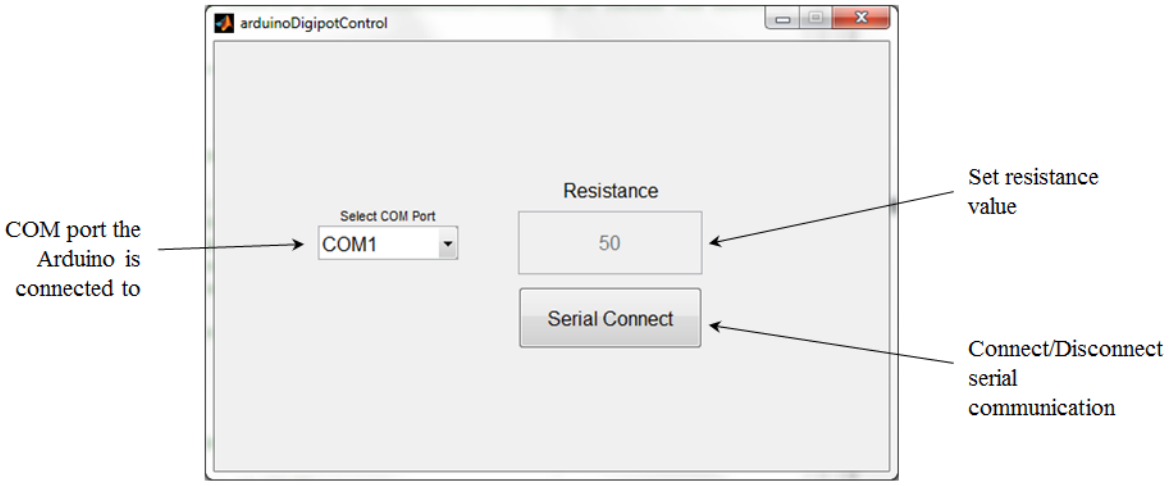


Figure H-7. Interface for serial connection between Arduino and Matlab.

Appendix H.2 Matlab code for digipot control

```
function varargout = arduinoDigipotControl(varargin)
% ARDUINODIGIPOTCONTROL MATLAB code for arduinoDigipotControl.fig
%                  ARDUINODIGIPOTCONTROL,   by   itself,   creates   a   new
ARDUINODIGIPOTCONTROL or raises the existing
%      singleton*.
%
%             H   =   ARDUINODIGIPOTCONTROL   returns   the   handle   to   a   new
ARDUINODIGIPOTCONTROL or the handle to
%      the existing singleton*.
%
%           ARDUINODIGIPOTCONTROL('CALLBACK',hObject,eventData,handles,...)
calls the local
%       function named CALLBACK in ARDUINODIGIPOTCONTROL.M with the given
input arguments.
%
%            ARDUINODIGIPOTCONTROL('Property','Value',...)  creates  a  new
ARDUINODIGIPOTCONTROL or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%        applied to the GUI before arduinoDigipotControl_OpeningFcn gets
called.  An
%          unrecognized property name or invalid value makes property
application
```

```
%        stop.  All inputs are passed to arduinoDigipotControl_OpeningFcn
via varargin.
%
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help arduinoDigipotControl

% Last Modified by GUIDE v2.5 08-Jan-2014 15:20:33

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',     @arduinoDigipotControl_OpeningFcn,
...
                   'gui_OutputFcn',  @arduinoDigipotControl_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before arduinoDigipotControl is made visible.
function  arduinoDigipotControl_OpeningFcn(hObject,  eventdata,  handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%  varargin      command  line  arguments  to  arduinoDigipotControl  (see
VARARGIN)

% Choose default command line output for arduinoDigipotControl
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

 % initialize values
 set(handles.edit1,'Value',50);
 set(handles.edit1,'string',50);
```

```
 set(handles.edit1,'Enable','off');


 i = get(handles.edit1,'String');
 i = str2double(i);
 inputData.prevData = i;
 setappdata(handles.output,'inputData',inputData);
```

```
% --- Executes on button press in togglebutton1.
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton1

state = get(hObject,'Value');
% set(handles.edit1,'string',state);
if state == 1
    set(handles.togglebutton1,'String','Connecting...');
    set(handles.popupmenu1,'Enable','off');

    contents = get(handles.popupmenu1,'Value')
    switch contents
        case 1
            com = 'COM1';
        case 2
            com = 'COM2';
        case 3
            com = 'COM3';
        case 4
            com = 'COM4';
        case 5
            com = 'COM5';
        case 6
            com = 'COM6';
        case 7
            com = 'COM7';
        case 8
            com = 'COM8';
    end

    % open comport
comport = struct('s',serial(com));
set(comport.s, 'DataBits', 8);
set(comport.s, 'StopBits', 1);
```

```
set(comport.s, 'BaudRate', 9600);
set(comport.s, 'Parity', 'none');
fopen(comport.s);

% handshaking
a = 'b';
while (a ~= 'a')
    a = fread(comport.s,1,'uchar');
end

if (a=='a')
    disp('serial read');
end

 fprintf(comport.s,'%c','a');
 mbox = msgbox('Serial Communication setup.'); uiwait(mbox);
 fscanf(comport.s,'%u');

 set(handles.togglebutton1,'String','Serial Disconnect');
 set(handles.edit1,'Enable','on');


 % save stuff
 setappdata(handles.output,'comport',comport);




else
    % load stuff
    comport = getappdata(handles.output,'comport');

    % close port
    fclose(comport.s);
    delete(comport.s);

    % enable com port select
    set(handles.popupmenu1,'Enable','on');
    set(handles.togglebutton1,'String','Serial Connect');
    set(handles.edit1,'Enable','off');
end

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double

inputData = getappdata(handles.output,'inputData');
```

```
i = get(handles.edit1,'String');
i = str2double(i);
minval = 0;
maxval = 127;

if isnan(i)
    set(handles.edit1,'String',inputData.prevData);
    set(handles.edit1,'Value',inputData.prevData);
elseif i > maxval
    i = maxval;
    set(handles.edit1,'String',i);
    set(handles.edit1,'Value',i);
    inputData.prevData = i;
elseif i < minval
    i = minval;
    set(handles.edit1,'String',i);
    set(handles.edit1,'Value',i);
    inputData.prevData = i;
else
    inputData.prevData = i;
end;

setappdata(handles.output,'inputData',inputData);

comport = getappdata(handles.output,'comport');
fwrite(comport.s,i,'uint8','sync');

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%  Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%              contents{get(hObject,'Value')} returns selected item from
popupmenu1


%--------------------------------------------------------------------------
------------------%
% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Outputs from this function are returned to the command line.
function  varargout  =  arduinoDigipotControl_OutputFcn(hObject,  eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

Appendix H.3 Arduino code for matlab GUI interface

```
#include <SPI.h>

const int slaveSelectPin = 10;
int data = 100;
int incomingByte = 0;
int i = 0;
void setup()
{
  pinMode (slaveSelectPin, OUTPUT);
  //setup serial
  Serial.begin(9600);
  Serial.println('a');
  char a = 'b';
  while (a !='a'){
    a = Serial.read();
  }
  //initialize spi
  SPI.begin();
  SPI.setBitOrder(MSBFIRST);
  SPI.setDataMode(SPI_MODE0);
}

void loop()
{
  if (i==0) {
```

```
    digitalWrite(slaveSelectPin,LOW);
    SPI.transfer(0b00000000);
    SPI.transfer(data);
    digitalWrite(slaveSelectPin,HIGH);
    i = 1;
   }


  if (Serial.available() > 0) {
    incomingByte = Serial.read();
    digitalWrite(slaveSelectPin,LOW);
    SPI.transfer(0b00000000);
    SPI.transfer(incomingByte);
    digitalWrite(slaveSelectPin,HIGH);
  }
}
```

Appendix I TILTED ELLIPSE CALCULATIONS

Appendix I.1 Velocity Correction Factor

In the ideal ellipse case, the horizontal and vertical (x and y, respectively) actuator motion are 90° out-of-phase such that the maximum x-velocity occurs at the minimum y-position when the die is contacting the workpiece. This relationship is shown in Figure I-8

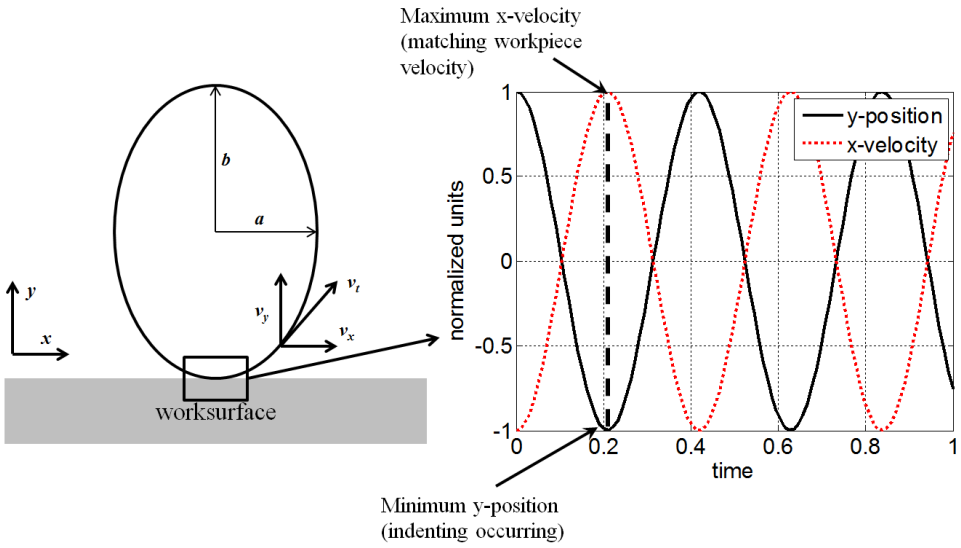


Figure I-8. Ideal elliptical indenting.

If the *x* and *y* motion of the ellipse are not 90° out-of-phase, a tilted ellipse is produced. This has an effect on the workpiece speed because at the point of minimum y-position (part contact), the horizontal velocity component of the ellipse is not at a maximum. This creates smear due to velocity mismatch between the die and workpiece. An illustration of the tilted ellipse is given in Figure I-9.

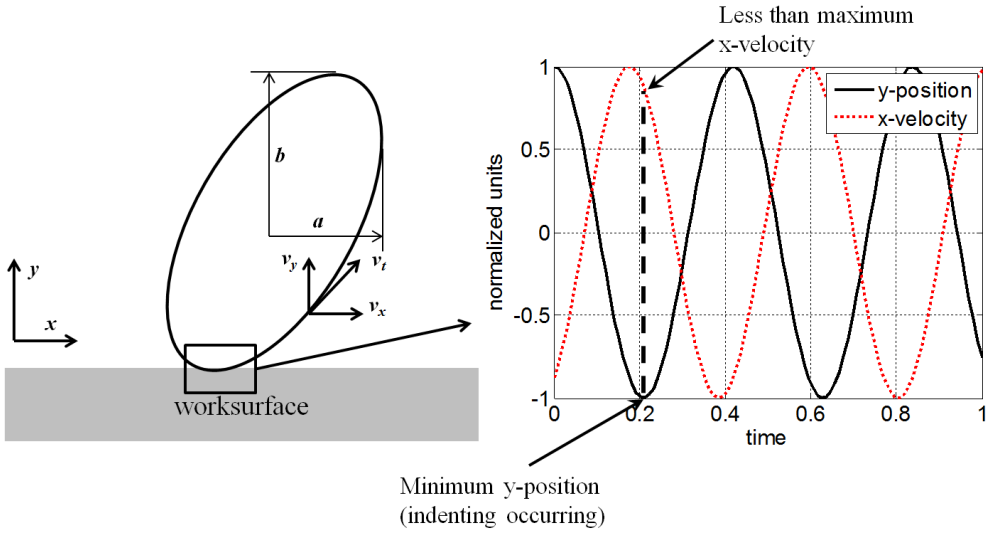


Figure I-9. Tilted elliptical indenting.

Since the governing equations of matching the speed assume a 90° phase between the *x* and *y* motions of the ellipse (Figure I-8), this tilted ellipse shape must be compensated for when setting the workpiece speed and crossfeed rate. The normalized horizontal velocity at minimum y-position is plotted as a function of phase angle between the *x* and *y* motions, where 90° is the ideal case. This is shown in Figure I-10.

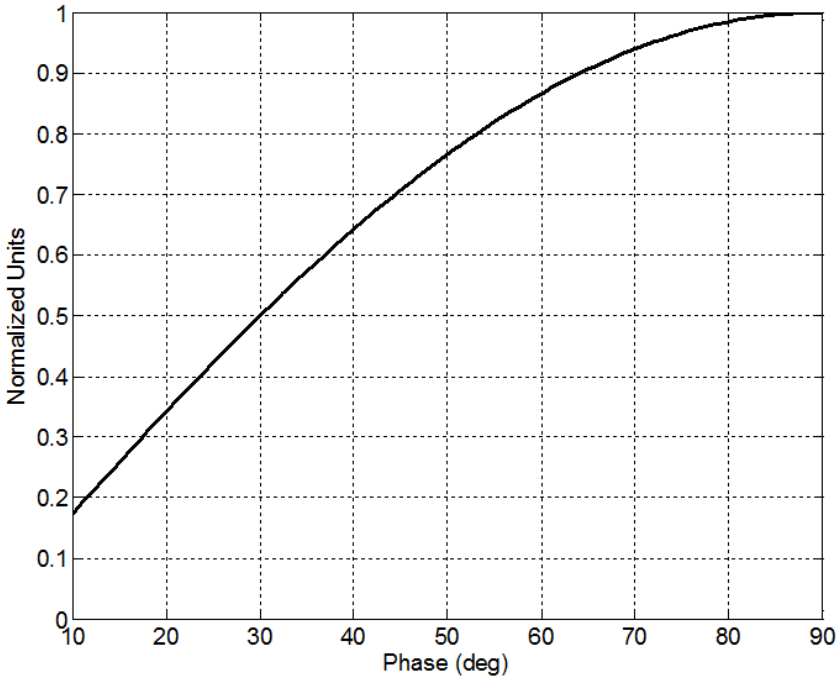


Figure I-10. Correction factor for workpiece velocity based on ellipse phase angle.

The correction factor can be used when the ellipse shape is not 90°. For example, if the ellipse motions are 60° out-of-phase, the workpiece speed and crossfeed rate should be multiplied by a factor of 0.86 to properly match the die and workpiece speed.

Appendix I.2 Correction Factor Matlab Code

```
%%% calculate correction factor based on
%%% ellipse phase angle

%%% John Nowak

clear all
clc

ph = linspace(10,90,90)';
w = 40000*2*pi;
A = 3.2e-6;
Vx = A*w*sind(ph);
Vdes = max(Vx);
Acomp = (Vdes/w)./sind(ph);
Acompum = Acomp*1e6;
```

```
figure(1),
plot(ph,Vx/Vdes,'k'),...
    xlabel('Phase (deg)'),ylabel('Normalized Units'),...
    title('Surface Velocity Correction'),...
    axis([10 90 0 1]),grid;
repPlotArial;
```

Appendix I.3 Calculating Curvature

The previously developed relationships for the required horizontal amplitude a of the elliptical path that is need given a die of length L_{die} assume a perfect ellipse where the indenting motion in the y-direction is 90° out-of-phase with the x, or bending, direction. This means that the maximum horizontal velocity v_x is occurring at the minimum y-position, where the die is contacting the workpiece. A schematic of the elliptical path is shown in Figure I-11.

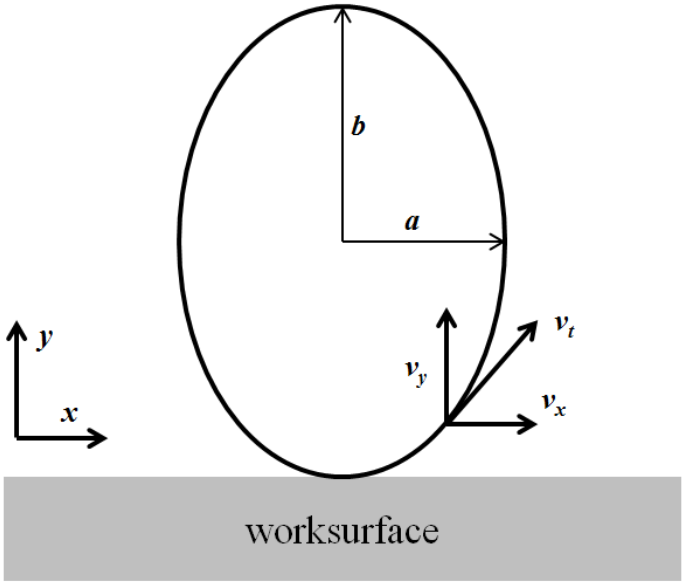


Figure I-11. Elliptical path.

The required amplitude a has previously been derived as:

$$a = \frac{L_{die}}{2\pi} \tag{I-1}$$

assuming that the x-velocity component is at a maximum during contact with the die such that:

$$v_t = v_x = a\omega \tag{I-2}$$

where v_t is the tangential speed of the die path, v_x is the x-component of velocity and ω is the frequency. Along with this velocity constraint, simulations have shown that for acceptable distortion, the ellipse aspect ratio must be greater than 1:1, or:

$$b > a \tag{I-3}$$

For the case of a tilted ellipse, where the x and y motions are not 90° out-of-phase, another strategy must be employed to compensate for when Equation (I-2) does not describe the motion during contact between the die and surface. Also, another method of estimating the equivalent aspect ratio must be derived. An example of the tilted ellipse shape is shown in Figure I-12.

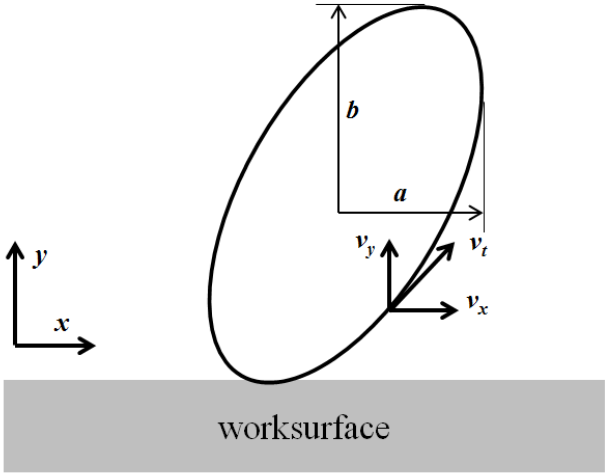


Figure I-12. Tilted ellipse schematic

Curvature k is defined as the change in the tangent vector T of a line over the change in the arc length s, or the inverse of the radius of a circle R that shares the same curvature. Mathematically this can be expressed as [59]:

$$k = \frac{1}{R} = \left\| \frac{dT}{ds} \right\| \tag{I-4}$$

where

$$T = \frac{r'}{\|r\|} \tag{I-5}$$

where the line is described by its position in XY space as:

$$r = \begin{bmatrix} x \\ y \end{bmatrix} \tag{I-6}$$

$$r' = \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{I-7}$$

For the ideal ellipse case with no curvature, the ellipse must have a vertical to horizontal aspect ratio of 3:1; however, for this analysis the goal is to determine an expression for a curvature requirement at any point on the ellipse for an aspect ratio of at least 1:1. This means that the smallest ellipse shape acceptable is a circle of radius a, defined in Equation (I-1). This a dimensions can be related to actuator frequency and workpiee velocity by rearranging Equation (I-2) as the following:

$$a = \frac{v_t}{\omega} \tag{I-8}$$

Therefore, the curvature requirement of the indenting path while the die is in contact with the workpiece for a 1:1 ratio elliptical path is a circle with radius *a*. From Equation (I-8), the curvature requirement for indenting can be expressed as:

$$k = \frac{1}{a} = \frac{\omega}{v_t} \tag{I-9}$$

Appendix I.4 Tilted Ellipse Calculator

A Matlab GUIDE program was developed to calculate the curvature and tangent velocity at each point on an ellipse. The ellipse phase can be set between 0 and 90° to consider tilted elliptical paths. Curvature and velocity markers are used to show where on the path these requirements are met. The required surface speed calculated by considering the tilt of the ellipse is also displayed. This GUI is shown in Figure I-13.

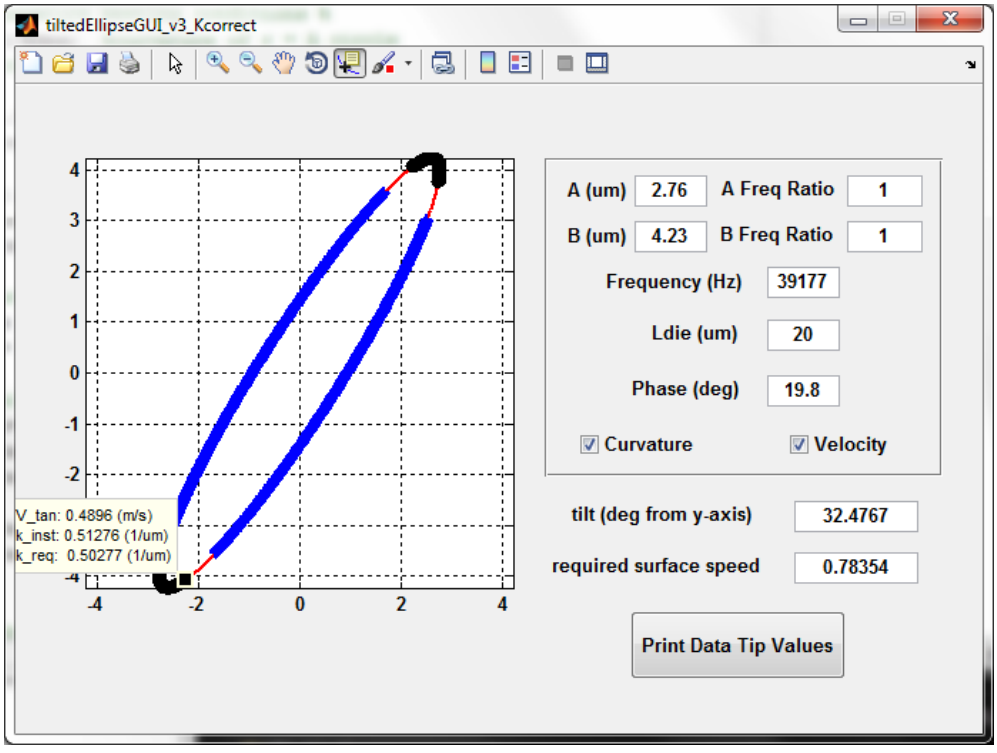


Figure I-13. Program to calculate instantaneous curvature and velocity for a tilted ellipse.

Appendix I.5 Tilted Ellipse Calculator Matlab Code

```
function varargout = tiltedEllipseGUI_v3_Kcorrect(varargin)
% TILTEDELLIPSEGUI_V3_KCORRECT MATLAB code for
tiltedEllipseGUI_v3_Kcorrect.fig
%      TILTEDELLIPSEGUI_V3_KCORRECT, by itself, creates a new
TILTEDELLIPSEGUI_V3_KCORRECT or raises the existing
%      singleton*.
%
%      H = TILTEDELLIPSEGUI_V3_KCORRECT returns the handle to a new
TILTEDELLIPSEGUI_V3_KCORRECT or the handle to
%      the existing singleton*.
%
%
TILTEDELLIPSEGUI_V3_KCORRECT('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in TILTEDELLIPSEGUI_V3_KCORRECT.M with the
given input arguments.
%
%      TILTEDELLIPSEGUI_V3_KCORRECT('Property','Value',...) creates a new
TILTEDELLIPSEGUI_V3_KCORRECT or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before tiltedEllipseGUI_v3_Kcorrect_OpeningFcn
gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to
tiltedEllipseGUI_v3_Kcorrect_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
tiltedEllipseGUI_v3_Kcorrect

% Last Modified by GUIDE v2.5 13-May-2014 16:36:34

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@tiltedEllipseGUI_v3_Kcorrect_OpeningFcn, ...
                   'gui_OutputFcn',
@tiltedEllipseGUI_v3_Kcorrect_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
```

```
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before tiltedEllipseGUI_v3_Kcorrect is made visible.
function tiltedEllipseGUI_v3_Kcorrect_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to tiltedEllipseGUI_v3_Kcorrect (see
VARARGIN)

% Choose default command line output for tiltedEllipseGUI_v3_Kcorrect
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes tiltedEllipseGUI_v3_Kcorrect wait for user response (see
UIRESUME)
% uiwait(handles.figure1);
clc;

set(handles.checkbox1,'Value',0);
set(handles.checkbox1,'Value',0);

A = get(handles.A, 'string');  %um
A = str2double(A);

B = get(handles.B, 'string');  %um
B = str2double(B);

wHz = get(handles.wHz, 'string');  %kHz
wHz = str2double(wHz);

Ldie = get(handles.Ldie, 'string');  %um
Ldie = str2double(Ldie);

phiDeg = get(handles.phiDeg, 'string');  %degrees
phiDeg = str2double(phiDeg);

bratio = get(handles.bratio, 'string');  %ratio
```

```
bratio = str2double(bratio);

aratio = get(handles.aratio, 'string');  %ratio
aratio = str2double(aratio);

defaults.A = A;
defaults.B = B;
defaults.wHz = wHz;
defaults.Ldie = Ldie;
defaults.phiDeg = phiDeg;
defaults.bratio = bratio;
defaults.aratio = aratio;

setappdata(handles.output,'defaults',defaults);

updateAxes(handles);

% --- Outputs from this function are returned to the command line.
function varargout = tiltedEllipseGUI_v3_Kcorrect_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function A_CreateFcn(hObject, eventdata, handles)
% hObject    handle to A (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function B_CreateFcn(hObject, eventdata, handles)
% hObject    handle to B (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function wHz_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to wHz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function Ldie_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ldie (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function phiDeg_CreateFcn(hObject, eventdata, handles)
% hObject    handle to phiDeg (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function tilt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to tilt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function Vreq_ms_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Vreq_ms (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end
function aratio_CreateFcn(hObject, eventdata, handles)
% hObject    handle to aratio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function bratio_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bratio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function A_Callback(hObject, eventdata, handles)
% hObject    handle to A (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of A as text
%        str2double(get(hObject,'String')) returns contents of A as a
double


% --- Executes during object creation, after setting all properties.

updateAxes(handles);
function B_Callback(hObject, eventdata, handles)
% hObject    handle to B (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of B as text
%        str2double(get(hObject,'String')) returns contents of B as a
double


% --- Executes during object creation, after setting all properties.

updateAxes(handles)
```

```
function wHz_Callback(hObject, eventdata, handles)
% hObject    handle to wHz (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of wHz as text
%        str2double(get(hObject,'String')) returns contents of wHz as a
double


% --- Executes during object creation, after setting all properties.

updateAxes(handles);
function Ldie_Callback(hObject, eventdata, handles)
% hObject    handle to Ldie (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Ldie as text
%        str2double(get(hObject,'String')) returns contents of Ldie as a
double


% --- Executes during object creation, after setting all properties.

updateAxes(handles);
function phiDeg_Callback(hObject, eventdata, handles)
% hObject    handle to phiDeg (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of phiDeg as text
%        str2double(get(hObject,'String')) returns contents of phiDeg as a
double


% --- Executes during object creation, after setting all properties.


% --- Executes on button press in checkbox1.

updateAxes(handles);
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1

% --- Executes on button press in checkbox2.
```

```
updateAxes(handles);
function checkbox2_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox2

updateAxes(handles);
function tilt_Callback(hObject, eventdata, handles)
% hObject    handle to tilt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of tilt as text
%        str2double(get(hObject,'String')) returns contents of tilt as a
double



% --- Executes during object creation, after setting all properties.

1+1;
function Vreq_ms_Callback(hObject, eventdata, handles)
% hObject    handle to Vreq_ms (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Vreq_ms as text
%        str2double(get(hObject,'String')) returns contents of Vreq_ms as
a double



% --- Executes during object creation, after setting all properties.

1;
function aratio_Callback(hObject, eventdata, handles)
% hObject    handle to aratio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of aratio as text
%        str2double(get(hObject,'String')) returns contents of aratio as a
double



% --- Executes during object creation, after setting all properties.
updateAxes(handles);
function bratio_Callback(hObject, eventdata, handles)
% hObject    handle to bratio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of bratio as text
%        str2double(get(hObject,'String')) returns contents of bratio as a
double


% --- Executes during object creation, after setting all properties.

updateAxes(handles);

% % My Functions
function updateAxes(handles)

% Reads and checks all inputs
A = checkInputs(handles.A,0,100);
B = checkInputs(handles.B,0,100);
wHz = checkInputs(handles.wHz,0,1000000);
Ldie = checkInputs(handles.Ldie,0,100);
phiDeg = checkInputs(handles.phiDeg,0,3600);
aratio = checkInputs(handles.aratio,0,100);
bratio = checkInputs(handles.bratio,0,100);

% Conversions
w = wHz*2*pi;
phi = phiDeg*pi/180;
phi0 = pi/2;
Tp = 1/wHz;
N = 1000;
t = linspace(0,Tp,N);

Ades = Ldie/2/pi;

%  Elliptical Motion
x = A*sin(aratio*w*t);  y = B*sin(bratio*w*t+phi);
vx = A*aratio*w*cos(aratio*w*t);  vy = B*bratio*w*cos(bratio*w*t+phi);

% calculate absolute distance from origin
pts = [x; y]';
hyp = [sqrt(x.^2+y.^2)]';
hypY = [sqrt(x.^2+y.^2)]'.*sign(y)';

% find closed and furthest points from origin on ellipse
% these are the relative major/minor axes
maxInd = find(max(hypY)==hyp);
minInd = find(min(hyp)==hyp);
HypMax = pts(maxInd,:);
HypMin = pts(minInd,:);

% Form lines to relative major/minor axes
```

```
Bx = [0 HypMax(1,1)];
By = [0 HypMax(1,2)];
Ax = [0 HypMin(1,1)];
Ay = [0 HypMin(1,2)];

% Calculate A/B relative to tilt
Brel = sqrt(Bx(2).^2+By(2).^2);
Arel = sqrt(Ax(2).^2+Ay(2).^2);


%%% Velocity Calcs %%%%
vx1 = Ades*w;  % Horizontal velocity @ 90 deg phase
vxrel = Arel*w;  % Relative horizontal velocity

% Resolve Velocity Components
vxMag = sqrt(vx.^2+vy.^2);
vxrel_alt = vxMag(maxInd);


% calculate horizontal velocity based on tilted ellipse/normal actuator
ind = find(min(y)==y);
vxrel2 = abs(vx(ind));  vxrel2 = vxrel2(1);  % relative horizontal
velocity

c1x = Bx(2)*-1;  c1y = By(2)*-1;
c2x = x(ind);  c2y = y(ind);

% % % % % % % % % % % % Calculate Curvature % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % http://tutorial.math.lamar.edu/Classes/CalcIII/Curvature.aspx % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

r = pts;  % curve
drdt = [vx' vy'];  %tangent vector
norm = sqrt(drdt(:,1).^2+drdt(:,2).^2);

n = length(norm);
for i = 1:n
    T(i,:) = drdt(i,:)./norm(i);  % % unit tangent vector
end

% k = ||dT/dr||

dT = diff(T);
dr = diff(pts);

dTmag = sqrt(dT(:,1).^2+dT(:,2).^2);
drmag = sqrt(dr(:,1).^2+dr(:,2).^2);

% plot velocity that meets requirements
```

```
Vreq = Ades*w;
VmatchInd = find(vxMag>=Vreq);
Vpts = pts(VmatchInd,:);

% plot curvature meeting conditions %
kcirc = 1/Ades;  %curvature of r = A circle
% new method of finding required k = kreq

kreq = w./vxMag;
kreq = kreq(1:length(kreq)-1)';

k = dTmag./drmag;
kind = find(k>=kreq);
kpts = pts(kind,:);

% Calculate tilt angle
tiltAngle = asin(Bx(2)/Brel)*180/pi;
Vreq_ms = Vreq/1000000;

% only calculate tilt when shape is ellipse or circle
if aratio==bratio
    set(handles.tilt,'Enable','inactive');
    set(handles.tilt,'string',num2str(tiltAngle));
else
    set(handles.tilt,'Enable','off');
    set(handles.tilt,'string','xxx');
end

% set required velocity to match speed
set(handles.Vreq_ms,'string',num2str(Vreq_ms,6));

% save data that may be plotted
plotdata.x = x;
plotdata.y = y;
plotdata.Bx = Bx;
plotdata.By = By;
plotdata.Ax = Ax;
plotdata.Ay = Ay;
plotdata.kpts = kpts;
plotdata.Vpts = Vpts;
plotdata.vxMag = vxMag;
plotdata.k = k;
plotdata.kreq = kreq;

setappdata(handles.output,'plotdata',plotdata);

GUIinputs.A = A;
GUIinputs.B = B;
GUIinputs.wHz = wHz;
GUIinputs.Ldie = Ldie;
GUIinputs.phiDeg = phiDeg;
```

```
GUIinputs.aratio = aratio;
GUIinputs.bratio = bratio;

setappdata(handles.output,'GUIinputs',GUIinputs);

pickCase(handles);
function pickCase(handles)

a = get(handles.checkbox1,'Value');
b = get(handles.checkbox2,'Value');

if a == 0 & b ==0
    plotstuff1(handles);
elseif a==1 & b==0
    plotstuff2(handles);
elseif a==0 & b==1
    plotstuff3(handles);
elseif a==1 & b==1
    plotstuff4(handles);
end

function plotstuff1(handles)

plotdata = getappdata(handles.output,'plotdata');
x = plotdata.x;
y = plotdata.y;
Bx = plotdata.Bx;
By = plotdata.By;
Ax = plotdata.Ax;
Ay = plotdata.Ay;
kpts = plotdata.kpts;
Vpts = plotdata.Vpts;


cla;
plot(x,y,'r','LineWidth',2);
axis equal;
grid on;

plotFormat(handles);
function plotstuff2(handles)

plotdata = getappdata(handles.output,'plotdata');
x = plotdata.x;
y = plotdata.y;
Bx = plotdata.Bx;
By = plotdata.By;
Ax = plotdata.Ax;
Ay = plotdata.Ay;
kpts = plotdata.kpts;
Vpts = plotdata.Vpts;
```

```
cla;
plot(x,y,'r',kpts(:,1),kpts(:,2),'ko','LineWidth',2);
axis equal;
grid on;

plotFormat(handles);
function plotstuff3(handles)

plotdata = getappdata(handles.output,'plotdata');
x = plotdata.x;
y = plotdata.y;
Bx = plotdata.Bx;
By = plotdata.By;
Ax = plotdata.Ax;
Ay = plotdata.Ay;
kpts = plotdata.kpts;
Vpts = plotdata.Vpts;


cla;
plot(x,y,'r',Vpts(:,1),Vpts(:,2),'b+','LineWidth',2);
axis equal;
grid on;

plotFormat(handles);
function plotstuff4(handles)

plotdata = getappdata(handles.output,'plotdata');
x = plotdata.x;
y = plotdata.y;
Bx = plotdata.Bx;
By = plotdata.By;
Ax = plotdata.Ax;
Ay = plotdata.Ay;
kpts = plotdata.kpts;
Vpts = plotdata.Vpts;


cla;
plot(x,y,'r',kpts(:,1),kpts(:,2),'ko',Vpts(:,1),Vpts(:,2),'b+','LineWidth'
,2);
axis equal;
grid on;

plotFormat(handles);

function val = checkInputs(h,min,max)
b = get(h,'Value');  % previous value
a = get(h,'String'); % new input
val = str2num(a);
```

```
% input range
if (nargin < 3),        max = +inf;  end
if (nargin < 2),        min = -inf;  end

% check inputs
if (isnan(val)),        val = b;
elseif (isempty(val)), val = b;
elseif (val < min),     val = min;
elseif (val > max),     val = max;   end

set(h,'String',num2str(val));  % set display value
set(h,'Value',val);            % set value
function plotFormat(handles)
set(findall(gca,'-property','FontName'),  'FontName',  'Arial');
set(findall(gca,'-property','FontSize'),  'FontSize',  10);
set(findall(gca,'-property','FontWeight'),'FontWeight','Bold');

dh = datacursormode();
set(dh,'Enable','on','DisplayStyle','datatip', ...
       'SnapToDataVertex','on','UpdateFcn',{@datatipMZ,handles});
function output_txt = datatipMZ(obj,event_obj,handles)
% DATATIPMZ  Display the position of the data cursor
%   output_txt = datatipMZ(obj,event_obj)
% Input
%   obj          Currently not used (empty)
%   event_obj    Handle to event object
% Output
%   output_txt   Data cursor text string (string or cell array of strings)

plotdata = getappdata(handles.output,'plotdata');

vxMag = plotdata.vxMag;
x = plotdata.x;
k = plotdata.k;
kreq = plotdata.kreq;

pos        = get(event_obj,'Position');
vxShow = vxMag(find(pos(1)==x))/1000000;
kShow = k(find(pos(1)==x));
kreqShow = kreq(find(pos(1)==x));

plotdata.vxShow = vxShow;
plotdata.kShow = kShow;
plotdata.kreqShow = kreqShow;

setappdata(handles.output,'plotdata',plotdata);

output_txt = {['V_tan: ',num2str(vxShow),' (m/s)'] ...
              ['k_inst: ',num2str(kShow),' (1/um)'] ...
              ['k_req:  ',num2str(kreqShow),' (1/um)']};
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

plotdata = getappdata(handles.output,'plotdata');
GUIinputs = getappdata(handles.output,'GUIinputs');

if isfield(plotdata,'vxShow')==0,
else
vxShow = plotdata.vxShow;
kShow = plotdata.kShow;
fprintf('Export Results:');
fprintf('\n');
fprintf(['x = ',num2str(GUIinputs.A),'sin(',num2str(GUIinputs.aratio),...
    'wt)']);
fprintf('\n');
fprintf(['y = ',num2str(GUIinputs.B),'sin(',num2str(GUIinputs.bratio),...
    'wt + ',num2str(GUIinputs.phiDeg),')']);
fprintf('\n');
fprintf(['k = ',num2str(kShow),' (1/um)']);
fprintf('\n');
fprintf(['Vt = ',num2str(vxShow), ' (m/s)']);
fprintf('\n');
fprintf('\n');
end
```