Finite Element Modeling of Silicon Bead Placement for Ultrasonic Vagus Nerve Excitation

Chinmay Murthy[1], Benyamin Davaji[2], and Amit Lal[2]

[1] Mission San Jose High School
chinmaymurthy474@gmail.com

[2]Dept. of Electrical and Computer Engineering
Cornell University
{bendavaji,amit.lal}@cornell.edu

2020/11/23

Abstract

In this paper, using finite element modeling, we investigate the possibility of using diffraction effects to focus and amplify ultrasonic waves via the placement of silicon beads in the body (water is used as the medium here for simplicity). We show that pressure waves are amplified and localized when specifically sized beads are placed, and the effects of amplification are seen even when the bead is not in line with the ultrasonic transducer, although optimum amplification results when the bead is exactly inline with the transducer. We also show that we can use stereo microphones to detect the specific location of the bead in relation to the pulse transducer so that precise control is possible for linear alignment to maximize pressure amplification. This work is motivated by the need to seek more efficient and less invasive ways of stimulating the vagus nerve in the human body, and has many applications in medicine.

1 Introduction

The Vagus nerves are the tenth pair of the 12 cranial nerves in the human body, and run from the brain stem, down both sides of the neck, to the chest. They are responsible for both sensory and motor functions, some of which include visceral sensation information for most of the digestive tract, touch information behind the ear, stimulating muscles in the digestive tract, and stimulating muscles in the heart, helping to lower the resting heart rate. Stimulation of these nerves is thought to be able to treat numerous medical conditions, such as epilepsy [1], depression, multiple sclerosis, Alzheimer's disease, and cluster headaches [2]. Current excitation methods typically involve an implanted electronic device that transmits pulses of electricity through small wires threaded under the skin to the nerve. However, these devices are around the size of a silver dollar, and require major surgery to implant. This surgery can result in a plethora of negative side-effects, including damage to the nerve, breathing issues, vocal chord paralysis (if the device moves after implantation), and throat pain [3].

This paper explores the possibility of using ultrasound to stimulate the nerve, paired with beads orders of magnitude smaller than the aforementioned device that are easily injected into the patient. Using ultrasound to stimulate this nerve directly has been shown to be possible in literature, for example in [4], and [5].

To analyze the effectiveness and potential methods of implementation, OnScale[1], a FEM analysis program is used to simulate the wave propagation dynamics.

This report is organized as follows. In Section 2, we

[1]Its product site: https://onscale.com

demonstrate OnScale's accuracy by comparing it to a known equation. In section 3 we simulate several scenarios. First, in Section 3.1, we simulate the wave propagation with a bead directly on the central axis. Next, in Section 3.2, we move the bead off of the central axis. Finally, in Section 3.3, we analyze sensor feedback to determine the bead position.

2 Theoretical Analysis

To confirm the accuracy of OnScale, its results can be directly compared to a calculated expected result.

The phasor velocity v of a particle in a medium with bulk velocity c_0, z distance away from the center of a transducer of diameter a oscillating at frequency ω at speed v_0 can be expressed by the following

$$v(z:\omega) = v_0(1 - \frac{z}{\sqrt{a^2+z^2}} e^{-j(\omega/c_0)(\sqrt{a^2+z^2}-z)}) e^{-j\omega z/c_0} \quad (1)$$

[6]

Figure 1 shows this equation when plotted given a relevant set of parameters.

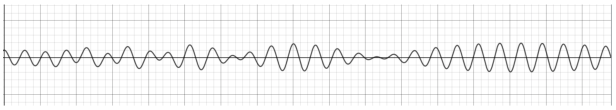
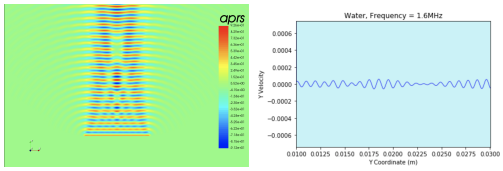


Figure 1: Calculated line plot of equation 1

Now, a simulation is run in OnScale with the same parameters as above. The equation only describes the expected values along the transducer's central axis, so that can be plotted as well, as shown in Figure 2.



(a) Simulation result (b) Simulation line plot along the central axis

Figure 2: Simulated results

The graphs in Figure 1 and Figure 2 are nearly identical. This confirms OnScale's simulation accuracy in this case.

3 Experimental Process

3.1 Axisymmetric Model

The first model that was run consisted of a sphere of either silicon or epoxy in a cylinder of water with air below (as seen in Figure 3). A piezoelectric disk was used as the transducer at the interface between the water and air to propagate ultrasound waves through the water. A test simulation with this setup is shown in Figure 4

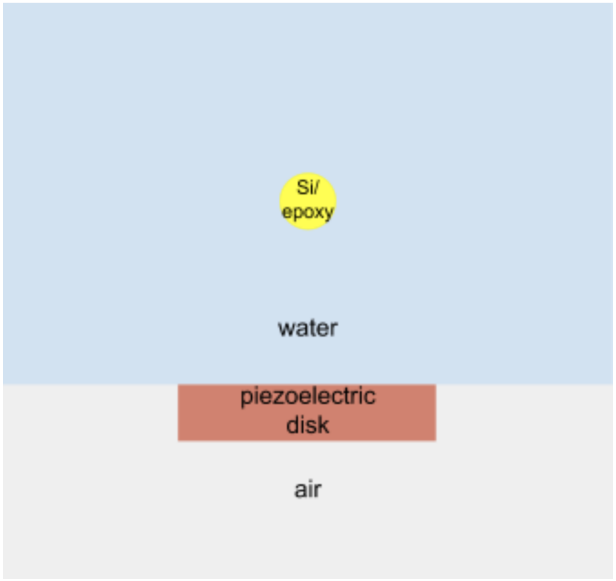


Figure 3: Axisymmetric model setup with piezoelectric transducer

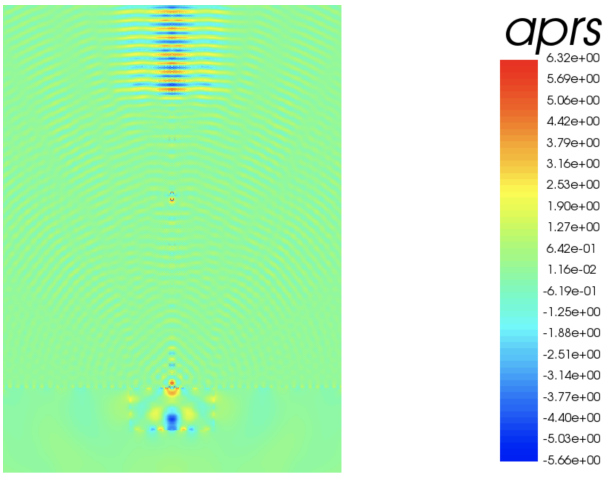


Figure 4: Simulation results of initial Axisymmetric setup

The legend labeled 'aprs' on the right hand side maps the colors in the simulation visualization to their calculated pressure values. While this did yield some interesting results, a major issue with it was that the transducer was not vibrating uniformly and was not producing plane waves, as visible in Figure 5.

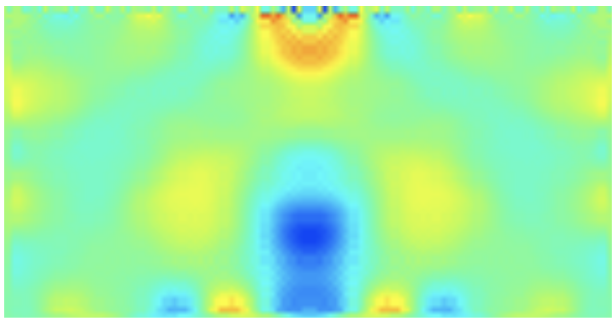


Figure 5: A closeup of the transducer reveals non-uniform vibration

To fix this, we tried various things, such as thickening the transducer, and replacing the air with a hard material. We ended up removing the transducer altogether and simply applying a pressure load to the interface between the water and air. The updated setup is shown in Figure 6.

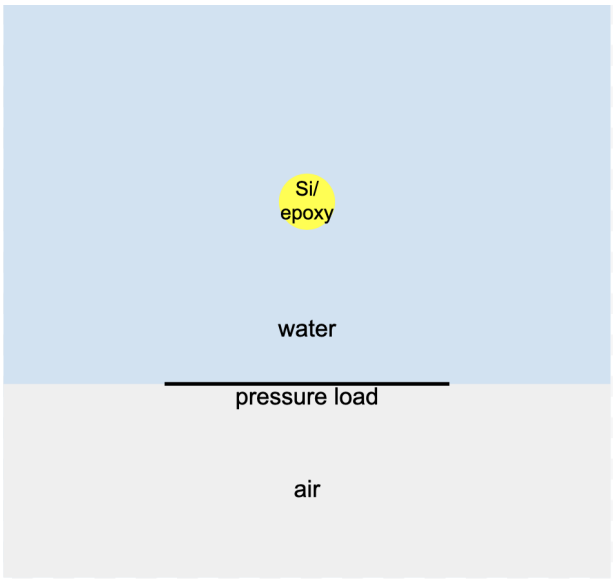


Figure 6: Axisymmetric model setup with pressure load transducer

With this model at a frequency of 1.6 mHz, results were more as we expected and higher pressure was becoming even more visible (Figure 7).

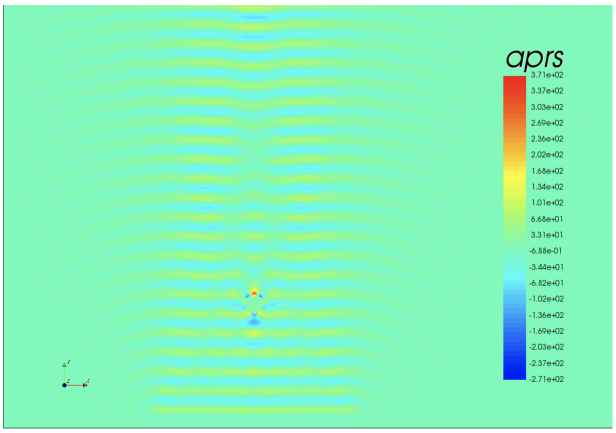


Figure 7: Results with bead at 1.6 mHz
Maximum pressure: 371 Pa

Figure 8 shows the same simulation without the bead for reference.

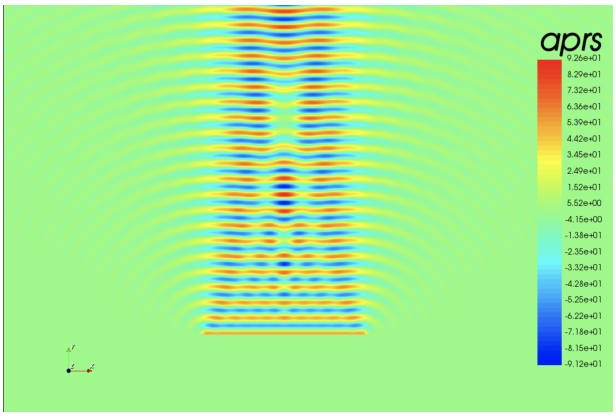
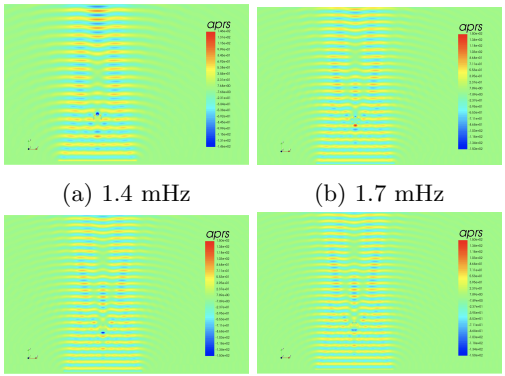


Figure 8: Same simulation, but without the bead (different color scale)
Maximum pressure: 92.6Pa

We ran it several times for a range of frequencies with an epoxy bead. Figure 9 shows some of the pressure visualizations (plotted using the script in Appendix A).

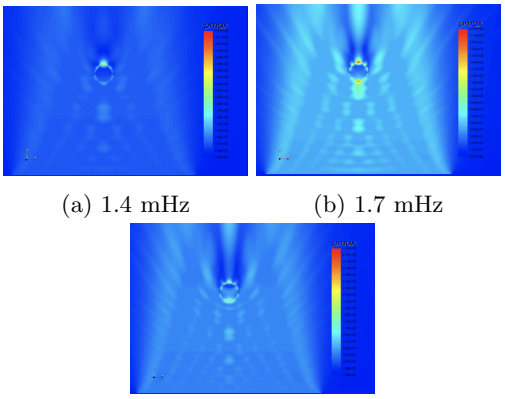


(a) 1.4 mHz (b) 1.7 mHz

(c) 1.8 mHz (d) 1.9 mHz

Figure 9: Broad frequency sweep with epoxy bead

We later re-ran all of these simulations, but recorded the maximum pressure over the entire simulation at each point as well (Figure 10; the maximum pressure can be seen by reading the topmost value on the bar at the right).



(a) 1.4 mHz (b) 1.7 mHz

(c) 1.9 mHz

Figure 10: Max pressure of broad frequency sweep with epoxy bead

In all of these runs, a spike in pressure amplitude is clearly visible at the interface between the bead and the water. To correlate the relationship between the frequency and the amplitude of these spikes, a python script was written to go through each max pressure output(the second set of images) and plot its maximum value. Running that over all the frequencies simulated yielded the graph in Figure 12.

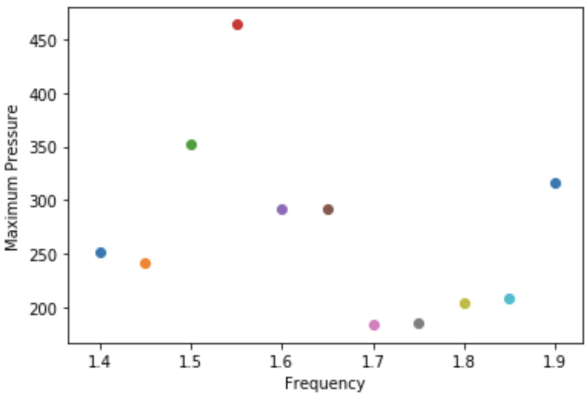


Figure 11: Broad sweep max pressure vs. frequency

The goal here was to identify the frequency range that produced the highest maximum pressure. To refine Figure 12, we ran several more simulations, this

time between 1.5 and 1.6 mHz, since there was a spike visible at 1.55 mHz (Figure 12).

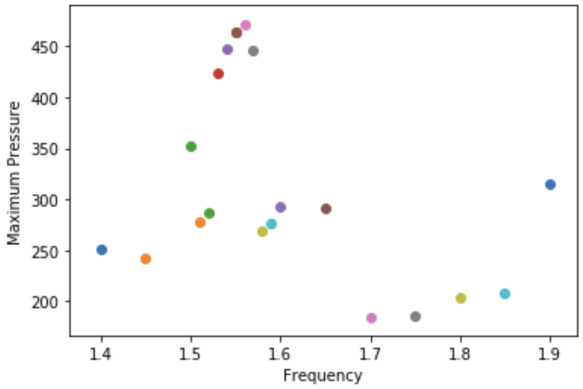


Figure 12: Fine sweep max pressure vs. frequency

Doing that confirmed that around 1.56 mHz gives the largest maximum pressure.

3.2 Off-Axis Model

One of the main goals for this project is that the patient will be able to treat themselves by holding an ultrasound transducer up against their neck. However, all of the previously shown simulations have assumed that the bead is perfectly in line with the center of the transducer. With human operation, this is very unlikely to occur in practice. As such, we decided to simulate having the bead at an offset on the x axis. This meant converting the model from an axisymmetric cylinder to a 2d simulation (Visible in Figure 13).

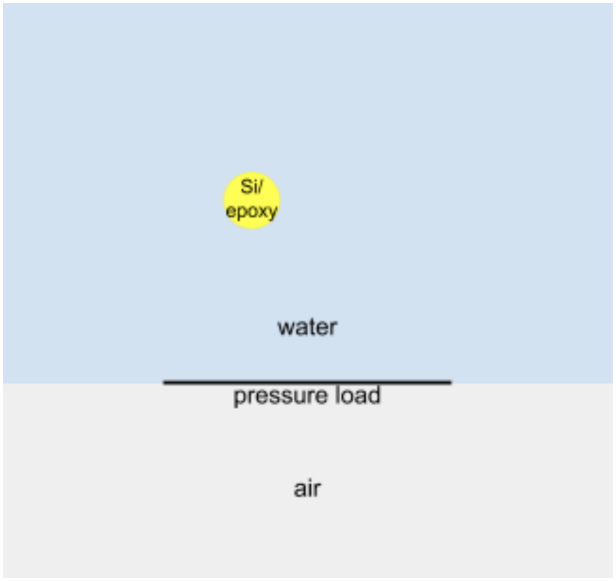


Figure 13: X-offset simulation setup

It looks fairly similar, the only difference being that we can now control the x-coordinate of the bead. We ran it with an offset of 0 (Figure 14), 1, 2, 3, and 4 (Figure 15) diameters (1mm) left, keeping the frequency at 1.6 mHz.

0 dia offset:

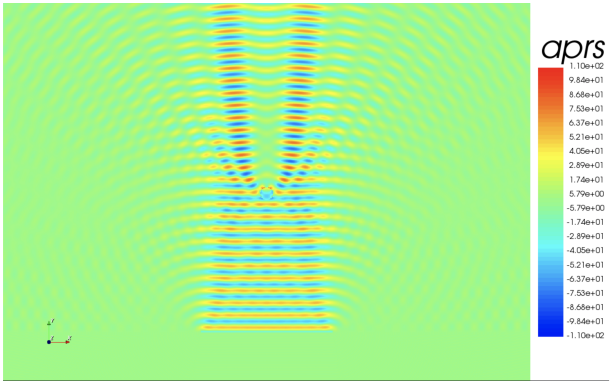


Figure 14: no offset

1-4 dia offset:

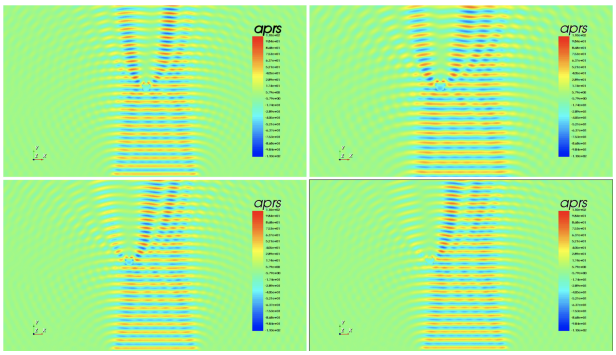


Figure 15: 1-4 diameter offset

Encouragingly, even with an offset, there are still areas of high pressure that occur along the center-line where the bead normally would have been.

3.3 Sensor Feedback Analysis

Although having the bead at an offset does still create a high pressure zone where it should be, it is not as dramatic as if it were in line. To address this, we set up a virtual 'probe' right in front of the transducer to simulate a closed loop feedback system. This probe records data values over time (e.g. pressure, y velocity, y displacement).

Figure 16 shows the y velocity recorded as the transducer sent 2 pulses at 1.6 mHz.

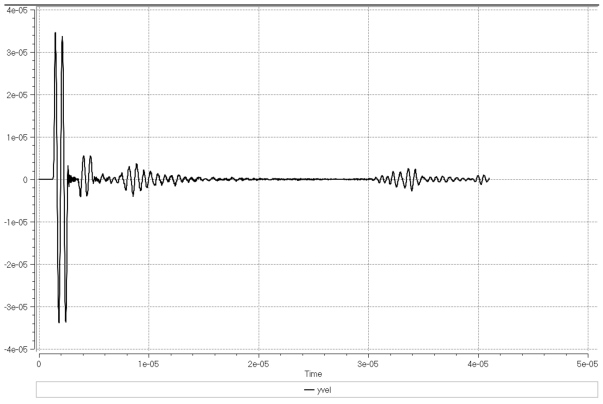


Figure 16: Probe feedback from 2 pulses

The initial pulses were fired after around 2ms, and the probe clearly picks up echoes at around 2 ms, 6 ms, and 32 ms. The simulation medium was water, where the speed of sound is 1457 m/s. The bead is 5mm away from the transducer and the probe is 2mm away from the transducer. This means the first two pulses would occur after the pulse traveled 2mm and 8mm. The calculated timings for the detected pulses would then occur after 1.37ms and 5.49ms, which matches the graph readings.

After establishing that pulses reflected can be detected and that they have a clear relationship to the bead's distance away, the next step was to, as the other simulations, test the effects that an x-offset would have on the readings. For this simulation (found in Appendix B), we chose to use a pair of passive piezoelectric pickup sensors in-line with the 'transducer' to more accurately simulate its real life application (Figure 17).

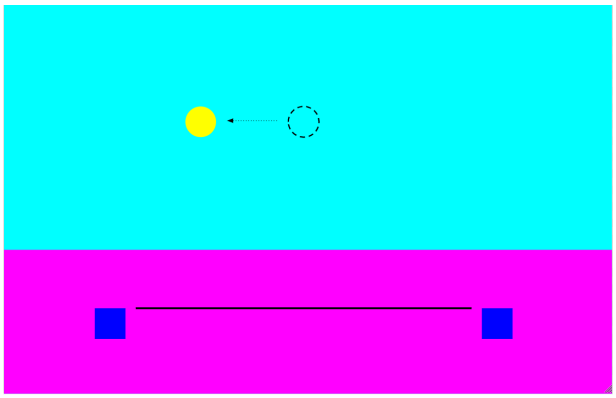


Figure 17: Stereo piezo mic off-axis setup

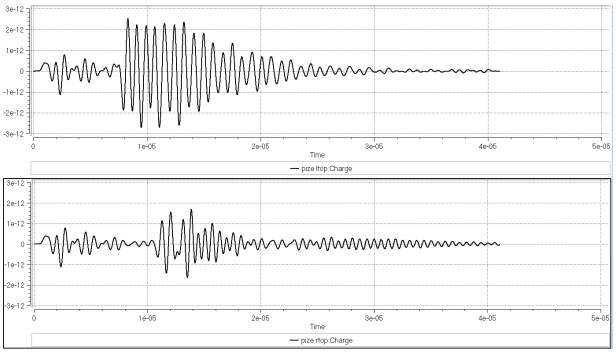


Figure 18: Stereo sensor feedback with 3mm x offset, bead 5mm away from transducer, at 1.6mHz

The reflected pulse in this setup was a lot more distinguishable (Figure 18). We ran the simulation several times with varying offsets and plotted the time difference in the reflected signal between the sensors over the offset (Figure 19).

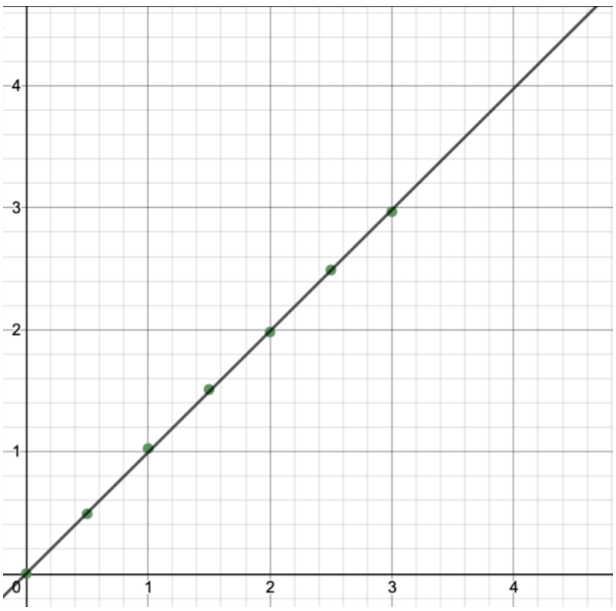


Figure 19: Peak detection time difference vs. bead offset (the units are in $sec * 10^{-6}$; best fit line: $y = 0.994593x$, $R^2 = 0.9998$

The strong linear correlation shows that with a stereo sensor setup such as this one, relative localization of the implanted bead is possible to a reasonable degree of accuracy. This holds across different frequencies and distances away.

4 Summary

This work has shown that the production of ultrasound-induced high pressure zones behind a solid bead via diffraction, necessary for stimulating the Vagus nerves, is possible. It has also shown that the setup is tolerant to slight deviations from perfect alignment. We have shown that, using stereo feedback microphones, it is possible to localize the bead via reflected pulse time measurements. This would allow feedback and guidance to be provided for optimal placement of the transducer, in line with the bead inside the neck. Finally, we have shown that an optimal frequency for creating pressure spikes around a bead in the body exists.

The maximum pressure with the bead is drastically higher than without. This shows that the existence of the bead and the diffraction effects it produces can amplify pressure at least by up to 4 times. We have also demonstrated that the pressure spike with the bead only exists locally. This means that there is no risk of interference with any other body parts in the vicinity. If we simply raised the intensity of the entire pulse, other parts could be affected, with undesirable side effects.

5 Conclusion/Future Work

Further research can be done by repeating some of these simulations in a virtual model of the bodily region in question, providing information more relevant to its real life application. Also, the simulations run to maximize the pressure-spike could be run with a more refined model to get more exact results. After that, actual experiments on animals could be performed to test these results in a real biological setting.

References

[1] M. S. George, H. A. Sackeim, A. Rush, L. B. Marangell, Z. Nahas, M. M. Husain, S. Lisanby, T. Burt, J. Goldman, and J. C. Ballenger, "Vagus nerve stimulation: a new tool for brain research and therapyâˆ—," *Biological Psychiatry*, vol. 47, no. 4, pp. 287 – 295, 2000.

[2] R. H. Howland, "Vagus nerve stimulation," *Current behavioral neuroscience reports*, vol. 1, pp. 64–73, 6 2014.

[3] J. Seladi-Schulman, "Vagus nerve: Anatomy and function, diagram, stimulation, conditions." http://www.healthline.com/human-body-maps/vagus-nerve, Aug 2018.

[4] E. J. Juan, R. GonzÃ¡lez, G. Albors, M. P. Ward, and P. Irazoqui, "Vagus nerve modulation using focused pulsed ultrasound: Potential applications and preliminary observations in a rat," *International Journal of Imaging Systems and Technology*, vol. 24, no. 1, pp. 67–71, 2014.

[5] K. M. Wasilczuk, K. C. Bayer, J. P. Somann, G. O. Albors, J. Sturgis, L. T. Lyle, J. P. Robinson, and P. P. Irazoqui, "Modulating the inflammatory reflex in rats using low-intensity focused ultrasound stimulation of the vagus nerve," *Ultrasound in Medicine & Biology*, vol. 45, no. 2, pp. 481 – 489, 2019.

[6] R. S. C. Cobbold, *Foundations of biomedical ultrasound*. Oxford University Press, 2007.

A Python Max Pressure Plotting Script

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jul 18 17:13:31 2020

@author: chinmay
"""
import numpy as np
import scipy.io as scio
import matplotlib.pyplot as plt


boxDim = 0.001457/15
water_size = 0.045  # length of water past transducer
water_width = 0.015     # radius of water
pzt_rad = 0.005
pzt_thk = 0.002
air_thk = 0.005
si_dim = 0.001

plt.figure()

files = np.array([i for i in np.arange(1.51, 1.59, 0.01)])

for i in range(0, files.shape[0]):
    #1.5-1.6mHz_51
    file = scio.loadmat('C:/Users/chinmay/Documents/Summer Project/results/epoxy 1.51-1.59mHz max/'
        + str(i+1) + '/results.mat')
    keys = list(file)
    prefix = str(keys[len(keys)-1]).split('_')
    prefix = prefix[0] + '_' + prefix[1] + '_'
    #print(prefix)
    aprs = file[prefix + 'pmax']

    shape = aprs.shape
    rows = shape[1]
    cols = shape[0]
    col = 0 #sweep entire model


    #plt.figure()

    ymax, ymin = 0.0, 0.0

    xs = np.array([(i*boxDim) for i in range(0,rows)])
    ys = np.array([aprs[2][i] for i in range(0, rows)])

    ymax = np.amax(ys)
    ymin = np.amin(ys)
```

```
    plt.scatter(files[i],np.max([ymax,-ymin]))
    print(i)
    print(np.max([ymax,-ymin]))

plt.title('')
plt.ylabel('Maximum Pressure')
plt.xlabel('Frequency')
plt.show()
plt.close()
```

B Onscale Script

```
c ***************************************************************************
c
c DESIGNER :  OnScale
c MODEL DESCRIPTION :   PZT 2D Axisymmtric Disc
c VERSION : 1.0
c
c ***************************************************************************

c Set model title
titl OnScale PZT 2D Disc Example

c Use all cores
mp omp * *

c Do not save restart file
rest no

c ***************************************************************************
c MODEL INPUTS
c ***************************************************************************

c Geometry
symbx water_size = 2e-2  /* length of water past transducer
symbx water_width = 15e-3       /* radius of water
symbx pzt_rad = 5e-3 /* PZT Radius
symbx sensor_thk = 1e-3
symbx air_thk = 5e-3
symbx Si_dim = 1e-3          /* diameter of spherical silicon implant
symbx offset = 3e-3
symb Si_half_dim = $Si_dim / 2
symb Si_y_pos = $water_size / 2 - $Si_half_dim
symbx Si_dist = 5e-3
symb Si_center_y = $air_thk + $sensor_thk + $Si_dist + $Si_half_dim
symb x_pos = $water_width + $pzt_rad - $offset
```

```
symb si_pzt_x_dist = $pzt_rad - $Si_half_dim
symb extra_dist = $water_size - $Si_dim - $Si_dist

symbx sensor_x_dist = 0.5e-3
symbx sensor_width = 1e-3
symb past_sensor = $water_width - $sensor_width - $sensor_x_dist

symb vel = 1457
c Frequency and mesh
c symbx quo_val = 1
c symb freqint = $vel / $Si_dim / $quo_val
symbx freqint = 1.8e6 /* Frequency of Interest

symb freqdamp = $freqint /* Set damping frequency to frequency of operation
symb wavelength = $vel / $freqint /* Calculate Smallest Wavelength to discretised
symb nepw = 30 /* Number of elements per wavelength
symb box = $wavelength / $nepw /* Calculate Element size for Meshing

symb transit_time = $water_size / $vel
c symb func_cycles = $transit_time * $freqint * 3
symb func_cycles = 2

c ***************************************************************************
c MESHING
c ***************************************************************************

c Create X-Y-Z Keypoints to accurately model dimensions
c symb #keycord x 1 0.0 $water_width $pzt_rad $pzt_rad $water_width
symb #keycord x 1 0.0 $past_sensor $sensor_width $sensor_x_dist $pzt_rad $pzt_rad $sensor_x_dist
    $sensor_width $past_sensor
symb #get { idx } rootmax x /* Store last keypoint indice in variable 'idx'

symb #keycord y 1 0.0 $air_thk $sensor_thk $Si_dist $Si_dim $extra_dist
symb #get { jdx } rootmax y /* Store last keypoint indice in variable 'jdx'

c Create I-J-K Keypoints
symb #keyindx i 1 $idx 1 $box 1 /* Calculate number of nodes required in the I direction
symb #keyindx j 1 $jdx 1 $box 1 /* Calculate number of nodes required in the J direction

symb indgrd = $i$idx /* Last node number in I direction
symb jndgrd = $j$jdx /* Last node number in J direction

c Create Grid
grid $indgrd $jndgrd pstn /* Generate Grid of Nodes and apply Axisymmetric condition to model grid

c Map Physical Geometry to Grid of Nodes
geom keypnt $idx $jdx /* Tie in all XY Keypoints (Physical Dimensions) to IJ Keypoints(Nodes)

c ***************************************************************************
c  MATERIAL PROPERTIES
```

```
c ********************************************************************************

symb epvacm = 8.854e-12               /* dielectric constant for vacumn
symb freqdamp = 1.e6 if noexist       /* specified frequency for damping model
symb rmu0 = 1.2566e-6

symb #msg 5
*****************************************************
Damping matched at $freqdamp Hz
Redefine variable 'freqdamp' if device centre frequency
varies significantly from this value
*****************************************************

c ------------------------------------------------------------------------
c    Now define the axis transformation - only posx used in this file
c ------------------------------------------------------------------------

axis
     form vctr
     defn posx car2 0. 0. 0.    1.  0.  0.   0. 1. 0.
     defn negx car2 0. 0. 0.   -1.  0.  0.   0. 1. 0.
     defn posy car2 0. 0. 0.    0.  1.  0.   0. 0. 1.
     defn negy car2 0. 0. 0.    0. -1.  0.   0. 0. 1.
     defn posz car2 0. 0. 0.    0.  0.  1.   1. 0. 0.
     defn negz car2 0. 0. 0.    0.  0. -1.   1. 0. 0.
     end

c ------------------------------------------------------------------------
c    Input material properties to program
c ------------------------------------------------------------------------

matr
c ---------------------------------------------------------------
c type : PIEZO :
c name : pmt3 :
c desc : CTS 3203HD :
c ---------------------------------------------------------------


c    define baseline dielectric coefficients

symb epxx = 1305.8                  /* dielectric constant (constant strain)
symb epyy = 1305.8                  /* dielectric constant (constant strain)
symb epzz = 1200.2                  /* dielectric constant (constant strain)
symb rho  = 7820                   /* density
symb qdmp = 90                     /* Mechanical Q at 1e6
symb qsdmp = 90                     /* Mechanical Q at 1e6
symb freqdamp = 1.e6 if noexist
symb freqloss = 1e+06 if noexist                    /* Frequency that loss was measured at
```

```
c    define baseline stiffness coefficients

symb c11 = 1.3745e+11                  /* stiffness constant (constant electric field)
symb c12 = 8.79e+10                   /* stiffness constant
symb c13 = 9.23e+10                   /* stiffness constant
symb c14 = 0                  /* stiffness constant
symb c15 = 0                  /* stiffness constant
symb c16 = 0                  /* stiffness constant
symb c22 = 1.3745e+11                  /* stiffness constant (constant electric field)
symb c23 = 9.23e+10                   /* stiffness constant
symb c24 = 0                  /* stiffness constant
symb c25 = 0                  /* stiffness constant
symb c26 = 0                  /* stiffness constant
symb c33 = 1.2574e+11                  /* stiffness constant
symb c34 = 0                  /* stiffness constant
symb c35 = 0                  /* stiffness constant
symb c36 = 0                  /* stiffness constant
symb c44 = 2.2279e+10                  /* stiffness constant
symb c45 = 0                  /* stiffness constant
symb c46 = 0                  /* stiffness constant
symb c55 = 2.2279e+10                  /* stiffness constant
symb c56 = 0                  /* stiffness constant
symb c66 = 2.4779e+10

c    define baseline piezoelectric coupling coefficients

symb ex1 = 0                   /* coupling constant
symb ex2 = 0                   /* coupling constant
symb ex3 = 0                   /* coupling constant
symb ex4 = 0                   /* coupling constant
symb ex5 = 16.054                   /* coupling constant
symb ex6 = 0                   /* coupling constant
symb ey1 = 0                   /* coupling constant
symb ey2 = 0                   /* coupling constant
symb ey3 = 0                   /* coupling constant
symb ey4 = 16.054                   /* coupling constant
symb ey5 = 0                   /* coupling constant
symb ey6 = 0                   /* coupling constant
symb ez1 = -9.44                  /* coupling constant
symb ez2 = -9.44                  /* coupling constant
symb ez3 = 22.495                  /* coupling constant
symb ez4 = 0                   /* coupling constant
symb ez5 = 0                   /* coupling constant
symb ez6 = 0                   /* coupling constant

c    scale material properties as specified above

symb aepxx = $epvacm * $epxx
symb aepyy = $epvacm * $epyy
symb aepzz = $epvacm * $epzz
```

```
     wvsp off
     type lean

     prop pmt3 $rho
     $c11     $c12     $c13     $c14    $c15    $c16    $c22
     $c23     $c24     $c25     $c26    $c33    $c34    $c35
     $c36     $c44     $c45     $c46    $c55    $c56    $c66

     elec pmt3 $aepxx $aepyy $aepzz

     piez pmt3 1 1 $ex1 1 2 $ex2 1 3 $ex3 1 4 $ex4 1 5 $ex5 1 6 $ex6 &
       2 1 $ey1 2 2 $ey2 2 3 $ey3 2 4 $ey4 2 5 $ey5 2 6 $ey6 &
       3 1 $ez1 3 2 $ez2 3 3 $ez3 3 4 $ez4 3 5 $ez5 3 6 $ez6

     rdmp pmt3 $freqdamp q $qdmp $qsdmp $freqloss 1.0

     axis pmt3 posy   /* relate materials local system to global system

c ---------------------------------------------------------------
c type : FLUID :
c name : watr :
c desc : Water at 25C :
c ---------------------------------------------------------------


   wvsp on
   type elas
   prop watr 1000 1496 0 0.
   vdmp watr $freqdamp db 0.002 0 1e+06 2 0.01 0
   thrm watr 4181 0.61 0.61 0.61 1.0 0 0 27
   symb eps = 3
   symb aeps = $epvacm * $eps
   elec watr $aeps

c ---------------------------------------------------------------
c type : EPOXY :
c name : back20 :
c desc : Backing, tungsten loaded epoxy, 20%vf, 8.6 MRayl :
c ---------------------------------------------------------------


   wvsp on
   type elas
   prop back20 4800 1800 962 0.01
   vdmp back20 $freqdamp db 310 930 3e+07 1 0.01 1
   symb eps = 4
   symb aeps = $epvacm * $eps
   elec back20 $aeps
```

```
   elec void $epvacm
   end


c -------------------------------------------------------------
c                   Project Material List
c -------------------------------------------------------------


c ---------------------------------------------------------------------------
c Global variables used in all the material definitions
c ---------------------------------------------------------------------------
c
symb epvacm = 8.854e-12              /* dielectric constant for vacumn
symb freqdamp = 1.e6 if noexist      /* specified frequency for damping model
symb rmu0 = 1.2566e-6

symb #msg 5
*******************************************************
Damping matched at $freqdamp Hz
Redefine variable 'freqdamp' if device centre frequency
varies significantly from this value
*******************************************************


c
c ---------------------------------------------------------------------------
c    Now define the axis transformation - only posx used in this file
c ---------------------------------------------------------------------------

axis
     form vctr
     defn posx car2 0. 0. 0.    1.  0.  0.   0. 1. 0.
     defn negx car2 0. 0. 0.   -1.  0.  0.   0. 1. 0.
     defn posy car2 0. 0. 0.    0.  1.  0.   0. 0. 1.
     defn negy car2 0. 0. 0.    0. -1.  0.   0. 0. 1.
     defn posz car2 0. 0. 0.    0.  0.  1.   1. 0. 0.
     defn negz car2 0. 0. 0.    0.  0. -1.   1. 0. 0.
     end

c ---------------------------------------------------------------------------
c    Input material properties to program
c ---------------------------------------------------------------------------

matr

c -------------------------------------------------------------
c type : BIOLOGICAL :
```

```
c name : conn :
c desc : Connective tissue :
c -------------------------------------------------------------

wvsp on
type elas
prop conn 1100 1537 0 0.01
vdmp conn $freqdamp db 0.56 0 1e+06 1 0.01 1
thrm conn 3430 0.5 0.5 0.5 1.0 3840 38 37

c -------------------------------------------------------------
c type : BIOLOGICAL :
c name : tofu :
c desc : Tofu, tissue mimick :
c -------------------------------------------------------------

wvsp on
type elas
prop tofu 1059 1520 0 0.01
vdmp tofu $freqdamp db 0.75 0 1e+06 0.75 0.01 0.75

elec void $epvacm
end


c -------------------------------------------------------------
c                   Project Material List
c -------------------------------------------------------------


c ---------------------------------------------------------------------------
c Global variables used in all the material definitions
c ---------------------------------------------------------------------------
c
symb epvacm = 8.854e-12               /* dielectric constant for vacumn
symb freqdamp = 1.e6 if noexist       /* specified frequency for damping model
symb rmu0 = 1.2566e-6

symb #msg 5
*****************************************************
Damping matched at $freqdamp Hz
Redefine variable 'freqdamp' if device centre frequency
varies significantly from this value
*****************************************************


c
c ---------------------------------------------------------------------------
c    Now define the axis transformation - only posx used in this file
c ---------------------------------------------------------------------------
```

```
axis
     form vctr
     defn posx car2 0. 0. 0.    1.  0.  0.   0. 1. 0.
     defn negx car2 0. 0. 0.   -1.  0.  0.   0. 1. 0.
     defn posy car2 0. 0. 0.    0.  1.  0.   0. 0. 1.
     defn negy car2 0. 0. 0.    0. -1.  0.   0. 0. 1.
     defn posz car2 0. 0. 0.    0.  0.  1.   1. 0. 0.
     defn negz car2 0. 0. 0.    0.  0. -1.   1. 0. 0.
     end

c ------------------------------------------------------------------------
c    Input material properties to program
c ------------------------------------------------------------------------

matr

c ---------------------------------------------------------------
c type : GAS :
c name : air :
c desc : Air, room temp :
c ---------------------------------------------------------------

wvsp on
type elas
prop air 1.24 343 0 0.
c define baseline dielectric coefficients

symb epxx = 1
symb epyy = 1
symb epzz = 1

c scale material properties as specified above
symb aepxx = $epvacm * $epxx
symb aepyy = $epvacm * $epyy
symb aepzz = $epvacm * $epzz

elec air $aepxx $aepyy $aepzz


elec void $epvacm
end


c ---------------------------------------------------------------
c                   Project Material List
c ---------------------------------------------------------------


c ------------------------------------------------------------------------
```

```
c Global variables used in all the material definitions
c -------------------------------------------------------------------------
c
symb epvacm = 8.854e-12               /* dielectric constant for vacumn
symb freqdamp = 1.e6 if noexist       /* specified frequency for damping model
symb rmu0 = 1.2566e-6

symb #msg 5
*****************************************************
Damping matched at $freqdamp Hz
Redefine variable 'freqdamp' if device centre frequency
varies significantly from this value
*****************************************************


c
c -------------------------------------------------------------------------
c    Now define the axis transformation - only posx used in this file
c -------------------------------------------------------------------------

axis
     form vctr
     defn posx car2 0. 0. 0.    1.  0.  0.   0. 1. 0.
     defn negx car2 0. 0. 0.   -1.  0.  0.   0. 1. 0.
     defn posy car2 0. 0. 0.    0.  1.  0.   0. 0. 1.
     defn negy car2 0. 0. 0.    0. -1.  0.   0. 0. 1.
     defn posz car2 0. 0. 0.    0.  0.  1.   1. 0. 0.
     defn negz car2 0. 0. 0.    0.  0. -1.   1. 0. 0.
     end

c -------------------------------------------------------------------------
c    Input material properties to program
c -------------------------------------------------------------------------

matr

c ---------------------------------------------------------------
c type : MISC :
c name : si :
c desc : Silicon, generic :
c ---------------------------------------------------------------

wvsp on
type elas
prop si 2330 7526 4346 0.01
vdmp si $freqdamp db 0.1 0.3 1e+06 1 0.01 1
thrm si 702 124 124 124 1.0 0 0 27
c define baseline dielectric coefficients

symb epxx = 11.9
```

```
symb epyy = 11.9
symb epzz = 11.9

c scale material properties as specified above
symb aepxx = $epvacm * $epxx
symb aepyy = $epvacm * $epyy
symb aepzz = $epvacm * $epzz

elec si $aepxx $aepyy $aepzz


elec void $epvacm
end



c ---------------------------------------------------------------
c                   Project Material List
c ---------------------------------------------------------------


c ------------------------------------------------------------------------
c Global variables used in all the material definitions
c ------------------------------------------------------------------------
c
symb epvacm = 8.854e-12              /* dielectric constant for vacumn
symb freqdamp = 1.e6 if noexist      /* specified frequency for damping model
symb rmu0 = 1.2566e-6

symb #msg 5
*****************************************************
Damping matched at $freqdamp Hz
Redefine variable 'freqdamp' if device centre frequency
varies significantly from this value
*****************************************************


c
c ------------------------------------------------------------------------
c    Now define the axis transformation - only posx used in this file
c ------------------------------------------------------------------------

axis
     form vctr
     defn posx car2 0. 0. 0.    1.  0.  0.   0. 1. 0.
     defn negx car2 0. 0. 0.   -1.  0.  0.   0. 1. 0.
     defn posy car2 0. 0. 0.    0.  1.  0.   0. 0. 1.
     defn negy car2 0. 0. 0.    0. -1.  0.   0. 0. 1.
     defn posz car2 0. 0. 0.    0.  0.  1.   1. 0. 0.
     defn negz car2 0. 0. 0.    0.  0. -1.   1. 0. 0.
```

```
     end

c ------------------------------------------------------------------------
c    Input material properties to program
c ------------------------------------------------------------------------

matr

c ---------------------------------------------------------------
c type : MISC :
c name : sapph :
c desc : Sapphire, generic :
c ---------------------------------------------------------------

wvsp on
type elas
prop sapph 3990 11100 6040 0.01
vdmp sapph $freqdamp db 0.1 0.4 1e+06 1 0.01 1
thrm sapph 761 24 24 24 1.0 0 0 27
c define baseline dielectric coefficients

symb epxx = 9.3
symb epyy = 9.3
symb epzz = 9.3

c scale material properties as specified above
symb aepxx = $epvacm * $epxx
symb aepyy = $epvacm * $epyy
symb aepzz = $epvacm * $epzz

elec sapph $aepxx $aepyy $aepzz


elec void $epvacm
end


c ---------------------------------------------------------------
c                   Project Material List
c ---------------------------------------------------------------


c ------------------------------------------------------------------------
c Global variables used in all the material definitions
c ------------------------------------------------------------------------
c
symb epvacm = 8.854e-12              /* dielectric constant for vacumn
symb freqdamp = 1.e6 if noexist      /* specified frequency for damping model
symb rmu0 = 1.2566e-6
```

```
symb #msg 5
*****************************************************
Damping matched at $freqdamp Hz
Redefine variable 'freqdamp' if device centre frequency
varies significantly from this value
*****************************************************


c
c -------------------------------------------------------------------------
c    Now define the axis transformation - only posx used in this file
c -------------------------------------------------------------------------

axis
     form vctr
     defn posx car2 0. 0. 0.    1.  0.  0.   0. 1. 0.
     defn negx car2 0. 0. 0.   -1.  0.  0.   0. 1. 0.
     defn posy car2 0. 0. 0.    0.  1.  0.   0. 0. 1.
     defn negy car2 0. 0. 0.    0. -1.  0.   0. 0. 1.
     defn posz car2 0. 0. 0.    0.  0.  1.   1. 0. 0.
     defn negz car2 0. 0. 0.    0.  0. -1.   1. 0. 0.
     end

c -------------------------------------------------------------------------
c    Input material properties to program
c -------------------------------------------------------------------------

matr

c ---------------------------------------------------------------
c type : EPOXY :
c name : arald1 :
c desc : Araldite MY750/HY956EN :
c ---------------------------------------------------------------

wvsp on
type elas
prop arald1 1146 2658 1237 0.01
vdmp arald1 $freqdamp db 4 12.59 1e+06 1 0.01 1
c define baseline dielectric coefficients

symb epxx = 3.5
symb epyy = 3.5
symb epzz = 3.5

c scale material properties as specified above
symb aepxx = $epvacm * $epxx
symb aepyy = $epvacm * $epyy
symb aepzz = $epvacm * $epzz
```

```
elec arald1 $aepxx $aepyy $aepzz


elec void $epvacm
end

c Allocate materials to grid
site
regn air
regn watr $i1 $i9 $j3 $j6               /* watr, conn
circle2d arald1 $x_pos $Si_center_y $Si_half_dim   /*  si, arald1
c regn si $i1 $i4 $j4 $j5
regn pmt3 $i2 $i3 $j2 $j3
regn pmt3 $i7 $i8 $j2 $j3
end

c Plot Model
grph
nvew 2 1 /* Set up 2 views
plot matr i $i1 $i9 j $j2 $j3 /* Plot Model
plot matr /* Replot Model
end
term /* Pause model

c ***************************************************************************
c  BOUNDARY CONDITIONS & LOADING
c ***************************************************************************

c External boundary conditions
boun
side xmin absr /* Assign Symmetry condition XMIN side of model
side xmax absr /* Assign Absorbing boundary condition (Infinite Medium)
side ymin absr /* to XMAX YMIN and YMAX side
side ymax absr
end

c Define time varying function
c func wvlt $freqint 1
func sine $freqint 50.0 0. $func_cycles 0. 0.0



piez
wndo $i2 $i3 $j2 $j3 /* Define electric window to PZT material - minimise for faster model run time

defn ltop  /* Define electrode - name 'top'
node $i2 $i3 $j3 $j3 /* Assign nodes for top electrode

defn lbot /* Define electrode - name 'bot'
node $i2 $i3 $j2 $j2 /* Assign nodes for bottom electrode
```

```
/* Electrical Boundary Condition

bc ltop on /* Apply voltage condition to 'top' electrode and assign driving signal 'func'
bc lbot grnd /* Apply ground condition to 'bot' electrode

c slvr pard /* Use PARDISO solver - our new faster electrostatic solver for PIEZO simulations
end

piez
wndo $i7 $i8 $j2 $j3 /* Define electric window to PZT material - minimise for faster model run time

defn rtop  /* Define electrode - name 'top'
node $i7 $i8 $j3 $j3 /* Assign nodes for top electrode

defn rbot /* Define electrode - name 'bot'
node $i7 $i8 $j2 $j2 /* Assign nodes for bottom electrode

/* Electrical Boundary Condition

bc rtop on /* Apply voltage condition to 'top' electrode and assign driving signal 'func'
bc rbot grnd /* Apply ground condition to 'bot' electrode

c slvr pard /* Use PARDISO solver - our new faster electrostatic solver for PIEZO simulations
end

plod
  pdef pld1 func
  vctr vct1 0 1 0
  sdef pld1 vct1 $i4 $i6 $j3 $j3
  end

c ***************************************************************************
c  OUTPUTS
c ***************************************************************************

c Request calculation of additional arrays
calc
pres acoustic  /* Calculate acoustic pressure
velm
disp    /* Calculate X and Y displacements
max aprs none pmax  /* Calculate maximum pressure field
max ydsp none ydmax  /* Calculate maximum pressure field
end

c Request time domain outputs - Graphs of Data vs Time
pout
hist func /* Driving Function stored in Time History 1
histname electrode vq ltop /* Driving Function stored in Time History 2 (V Top) 3 (Q Top) 4 (V Bot) 5 (Q Bot)
histname electrode vq lbot
```

```
histname electrode vq rtop
histname electrode vq lbot
hist ydsp $i1 $i1 1 $j2 $j2 1     /* y-displacement at node on the top electrode - array '6'
end

shap /* Calculate Harmonic Behaviour at Particular Frequencies
  data xdsp /* Request data to be calculated for harmonic analysis
  data ydsp
  data aprs
freq $freqint /* Request frequency for harmonic analysis
end

c ***********************************************************************
c  PROCESS MODEL
c ***********************************************************************

c Process model
prcs      /* Check models, sets up required data arrays, calculates stable timestep for Solver to Run model

c ***********************************************************************
c   MODEL EXECUTION
c ***********************************************************************

c Runtime calculations
symb #get { step } timestep /* Extract calculated timestep from PRCS command
symb simtime = $transit_time * 3/* Simulation time - 3x transit time
symb nexec = $simtime / $step /* Determine how many timesteps to run full model
symb nloops = 30 /* Plotting snapshots during model execution
symb nexec2 = $nexec / $nloops /* Partition full simulation into smaller segments for Plotting

c Set up plotting
grph
arrow off /* Turn off arrows
line off /* Turn off mesh lines
set imag mpeg  /* Set image capture for movie file generation
nvew 3 1 /* Set up 3 plotting windows in layout number 3
end

c Create Procedure - Code that can be executed 'x' number of times
proc plot save /* Name procedure 'plot' and save the code

exec $nexec2 /* Run model partially

grph
plot aprs /* Plot calculated data array
plot 1 /* Plot Time History 1 - Drive Function
  plot 3  /* Plot Time History 3 - Charge on Top electrode
imag /* Snapshot of Graphics window for Movie generation
end
```

```
end$ proc /* End of Procedure Code

c Run model by calling 'plot' procedure 'nloops' times
proc plot $nloops

term /* Pause model - Allow user to call procedure 'plot' from console window to run model for longer

c ***********************************************************************
c ADDITIONAL OUTPUTS
c ***********************************************************************

data
file out 'model.flxdato'
out shap/all
out modl
out aprs
out ydsp
out yvel
out ydmax
end

data
form out matlab d1
file out results.mat
out ydsp
out aprs
out yvel
end

c Save symbols to file for later use
symb #get { labl } jobname /* find name of run
symb #save '$labl.symb'/* save in symb file

c End of input file
stop
```