



## PZFLEX BEGINNER'S GUIDE

A beginner's guide to using PZFlex: understanding basic ultrasound theory and analysing different model scenarios

## Contents

Table of Figures .....	5
Modelling/Coding Rules and Tips .....	7
Section 1 – Basic Theory .....	8
1.1 Ultrasonic Waves .....	8
1.2 Piezoelectric Effect.....	9
1.3 Piezoelectric Materials.....	9
1.4 Piezocomposites .....	10
1.5 Matching Layers .....	11
1.6 Backing Block .....	11
1.7 Resonance in Piezocomposite Devices .....	11
Section 2 – Starting to Model .....	14
2.1 Finite Element Analysis (FEA).....	14
2.2 General Code Layout.....	16
2.3 A Simple Model – Planewave Propagation .....	17
2.3.1 Physical Dimensions.....	17
2.3.2 X-Y-Z Co-ordinate System .....	18
2.3.3 Element Size.....	18
2.3.4 I-J-K Co-ordinate system .....	19
2.3.5 GEOM Definition .....	19
2.3.6 Material Definition.....	19
2.3.7 Material Site Definition .....	20
2.3.8 Boundary Conditions.....	20
2.3.9 Driving Function .....	21
2.3.10 Property Calculation .....	21
2.3.11 Pressure Load Application.....	21
2.3.12 Time Histories .....	21
2.3.13 Execution Command .....	22
2.3.14 Summary .....	22
2.4 Changing the Target.....	23
2.4.1 X-Y-Z Co-ordinate System .....	23

2.4.2 I-J-K Co-ordinate system .....	24
2.4.3 GEOM Definition .....	24
2.4.4 Material Site Definition .....	25
2.4.5 Boundary Conditions.....	25
2.4.6 Pressure Load Application.....	25
2.4.7 Summary .....	25
2.5 A Circular Object .....	26
2.6 Spherical Waves .....	27
2.7 Angled Boundaries .....	28
2.8 Piston Source and Beam Profiles .....	30
2.9 Effect of Varying Piston Size.....	32
2.10 Transient Wave Analysis vs. Continuous Wave Analysis .....	36
2.11 Piezoceramic Disk (Hard-code) .....	40
2.11.1 Electric Field Application.....	40
Section 3 – Wizards.....	42
3.1 Introduction .....	42
3.2 Piezoceramic Disk .....	43
3.3 2D and 3D Piezocomposite Structure .....	47
3.3.1 2D vs. 3D .....	52
3.4 Tonpilz Transducer.....	54
3.5 2D Array Element Transducers .....	56
Section 4 – Phased Delay Laws .....	58
4.1 Stainless Steel Block.....	58
4.2 Wedge and Stainless Steel Block .....	60
4.2.1 Batch File.....	62
Section 5 – ‘How to’ s.....	63
5.1 Arrays .....	63
5.1.1 Creating an array.....	63
5.1.2 Setting values in an array.....	63
5.1.3 Returning values from an array .....	63
5.1.4 Creating Files to output data .....	64

5.2 Do Loops .....	65
5.3 If Else Structure .....	65
5.4 Using the FUNC command .....	66
5.4.1 Introduction .....	66
5.4.2 Continuous Sine Wave .....	67
5.4.3 Sine Impulse .....	68
5.4.4 Three Cycle Burst with Phase Inversion .....	69
5.4.5 Ramp-Up 3 Cycle Burst with Time Delay .....	70
5.4.6 Blackman Harris .....	71
5.4.7 Wavelet .....	72
5.4.8 Chirp .....	73
5.5 Using the PLOD command .....	74
5.5.1 Introduction .....	74
5.5.2 Wave Applied from Sides of Model .....	75
5.5.3 Wave Applied from different side of Model .....	77
5.5.4 Generating a Spherical Wave .....	78
5.5.5 Using SPHR .....	79
5.5.6 Using CYLN .....	81
References .....	83

## **Table of Figures**

Figure 1: Visual representation of Longitudinal and Shear waves.....	8
Figure 2: Diagram of aligning dielectric moments (poling) [3].....	9
Figure 3: Model of a plane wave propagating through water and interacting with a boundary.....	17
Figure 4: Marking out key points on the model.....	18
Figure 5: Sketch of boundary setup.....	20
Figure 6: Planwave interacting with a square void in water .....	23
Figure 7: Key points marked on each axis of the model.....	23
Figure 8: Simplified Model in Figure 6.....	24
Figure 9: Spherical wave model .....	27
Figure 10: A wave reflecting off an angled boundary.....	28
Figure 11: Model of a piston generating constant pressure wave .....	30
Figure 12: A quarter wavelength radius .....	33
Figure 13: A half wavelength radius .....	33
Figure 14: A wavelength radius .....	34
Figure 15: Double wavelength radius .....	34
Figure 16: Quadruple wavelength radius.....	35
Figure 17: Pressure field at 62.5 kHz.....	37
Figure 18: Pressure field at 125 kHz.....	37
Figure 19: Pressure field at 250 kHz.....	38
Figure 20: Pressure field at 500 kHz.....	38
Figure 21: Pressure field at 1 MHz .....	39
Figure 22: Ceramic disk analysis .....	40
Figure 23: Wizards pull down menu.....	42
Figure 24: Piezoceramic Disk Wizard 1st Interface.....	43
Figure 25: Post-processing options .....	44
Figure 26: Post-processing screen.....	45
Figure 27: Generating mode shape at phase of $110^\circ$ .....	46
Figure 28: Generating beam profile at 690 kHz.....	46
Figure 29: 2D Composite Device interface .....	47
Figure 30: 0.75, 1.5 and 3 mm pillar width (top to bottom) .....	48
Figure 31: Impedance profiles.....	49
Figure 32: Beam profiles from each device.....	51
Figure 33: 3D 2-2 Composite Interface .....	52
Figure 34: Impedance profiles from the 3D (left) and 2D model .....	53
Figure 35: Beam Profile from the 3D (left) and 2D model.....	53
Figure 36: Tonpilz Wizard interface .....	54
Figure 37: Tonpilz model with settings shown in Figure 35.....	55
Figure 38: 3D 1-3 composite transducer wizard interface.....	56
Figure 39: 3D 1-3 composite model running.....	57

Figure 40: Model of a steel block with delay laws applied to transducer.....	58
Figure 41: Model of 16 elements firing into a wedge and block of material.....	60
Figure 42: Diagram of model with available information in darkest blue text.....	60
Figure 43: Showing created text file with stored data.....	64
Figure 44: Continuous input wave function .....	67
Figure 45: Fast Fourier Transform form input to show fundamental frequency of 1MHz.....	67
Figure 46: Impulse function consisting of a single sine wave cycle.....	68
Figure 47: FFT of the sine wave impulse .....	68
Figure 48: 3 cycles of a sine wave with a 180 degree phase shift.....	69
Figure 49: FFT of the 3 cycle burst in Figure 47.....	69
Figure 50: waveform showing ramp-up of sine wave and time delay in use.....	70
Figure 51: FFT of ramp-up SINE function .....	70
Figure 52: Blackman Harris input signal .....	71
Figure 53: Frequency content of the Blackman Harris signal .....	71
Figure 54: Wavelet waveform.....	72
Figure 55: FFT of wavelet.....	72
Figure 56: Up-chirp signal beginning at 500 kHz to 1 MHz.....	73
Figure 57: FFT of the up-chirp in Figure 55 .....	73
Figure 58: Pressure loaded into the left hand side of the model .....	75
Figure 59: Pressure wave applied to right side of model, propagating in the -x direction .....	77
Figure 60: Generating a Spherical wave .....	78
Figure 61: Generating a pressure load to the declared spherical boundary using SPHR.....	79
Figure 62: Applying pressure load using CYLN to 'pac-man' boundary .....	81
Figure 63: The cylinder oriented along the Y-axis.....	82

## Modelling/Coding Rules and Tips

1. LOOK in the manual!
2. Remember spaces between symbols, operands and parenthesis
3. Integer Variables begin with letters I – N
4. Use symmetry where possible
5. Look for patterns in certain areas of code i.e. in IJK co-ordinates section
6. Material poling direction needs to be altered in the material file – use “arrow pole” in GRPH command to show current poling direction
7. Use meaningful variable names
8. Comment as much as possible
9. Array index begins at 1 not 0 – used when accessing data in arrays
10. Errors in code show up in the Print File as well as variable values – Learn to use the print file to debug your code
11. Math functions such as sin, cos and tan all operate in RADIANS
12. Mode Shape Movies may look odd due to the displacement scaling – adjust accordingly

## Section 1 – Basic Theory

### 1.1 Ultrasonic Waves

Acoustics can be defined “as the generation, transmission and reception of energy in the form of vibrational energy in matter” [1]. Sound waves travelling through the air are a simple example of acoustics. Ultrasonic waves are the same as all other acoustic waves, however, the frequency determines it as ultrasound: acoustic waves with frequencies above 20 kHz are defined as ultrasound. Ultrasound is an extremely useful due its non-invasive and non-destructive nature. The ability to penetrate through almost any material makes it ideal for detection/inspection/monitoring applications.

The two fundamental wave types that will be encountered are: longitudinal and shear. For longitudinal waves, particles oscillate parallel to the direction of the propagating wave. For shear waves, particle displacement is perpendicular to the direction of wave propagation. Both wave types are illustrated in Figure 1.

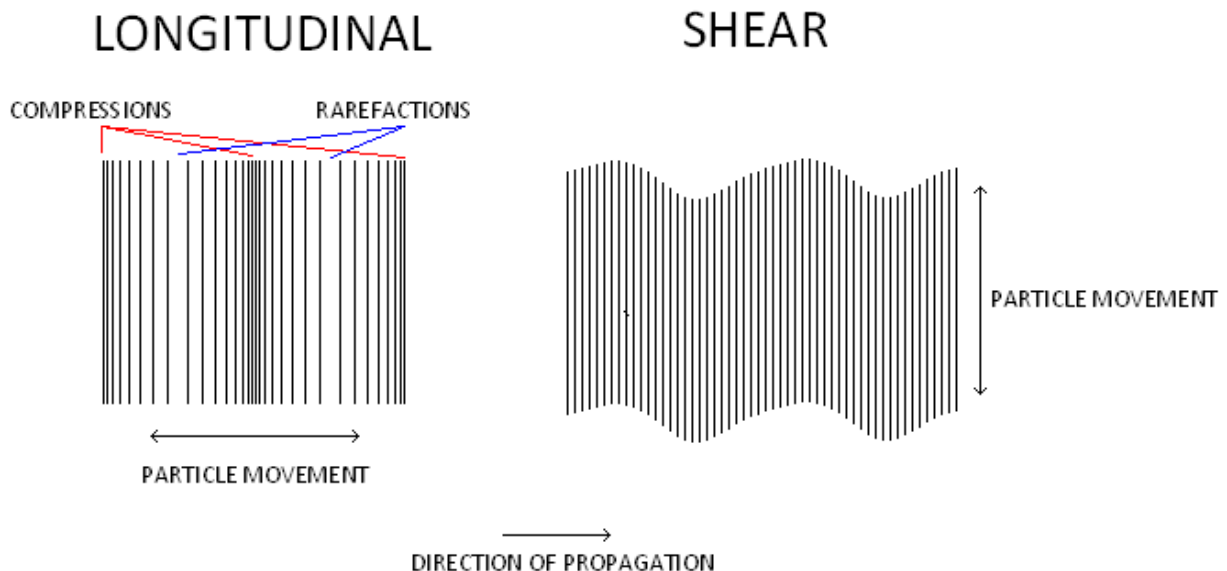


Figure 1: Visual representation of Longitudinal and Shear waves

Longitudinal waves can travel in solids, liquids and gases generating regions of high pressure and density called compressions and low pressure and density regions called rarefactions. Shear waves are only found in solids and not in liquids or gases due to their low viscosity [2]. The velocity of waves varies with certain material properties such as density. Pressure and temperature are other factors which also have an effect on the speed of waves in media.



## 1.2 Piezoelectric Effect

A piezoelectric material under strain (mechanical deformation) will generate electrical charge at the surface of the material. *Conversely* an electric field applied across a piezoelectric device will undergo mechanical deformation

In man-made ferroelectric materials, the internal dielectric moments are randomly aligned therefore no overall polarisation exists for the material. The ferroelectric material, however, subjected to a constant electric field and heated to a certain temperature (Curie temperature) will polarise the material and will remain permanently poled as the material is cooled. This process is known as **poling**. Poling will ensure the piezoelectric material will be anisotropic

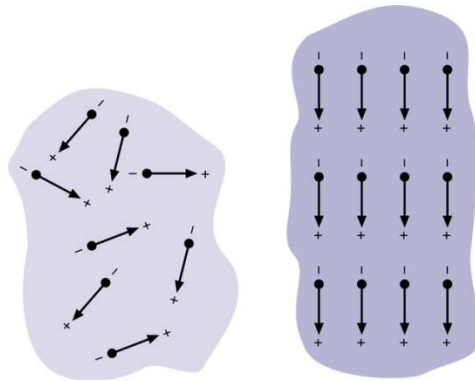


Figure 2: Diagram of aligning dielectric moments (poling) [3]

## 1.3 Piezoelectric Materials

The most common piezoelectric materials used today are **piezoceramics**. There are several characteristics making them suitable for a range of ultrasonic transducer applications which include:

- High specific acoustic impedance – direct bearing upon the transmission and reflection energy at the material boundaries (good coupling for metals such as copper, 42 MRayl, and steel, 45 MRayl, but poor for low impedance media such as water, 1.5MRayl, and air, 434 Rayl)
- High Electromechanical coupling coefficients – from 0.05 to 0.7
- High permittivity ( $\epsilon_r$ , ranging from 140 - 1750), to reduce transducer input impedance and allow good electrical matching
- A wide range of lateral coupling coefficients ranging from  $-10 \times 10^{-12} \text{ CN}^{-1}$  to  $-274 \times 10^{-12} \text{ CN}^{-1}$  for common ceramics.

Designers must choose ceramic carefully for each individual task.

- Common Ceramics
  - Lead Zirconate Titanate Family (PZT)
    - Variation of zirconate/titanium ratio and the addition of impurities
    - Extremely high electromechanical coupling factors ( $\leq 0.7$ ) allowing construction of high efficiency devices.

- PZT-4 Family – extremely low losses whilst maintaining large mechanical displacements. Applications – Sonar
- PZT-5 Family – greater thickness mechanical displacement properties at the expense of increased lateral coupling and increased losses. Applications – Medical Imaging and Non-Destructive Testing
- Modified Lead Titanate Family (MPT) and Lead Metaniobate (PMN)
  - Both low piezoelectric strain constants but have the advantage of extremely low lateral coupling coefficients, greatly reducing cross coupling in transducers utilising these materials
  - Low relative permittivities
  - PMN has a very high Curie temperature and mechanical damping making it appropriate for high temperature and wideband applications. Useful for array transducers with no kerf.
- Polyvinylidene fluoride (PVDF)
  - Produced as 25 to 110  $\mu\text{m}$  thin flexible sheets with very low specific acoustic impedance (4 MRayl), allowing higher transmission coefficients to material such as water and air. Unfortunately it has a low electrochemical coupling coefficient (0.2) and a very low relative permittivity making a poor transmitter of ultrasound. Useful as a hydrophone.

There are continuous improvements in the characteristics of piezoelectric materials. Limitations exist when manufacturing and temperature dependency for certain materials. Piezocomposite materials were developed in a desire to create the material that had most of the advantages of piezoceramics (high electromechanical coupling coefficients, high permittivity) and with the advantages of polymers (lower acoustic impedance, low lateral coupling).

## 1.4 Piezocomposites

Piezocomposites are constructed from two fundamental phases: the active piezoelectric ceramic and the passive (usually polymer) phase. Polymeric materials have a number of characteristics that make them a useful component of piezocomposites: low specific acoustic impedance (2 to 4 MRayl) to allow better matching to low acoustic impedance media; high mechanical losses to reduce the effect of lateral resonances and low relative permittivity ( $\epsilon_r \approx 4$ )

By combining the ceramic polymer into one composite material, many of the material properties can be altered. By variation of the volume of the ceramic within the overall composite volume (referred to as Volume Fraction, VF) specific acoustic impedance can be made lower than that of pure ceramic thus greater acoustic transmission energy to low impedance media. Relative permittivity will be reduced from that of pure ceramic, however, it will still be higher than the PVDF. The thickness mode electromechanical coupling coefficient can exceed that of the original ceramic too due to favourable pillar aspect ratio. Lateral resonances are also reduced due to the mechanically lossy polymer while bandwidth is increased. Thus, we have a device that should have high electromechanical coupling

efficiency, high relative permittivity and a specific acoustic impedance that can be tailored to the application.

The advantages of Piezocomposites are clear but there is the matter of the physical layout of the ceramic within in the polymer phase (or microstructure) to decide. This is often referred to as the *connectivity* of the composite.

“Dice and Fill” – ceramic placed in a mechanical saw and cut at regular spacing intervals to a specific depth. Depending on the type of composite (1-3, 2-2) ceramic will be orientated appropriately to fulfil the design. Once the ceramic is cut into pillars, a passive filler is chosen to fill the gaps of air and left to set. The composite is then returned to the saw to be cut at certain thicknesses to determine its frequency of operation.

“Injection moulding” is another manufacturing technique that allows finer control over the physical structure of the ceramic such as pillar shape and spacing. This technique lends itself well to mass production but requires high start-up cost.

### 1.5 Matching Layers

Large mismatch in acoustic impedance between transducer and load media will lead to acoustic energy reflected at the boundary and possibly phase inversion ( $180^\circ$ ). Matching layers are used to combat these problems to create a more efficient transducer maximising energy output from the front face of the transducer. Ideally the matching layer should be a quarter wavelength thick and as long as the acoustic impedance of the layer matches the geometric mean of the active and load media, then perfect transmission for one frequency can be achieved. Multiple matching layers can be used to achieve the more accurate acoustic impedance. The downside is manufacturing becomes more difficult.

### 1.6 Backing Block

Backing Block is to maximise **energy loss** at the rear face of the transducer by closely matching the acoustic impedance of the block to the transducer. This will theoretically provide perfect absorption. Typical methods of constructing backing blocks involve loading a hard setting polymer with powdered metal of high acoustic impedance (tungsten). As more energy is absorbed at the rear face of the transducer, signal ring down time decreases thus increasing bandwidth of the device. This increase in bandwidth results in a decrease in sensitivity so backing block must be designed carefully to provide the best solution.

### 1.7 Resonance in Piezocomposite Devices

**Thickness Mode** in ultrasonic transducers is usually the dominant mode of any present, occurring due to the finite thickness of the device and its wavelength equal to twice the thickness of the transducer.

The frequency of this mode is dependent upon the thickness of the device the velocity of sound in the thickness direction.

$$a_T = \frac{n\lambda_T}{2} \text{ \& } f_T = \frac{nv_T}{2a_T}$$

Where:

$a_T$  – thickness of transducer &  $n = 1, 3, 5, \dots$

$\lambda_T$  – wavelength in the thickness direction

$v_T$  – particle velocity in the thickness direction

$f_T$  – frequency of the thickness mode

The equation assumes that the device has higher acoustic impedance than the surrounding media therefore, only odd harmonics of this mode can be sustained.

Any resonant mode is characterised by the 2 distinct points on an electrical impedance profile:

#### **The electrical resonant frequency ( $f_n$ )**

- Associated with the point of the lowest impedance on the impedance plot
- Also when the frequency at which the impedance phase crosses from  $-90^\circ$  to  $+90^\circ$
- Generally frequency at which surface displacement of the transducer is the greatest thus used for transmission.

#### **The mechanical resonant frequency ( $f_m$ )**

- Determined by the impedance maxima occurring at a slightly higher frequency than  $f_n$
- Also when the impedance phase shifts from  $+90^\circ$  to  $-90^\circ$

Non-thickness mode resonances can also have significant affect on transducer behaviour so must be considered to when modelling transducer.

**Width Mode:** occurs due to finite width of the transducer which travels perpendicular to the thickness mode. The frequency of this mode is dependent on width of transducer and speed of sound in the width direction. If the width varies in the 2 axes then they will be more than one width mode

The remaining modes depend on the device microstructure:

#### **Inter-pillar Resonances**

- Occurs in 3-1 composite and arise due to standing wave patterns within a periodic composite lattice
- Lateral motion in the axes perpendicular to the thickness mode gives rise to shear waves that may give rise to standing wave patterns, coupling strongly into the composite structure.

### **Intra-pillar Resonances**

- Standing waves set up within a ceramic pillar due to the waves reflected at the ceramic/polymer boundary
- High velocity of sound in ceramic and low AR requirements set by the inter-pillar modes ensure that any intra-pillar modes are of a very high frequency. Consequently, it has very little impact on the device performance.

## **Section 2 – Starting to Model**

### **2.1 Finite Element Analysis (FEA)**

If any structure is divided into a sufficient number of small (finite) elements approximations to shape and stress can be readily made and only the magnitude remains to be found. The equations governing the behaviour of each element ensure that displacements are continuous across element boundaries and that all boundary conditions are satisfied. Large number of elements generally means a better approximation but is not always necessary.

Here are some terms used in FEA:

- Domain – the outside environments that can affect the model i.e. a water bath
- Nodes – discrete points within the domain forming the framework for the model Nodes connected together into a series of finite elements each having a set of governing equations and material properties.
  - Meshing – Division of the modelled volumes
  - Degrees of Freedom - temperature, pressure, voltage and displacement in each of the axes

***“The ‘art’ of FEA is to make the correct assumptions which will allow solution in a reasonable amount of time whilst maintaining accuracy.”***

The structure of FEA can be split into 3 sections:

#### **Pre-processing**

- Generating model using appropriate elements, boundary conditions , loads and selecting the desired solution type – the input stage of the analysis

#### **Solution**

- Solves the matrix that has been generated during pre-processing in an efficient manner

#### **Post processing**

- Analysis and presentation results

Different types of FEA:

#### **Static**

- Calculates steady state behaviour of a system when a constant load is applied

#### **Modal**

- Used to extract the natural frequencies and mode shapes of linear elastic structures

### Harmonic

- Used when a sinusoidal load of known amplitude and frequency is applied to a structure

### Transient

- Calculates the dynamic response of a structure that is subjected to a time dependent load

Each of these there can be sub-types

Information gained from FEA:

### Electrical impedance/frequency characteristics

- Simulated result can be compared with experimental plot obtained from impedance analyser
- input impedance reveals location of all resonant and anti-resonant modes
- shows if the transducer is 'unimodal' and if desired mode is coupled to other resonances
- Transducer efficiency measured by electromechanical coupling coefficient **k** – **electrical to acoustic energy conversion**

### Surface displacement profile (SDP)

- Magnitude and phase displacement in a plane of the transducer, often in front of the composite itself or the matching layer
- insight into actual transducer performance that may not be evident in impedance profiles
- Utilised to determine pressure beam profile from the transducer
- 3 figure of merit contained within SDP
  - Amplitude Dilation Quality
  - Phase Dilation Quality
  - Average Displacement

Dilation qualities express in a single number how uniform the measured aspect is across the transducer surface. They cannot give as much information as a full SDP but are a useful guide to transducer's behaviour.

## 2.2 General Code Layout

PZFlex provides a template input file for the model to be written in. The general structure is an excellent way to learn the different sections that are required to create a full model. The general structure is listed below:

- Physical Dimensions
  - Useful dimensions used to mark out key points
- X-Y-Z Co-ordinate System
  - Using physical dimensions to mark out the key points along the axes in use
- Element Size
  - Calculate size of element for model
- I-J-K Co-ordinate system
  - Mapping out grid of nodes based on physical dimensions and element size
  - GRID command to create grid of nodes
- GEOM Definition
  - Associates the I-J-K nodes with the corresponding physical dimensions
- Material Definition – MATR command or read in from file
- Material Site Definition – SITE command
  - Add in materials to region of nodes – REGN, CYLN, SPHR subcommands
- Boundary Conditions – BOUN command
  - Establish behaviour at the edges of the model – SIDE sub-command
- Driving Function – FUNC command
  - Declare type of wave stimulation
- Property Calculation – CALC command
  - Select the properties to calculate for model – PRES, DISP sub-commands
- Electric Field/ Pressure Load Application – PIEZ and PLOD command
- Time Histories – POUT command
  - Storing specific data to be analysed later
- Process Model (DEFAULT)
- Execution Command
  - Run Model for a set amount of time
- Plotting
  - Analysis of the model after model finishes executing

A simple model will be examined and each section will be discussed in more detail as well as displaying how code is written in PZFlex.



## 2.3 A Simple Model – Planewave Propagation

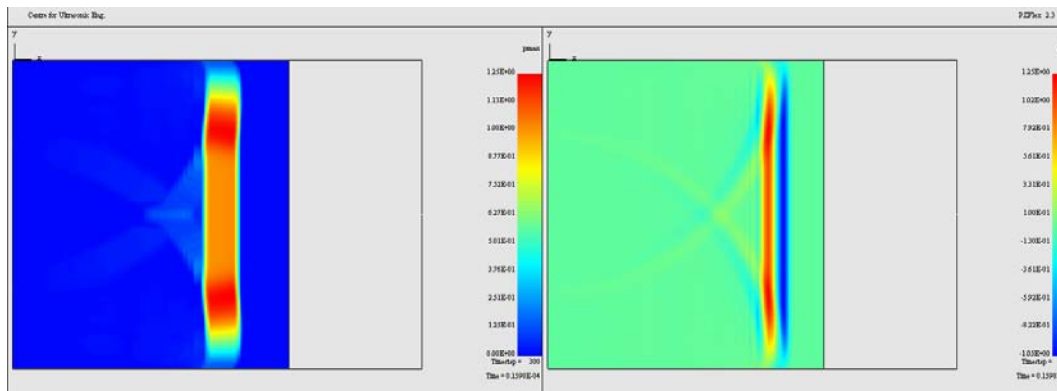


Figure 3: Model of a plane wave propagating through water and interacting with a boundary

This example will demonstrate how to set up a simple model of a plane wave propagating through water and reflecting off a (perfect) boundary. A plane wave is a constant-frequency wave whose wavefronts (surfaces of constant phase) are infinite parallel planes of constant amplitude normal to the phase velocity vector [4]. At the boundary, the wave will experience an 180° phase shift.

### 2.3.1 Physical Dimensions

Following the general code layout, the first step would be to include useful dimensions that would mark out important points used in the model.

```
c IMPORTANT** Integer Variables begin with letters I - N
synd areawdth = 40.e-3 /* All dimensions are CONSISTENTLY in Metres (m)
synd areahght = 15.e-3

synd targetcentre = 30.e-3 /* object dimensions within medium
synd targetwdth = 6.e-3
synd targethght = 6.e-3
```

Code 1: Physical dimension declaration

### 2.3.2 X-Y-Z Co-ordinate System

The models will always have an origin x1 and y1. More x and y variables are created to reference particular points, along each axis, in relation to the origin or previous reference points using the physical dimensions.

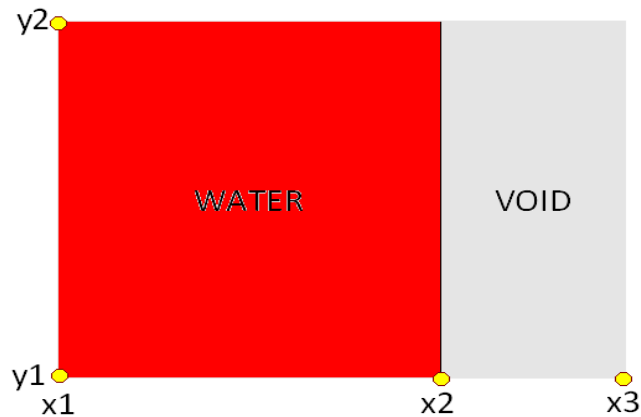


Figure 4: Marking out key points on the model

```

symb x1 = 0.0
symb x2 = $x1 + $targetcentre
symb x3 = $x1 + $areawidth

symb y1 = 0.0
symb y2 = $y1 + $areahght

```

Code 2: Associating x and y points with the physical dimensions

Code 2 shows the general code syntax for declaring the x and y variables: they should always relate back to a previous point whether it is the origin or a key point.

### 2.3.3 Element Size

The size of elements is then calculated to provide an accurate model. To calculate the element size, 3 pieces of information are needed: frequency of interest; wave velocity and the number of elements per wavelength. The frequency of interest is simply the operation frequency. The wave velocity is set to the lowest velocity that it will reach in the model which will be dependent on the materials used. The recommended elements per wavelength is 15 or greater.

```

symb wavevel = 1496.          /* Wave(long) velocity in medium (WATER at 25C)
symb freqint = 500.e3         /* Frequency of Interest (Hz)
symb wavelgth = $wavevel / $freqint /* Wavelength of Sound in Material (m)

symb numelem = 30             /* Number of elements per wavelength (>=15)
symb box = $wavelgth / $numelem /* The more elements, the finer the model

```

Code 3: Calculation of element size

### 2.3.4 I-J-K Co-ordinate system

Now that we have the element size and x and y key points, a grid of nodes can be created to represent the physical distances.

```

symb i1 = 1
symb i2 = $i1 + nint ( ( $x2 - $x1 ) / $box )
symb i3 = $i2 + nint ( ( $x3 - $x2 ) / $box )
symb indgrd = $i3

symb j1 = 1
symb j2 = $j1 + nint ( ( $y2 - $y1 ) / $box )
symb jndgrd = $j2

```

Code 4: Creating nodes to represent the physical dimensions

The process to create the number of nodes is straightforward: start with the very first node, the distance between consecutive key points is divided by the size of an element to find out how many nodes are required to represent that distance. This is repeated for the remaining key points and for each axis.

```

grid $indgrd $jndgrd /* maps out a grid previously declared number of ijk nodes

```

Using the grid command, the number of nodes determined for the I and J directions are used to create the respective grid of nodes.

### 2.3.5 GEOM Definition

With the grid of nodes created, the key points are mapped exactly to specific node points using the GEOM command.

```

geom
  xcrd $x1 $x2 $i1 $i2
  xcrd $x2 $x3 $i2 $i3
  ycrd $y1 $y2 $j1 $j2
end

```

Code 5: Associating X and Y points to the I and J nodes

### 2.3.6 Material Definition

PZFlex provides a generic materials file with the most common materials used in models. The data is simply read in to allow access to the file.

```

symb #read general.matr /* Allows access, from this file, to all typical materials

```

Users can also define their own materials – isotropic, anisotropic etc.

### 2.3.7 Material Site Definition

With all the key points declared previously, the site definition becomes effortless. All that is required is to define an area between key points as a specific material. In Figure 4, the area from x2 (i2) onwards is region of no material and extends for the whole of the y-axis.

```
site
  regn watr
  regn void $i2 $i3 $j1 $j2
end
```

Code 6: Site definition code

If no I-J-K coordinates are entered for the REGN subcommand, the whole grid is defined as that particular material. The REGN command takes in the node points and not the X-Y-Z dimensions. It should be noted that each subsequent issue of the REGN or site defining subcommand will overwrite the previous material defined in that area i.e. 'void' overwrites the 'watr' in the region stated in Code 6.

### 2.3.8 Boundary Conditions

The boundary conditions tell the model how to react at the edges where the defined model ceases. The most commonly used settings are FREE, ABSR and SYMM. If the boundary is set to FREE, it acts as a perfect reflector. If the boundary is set to ABSR, the energy incident on this boundary is absorbed. If the model continues 'infinitely' after the boundary then SYMM should be used. SYMM is also useful when symmetry is used to create the full model which will be discussed later (see '2.3.1 Physical Dimensions').

```
boun
  side xmin free
  side xmax absr
  side ymin symm
  side ymax absr
end
```

Code 7: Each side of mode set to different conditions

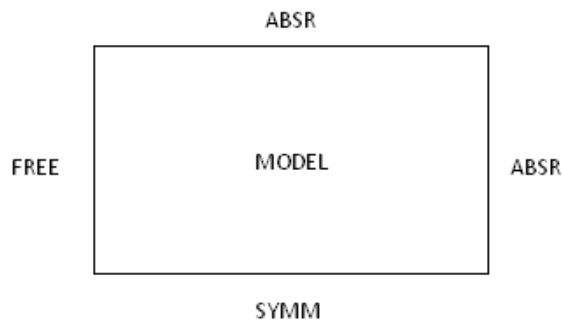


Figure 5: Sketch of boundary setup

Code 7 sets up the boundaries as shown in Figure 5.

### 2.3.9 Driving Function

```
c PARAMETERS: FUNC <type> <frequency> <amplitude> <phase> <periods/cycles>
func sine $freqint 1. 0. 1.      /* apply single cycle sinusoid
```

The driving function can be set to various waveforms available in the FUNC command (refer to section '5.4 Using the FUNC command'). For this example, a single sinusoid cycle at the previously declared frequency will be used as the propagating wave.

### 2.3.10 Property Calculation

The different properties of the model can be calculated during the execution of the model. The common calculated properties are pressure, displacement, stresses and strains. There are other secondary functions allowing a lot more to be calculated. Additional available functions are listed in the manual.

```
calc
  disp
  pres
  max pres pmin pmax|
end
```

Code 8: Calculates surface displacement, pressure minimum and maximum

### 2.3.11 Pressure Load Application

To apply a pressure wave, the PLOD function is used. For this example, a pressure will be loaded into the xmin boundary.

```
pload
  pdef pld1 func          /* use previously defined 'func' - sine
  vctr vct1 1. 0. 0.      /* Pressure wave in x-axis direction
  sdef pld1 vct1 $i1 $i1 $j1 $j3 /* Applying pressure wave to left hand nodes of material
end
```

Code 9: Generation of a pressure wave travelling in the positive x direction

PDEF requires a name for the pressure wave and the type of wave which was declared in the Driving Functions section. VCTR is then called to specify the direction the wave propagates in the model. Finally, SDEF takes the type of wave and direction and applies it to the set of nodes in the declared region. For a more in depth analysis of the PLOD function, refer to section '5.5 Using the PLOD command'.

### 2.3.12 Time Histories

The time histories are the data stored for analysis once the model has finished executing. Data such as pressure can be collected from a single or group of nodes in the model. In this model, the driving function is to be stored and can be viewed when the model has finished running.

```
pout
  hist func /*Stores the The input function - CAN be viewed by using INSIGHT in toolbar
end
```

### 2.3.13 Execution Command

```
symb #get { step } timestep
symb simtime = ( $areawidth / $wavevel ) * 1.5 /*Increase multiplier to increase simulation time
symb nexec = nint ( $simtime / $step )

symb nloops = 25
symb nexec2 = nint ( $nexec / $nloops )
```

Code 10: Calculates the required timestep for a specified simulation time and number of loops

To execute the code, time information is required which is obtained using Code 10. The two parameters that are required from the user are the simulation time and the number of loops.

```
c *****
c PLOTTING PROCEDURE
c *****
proc plot save
exec $nexec2

grph
  nview 2 2          /* NVIEW opens a window with a specific setup i.e split into 2 subwindows vertically
  color tabl data 6
  mirror y           /* MIRROR mirrors the current model in a specific axis, useful for symmetrical models
  ttl
Max Pressure Plot (LEFT) and Standard Pressure Plot (RIGHT)
  plot pmax
  plot pres          /* plots the pressure profile of the model
  imag              /* used for .avi generation
end
set pmax 0.0

end$ proc

c *****
c Calls procedure PLOT
c *****
proc plot $nloops /*executes 'plot' nloops times
```

Code 11: runs model and displays pressure plots called within a procedure

Once Code 10 has been set up, the model is run using the EXEC command. In Code 11, a procedure has been created to which, not only runs the code but plots the pressure fields too. A procedure is a segment of code that can be called on by its name: similar to functions in C programming. All the code in the procedure run every time 'plot' is called from the PROC command and can be looped a specific amount of times by entering an integer value/variable as following parameter as shown in Code 11.

### 2.3.14 Summary

Most of the basics to setting up a model have been visited and with a good understanding of these fundamentals, all that is required is to learn more functions to allow the generation of more complex models.

## 2.4 Changing the Target

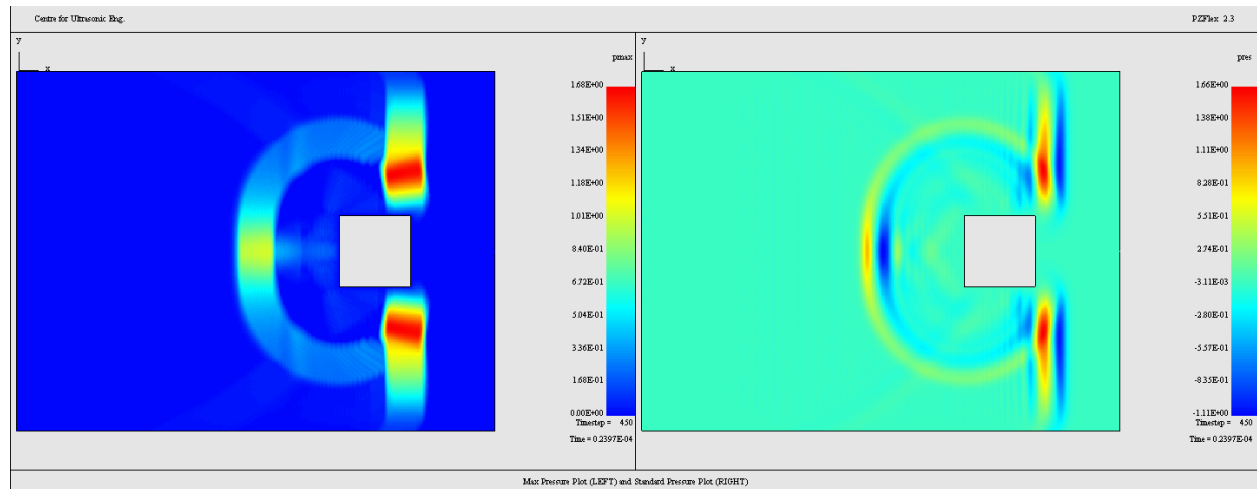


Figure 6: Planwave interacting with a square void in water

Using the “2.3 A Simple Model – Planewave Propagation” as the base model, it can easily be altered to the scenario shown in Figure 6. The changes to the base model will be highlighted in the following sections.

### 2.4.1 X-Y-Z Co-ordinate System

The defined XYZ co-ordinates will need to be amended as there will be extra key points along both the X and Y axes to reference the corners of the square void. A sketch of the model should look as follows:

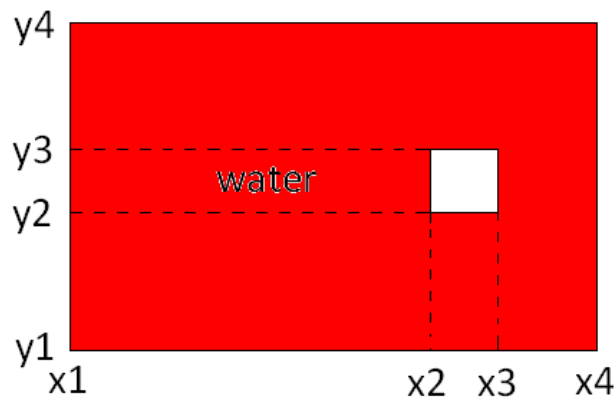


Figure 7: Key points marked on each axis of the model

This can be further simplified due to the ability to use symmetry in PZFlex: only half the model needs to be analysed and then symmetry can be applied to complete the model. Symmetry should always be considered in modelling as it can greatly reduce the burden on the processor and time taken for models to execute.

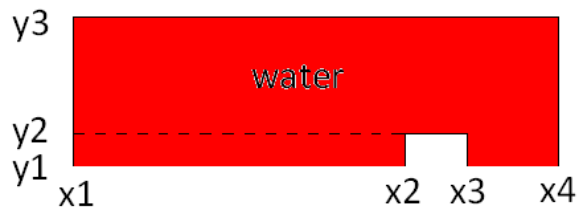


Figure 8: Simplified Model in Figure 7

Symmetry would be applied along the x-axis in Figure 8 to create the same model in Figure 7.

```
symb x1 = 0.0                                /* Mapping out key distances from the origin for each axis used
symb x2 = $x1 + ( $targetcentre - ( $targetwidth / 2 ) ) /*Remember spaces between each operation
symb x3 = $x2 + $targetwidth
symb x4 = $x1 + $areawidth

symb y1 = 0.0
symb y2 = $y1 + ( $targetheight / 2 )
symb y3 = $y1 + $areahght
```

Code 12: Extra points mapped out using previously defined physical dimensions

An extra symbol will need to be added along the x-axis to mark out where the square void begins and ends. Similarly along the y-axis, a symbol is added to show where the square void ends.

## 2.4.2 I-J-K Co-ordinate system

Changes made to the XYZ co-ordinates will also result in changes in the IJK system. As stated before, there should equal IJK points as there are XYZ points.

```
symb i1 = 1
symb i2 = $i1 + nint ( ( $x2 - $x1 ) / $box ) /* finding the number of nodes between the distance x2 and x1
symb i3 = $i2 + nint ( ( $x3 - $x2 ) / $box )
symb i4 = $i3 + nint ( ( $x4 - $x3 ) / $box )
symb indgrd = $i4 /*reference to end of grid in the i direction

symb j1 = 1
symb j2 = $j1 + nint ( ( $y2 - $y1 ) / $box )
symb j3 = $j2 + nint ( ( $y3 - $y2 ) / $box )
symb jndgrd = $j3
```

Code 13: Code follows the same pattern for the new points

No changes are required for the GRID command.

## 2.4.3 GEOM Definition

With the extra defined points, the XYZ points and IJK points will need to be linked by the GEOM command.

```
geom
xcrd $x1 $x2 $i1 $i2
xcrd $x2 $x3 $i2 $i3
xcrd $x3 $x4 $i3 $i4
ycrd $y1 $y2 $j1 $j2
ycrd $y2 $y3 $j2 $j3
end
```

Code 14: Associating all XYZ and IJK points



#### 2.4.4 Material Site Definition

To define square void in the model, the REGN command is used in exactly same way as for the boundary example before: the area that is now required to be void is between x2, x3, y1 and y2.

```
site
  regn watr
  regn void $i2 $i3 $j1 $j2
end
```

Code 15: Declaring new void region

#### 2.4.5 Boundary Conditions

When symmetry is applied to models, the axis to which it is applied will need to be set SYMM in the boundary conditions. No changes need to be made.

#### 2.4.6 Pressure Load Application

The pressure load will need to be altered slightly due to the newly defined point on the y-axis. The load should be applied to all the J nodes and so will require a simple change to cover all the appropriate nodes.

```
pload
  pdef pld1 func
  vctr vct1 1. 0. 0.
  sdef pld1 vct1 $i1 $i1 $j1 $j3
end
```

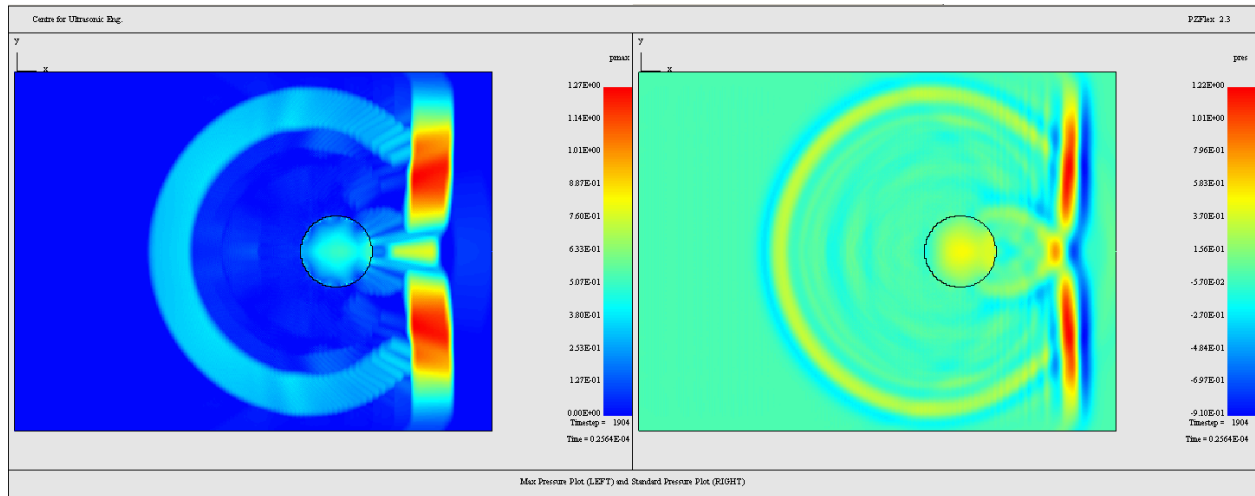
Code 16: Small change made to the SDEF command to apply pressure from nodes j1 to j3

The remaining code can be left unchanged.

#### 2.4.7 Summary

A simple change in the model can lead to changes in various sections of the code as demonstrated in this example. With a basic grasp of where and what effect each section of code has on the model, more complex examples can be explored. To develop and test your knowledge of PZFlex, the guide will no longer highlight specific sections where changes have been made to the code.

## 2.5 A Circular Object



The square void in the previous example can be changed into a circular target very easily. There are two methods of creating a circular target either using the SPHR or CYLN subcommand when declaring the regions of materials (SITE). Both can be set up using one key point: the centre point of the circle. Both subcommands will take in a radius constraint, allowing the size of the circle to be controlled. The material of the circular target can also be modified to any of the materials present in the 'general.matr' file.

```
site
  regn watr
  sphr steel $x2 $y1 0.0 3.e-3 0.0
end
```

Code 17: Use of SPHR subcommand including centre position and radius declaration

## 2.6 Spherical Waves

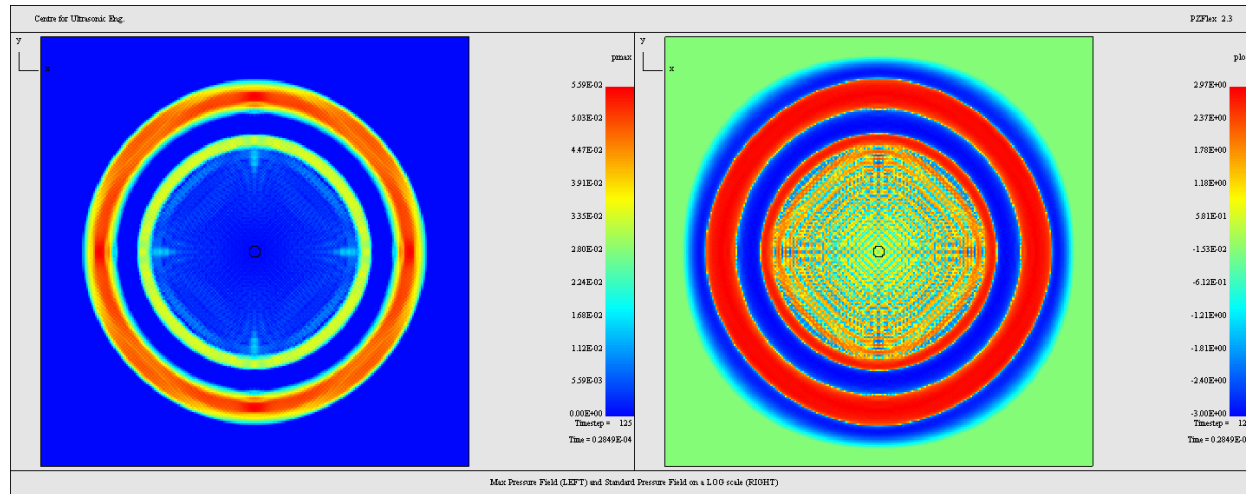


Figure 9: Spherical wave model

A spherical wave propagates through space like the outer surface of a continually growing sphere. This example shows how spherical wave can be set up. Set up a model of a size suitable for the wavelength under analysis. During the process of declaring materials, create two materials with the same properties but with different names. The reason for this is to allow a pressure load to be applied to a small region of nodes or small circular boundary.

```
pload
pdef pld1 func
spot spot1 $x2 $y2
sdf2 pld1 spot1 watr watr2
end
```

Code 18: Generation of a spherical wave

Like in previous examples, define the function that will be applied as the pressure load. The subcommand SPOT locates the side of a boundary to apply pressure to: in this case, the small circular boundary around 'spot1' that separates 'watr' and 'watr2'. With the defined pressure load and spot location, SDF2 can be used to automatically apply the pressure wave at the boundary between the two materials without specifying the exact nodes.

NOTE: Symmetry can be used to generate this model

## 2.7 Angled Boundaries

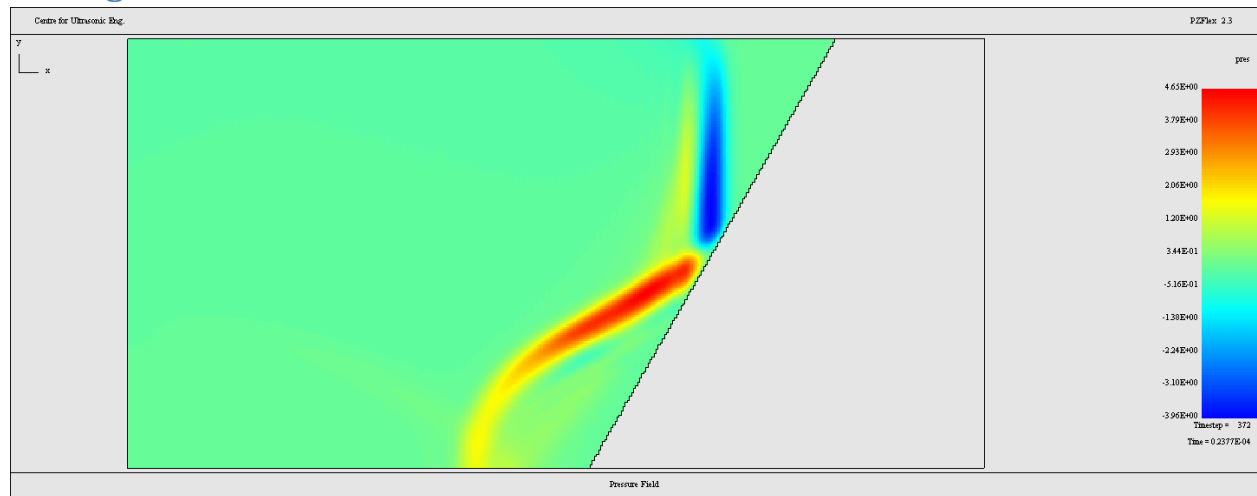


Figure 10: A wave reflecting off an angled boundary

Angled boundaries can be produced by using the BLOK or CYLN subcommand available in the SITE primary command. For the simple model shown in Figure 10, the CLYN command would be the easier method to use as it only requires one line of code.

```
site
  regn watr
  cyln void stnd z 0 1 $x2 $y1 50.e-3 0 0 0 60
end
```

Code 19: Creating an angled wedge using CYLN subcommand

The CYLN command looks to have a lot of parameters but most of them are either default values or do not require a value other than zero. The first parameter is the type of material; STND aligns the cylinder to the global axis; Z declares the axis the length of the cylinder lies on; the next two parameters declares where the cylinder starts and ends on the Z axis (2D model not important); the following two parameters defines the centre point of the circular face of the cylinder; the radius of the circular face can then be set to an appropriate size to cover the desired area and the last two parameters controls the starting and finishing angle of the circular face.

```
symb angle = 60/* Angle at which BLOK will be angled at

axis
  form angl
  defn src1 cart $x4 $y1 0 0. 0. $angle
  end

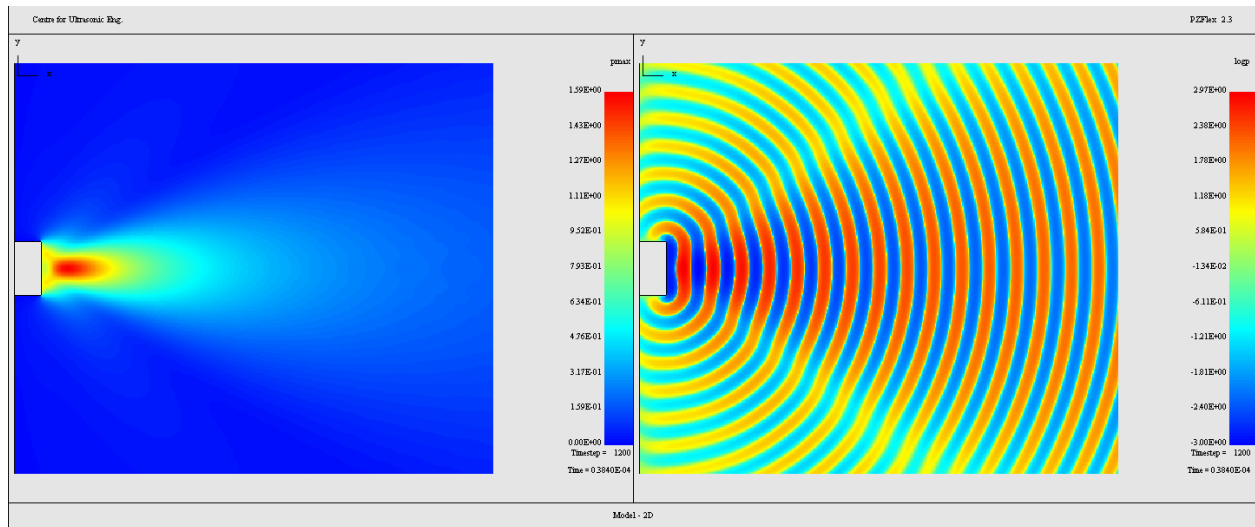
site
  regn watr
  blok void cent src1 -25e-3 25e-3 -35e-3 20e-3
  end
```

**Code 20: Creating angled wedge using the BLOK subcommand**

The BLOK command requires a new axis to be formed to allow specific orientation of the block. The AXIS command requires you to name, define the origin location and the angle which it will be rotated. When defining the material, the BLOK subcommand will use the previously declared axis and require the dimensions of the block to be entered as well.

NOTE: by combining the angled surface with the SDF2 subcommand for pressure loading, an angled wave can also be created. For more information on using SDF2 refer to section '5.5.4 Generating a Spherical Wave'.

## 2.8 Piston Source and Beam Profiles



**Figure 11: Model of a piston generating constant pressure wave**

In this model, a constant pressure load will be applied to the front face of a piston. Ideally, ultrasonic transducers would behave like pistons but this is not the case due various modes interfering with the main resonant mode (thickness). A useful measure of ultrasonic transducers is a beam profile which shows how the pressure wave constructively interferes to create a concentrated beam of ultrasound.

To obtain a beam profile, in the CALC section, the maximum and minimum pressure values should be calculated.

```
calc
  disp
  pres
  max pres pmin pmax
end
```

**Code 21: Calculating the relevant pressure values**

In the plotting procedure, 'pmax' can be plotted to obtain the beam profile.

```
proc plot save
exec $nexec2

data log pres logp

grph
  nview 2 2
  colr tabl data 6
  mirr y
  plot pmax
  plot logp
  imag
  end

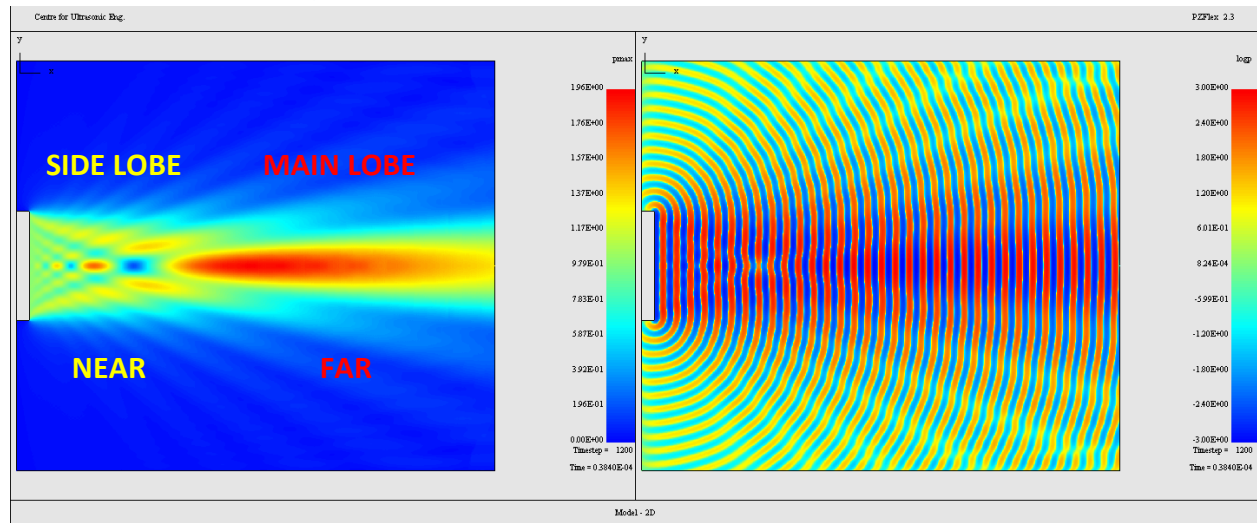
set pmax 0.0

end$ proc
```

**Code 22: Changes made to the plot procedure to obtain beam profile**

After plotting 'pmax', the next timestep there may be a different max pressure value so normalising to the maximum value will give a better contrast between areas of high and low pressure.

## 2.9 Effect of Varying Piston Size



The piston size or, relating back to the ultrasonic transducers, the diameter/radius of a circular piece of piezo material has a large effect on its performance. Theory states that when the radius of the piston is greater than the wavelength of the propagating wave, there will be a distinct near-field and far-field region in the beam profile. In the near field region, a complex interference pattern exists creating undesired side lobes. Side lobes are areas of high acoustic pressure at different and defined angles from the main lobe [5]. The far-field region is where main lobe begins and steadily attenuates with increasing distance. For piston radii smaller than the wavelength in consideration, the beam profile will resemble that of a simple point source. This is an important consideration for applications that require a distinct focal point (imaging).

To establish that Finite Element Analysis is consistent with theory, a piston can be modelled in water to demonstrate how the beam profile changes.

```
symb wavevel = 1500.          /* Wave(long) velocity in medium (WATER at 25C)
symb freqint = 1.e6           /* Frequency of Interest (Hz)
symb wavelgth = $wavevel / $freqint /* Wavelength of Sound in Material (m)

symb sourcerad = $wavelgth * 4/* object dimensions within medium
symb sourcethck = $wavelgth * 1

symb areawdth = $wavelgth * 35 /* All dimensions are CONSISTENTLY in Metres (m)
symb areahght = $wavelgth * 15
```

All physical dimensions should be in terms of the wavelength so that a multiplier can simply be changed to alter the radius of the piston. A model of a piston will be executed at radius of  $\frac{1}{4}$  wavelengths and then a multiple of two for further simulations to observe the effects of different piston sizes.



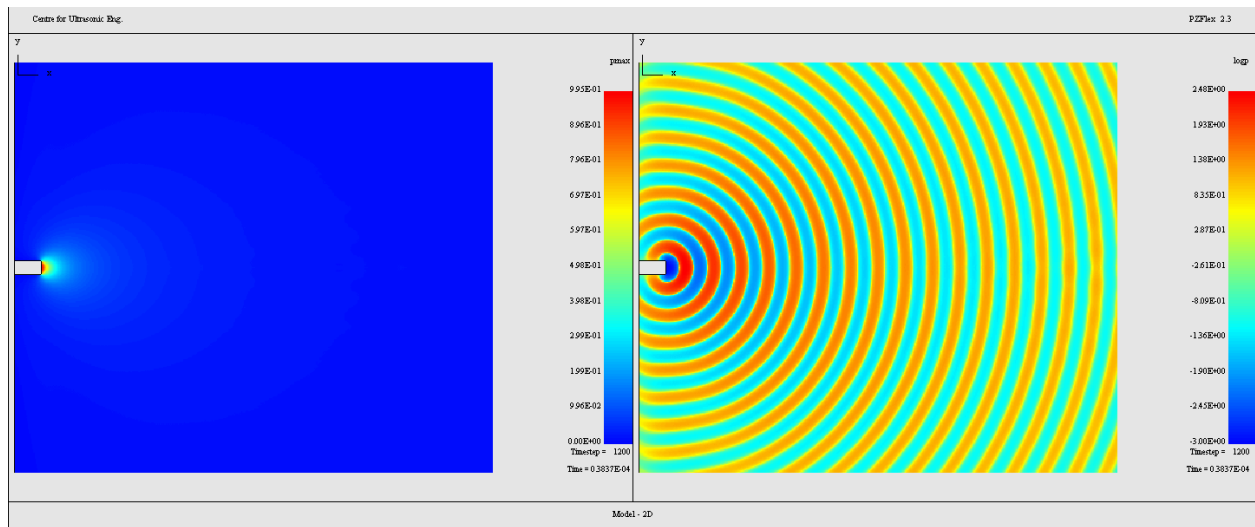


Figure 12: A quarter wavelength radius

Figure 12 shows that the piston is essentially a point source as circular waves propagate from the piston and the only high pressure region is at the front face of the source.

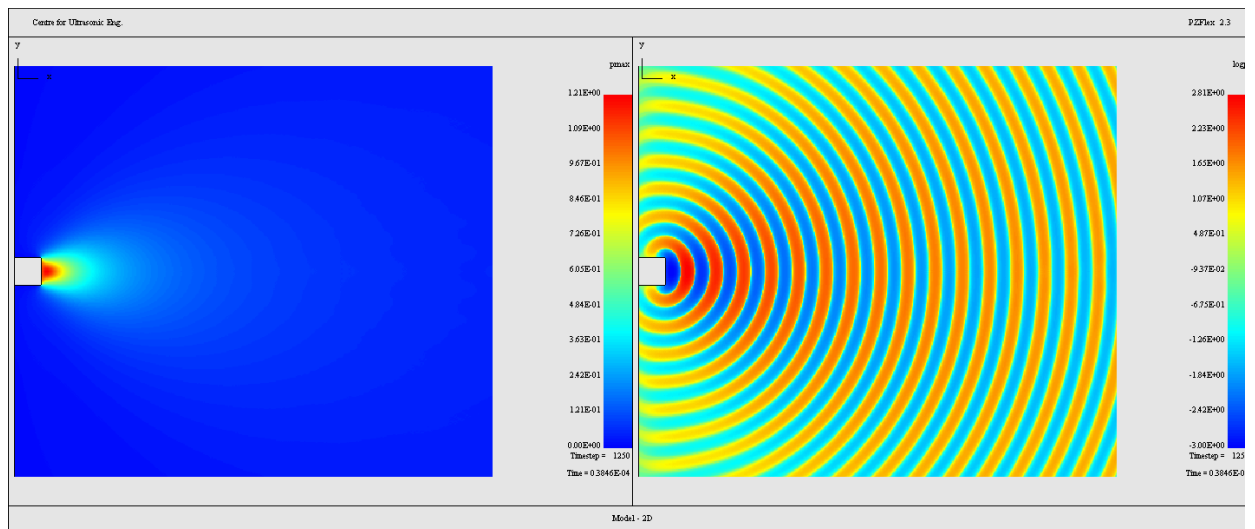


Figure 13: A half wavelength radius

At a half wavelength, the piston is still behaving as a point source. However, there is a slight change to the beam profile as the high pressure region has extended slightly beyond the front face of the piston.

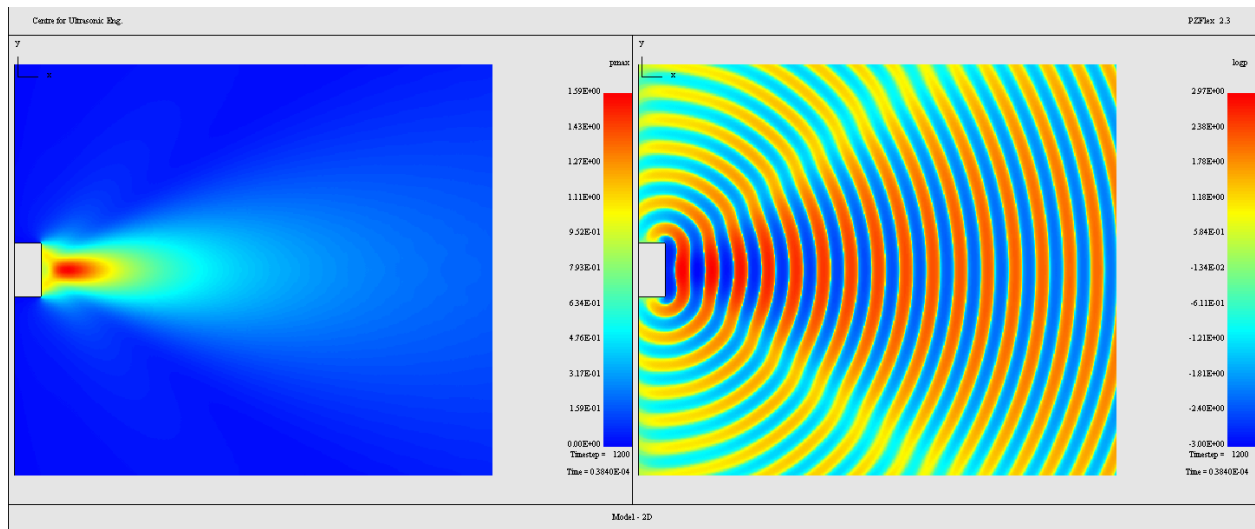


Figure 14: A wavelength radius

The beam profile shown in Figure 14 has now a much more noticeable main lobe. The effects of side lobes are still not visible with a radius of one wavelength.

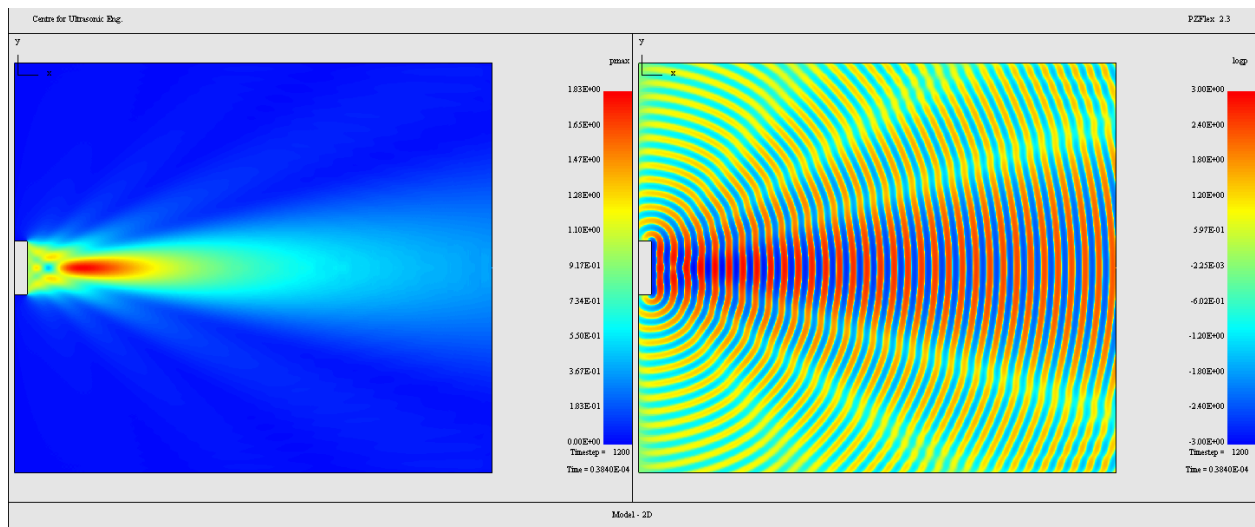
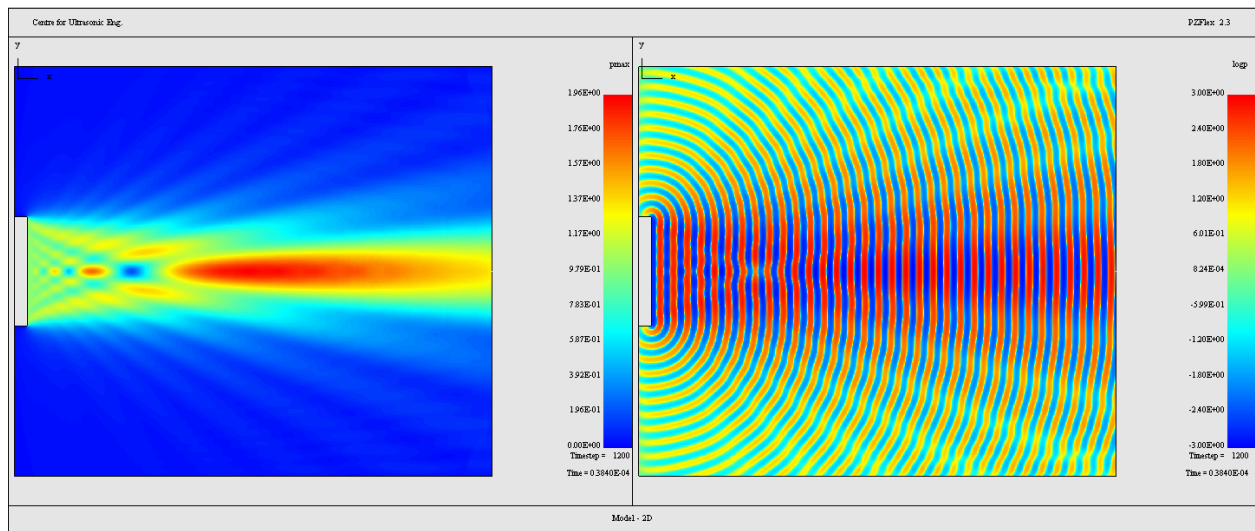


Figure 15: Double wavelength radius

Again the ultrasonic beam has become more concentrated and it can be seen in Figure 15 that the near-field and far-field regions are beginning to develop.

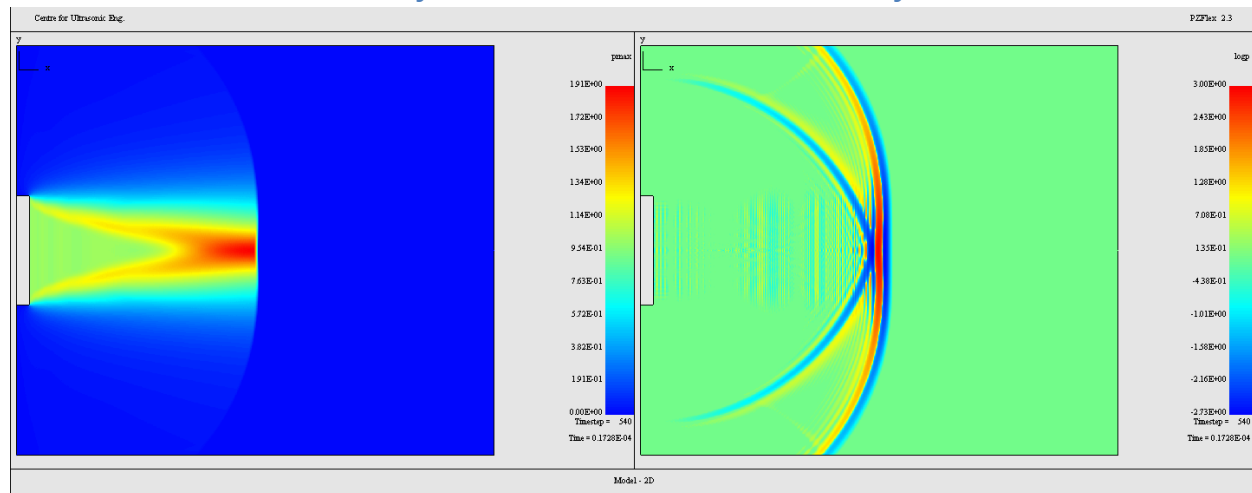


**Figure 16: Quadruple wavelength radius**

At a radius of four times the wavelength, the near-field and far-field regions have become unmistakable. Side lobes can be identified at the edges of the beam profile at the near field region as well as the complex interference pattern closer in the midst. The main lobe has again extended further which would allow focal points further away from the source, however, the trade-off is greater side lobes.

The models have confirmed the theory: for radii less than a wavelength, the piston behaves as a simple point source and for radii greater than a wavelength, the beam profile is separated into a near-field region where the complex interference pattern results in side lobes and a far-field region where the main lobe lies.

## 2.10 Transient Wave Analysis vs. Continuous Wave Analysis



Instead of using continuous wave stimulation to find the different field profiles, a transient wave could have been used i.e. using an impulse as the driving function. An impulse function is generally used to find the response of a system. In theory, an impulse contains all frequency components and so it is able to determine how the system responds to any particular frequency. For the beam profiles, a similar method can be adopted to obtain the pressure fields as if the piston size was changing.

All that is required is to keep the piston size and frequency of interest the same and apply an impulse as the pressure load. In the example, the frequency of interest is set to 1 MHz and so, essentially, the source is being driven by all frequencies up to 1 MHz. The same beam profiles from the previous example, '2.9 Effect of Varying Piston Size', can all be obtained from this one model by the extracting the pressure fields at the different frequencies. Note the beam pattern with a transient wave has no interference pattern due to the lack of iteration of edge waves.

The key command to use in the code is the SHAP command followed by the FREQ subcommand to define the frequency that the pressure field will be extracted at.

```
shap
  data pres
  freq 62.5e3
  freq 125.e3
  freq 250.e3
  freq 500.e3
  freq 1.e6
end
```

Code 23: Using the SHAP command

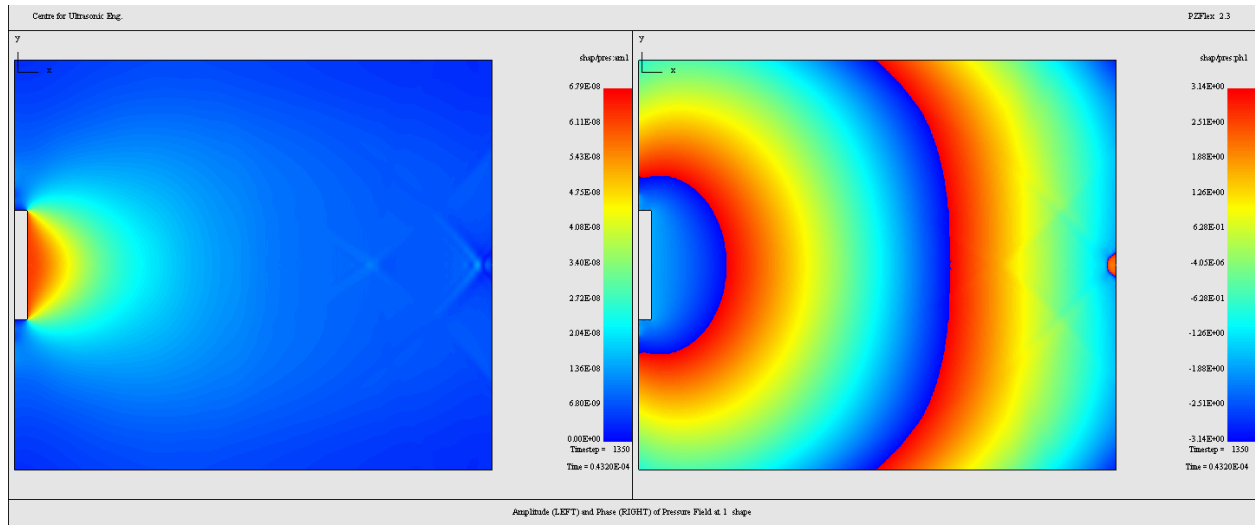


Figure 17: Pressure field at 62.5 kHz

Figure 17 is the same pressure field as Figure 12.

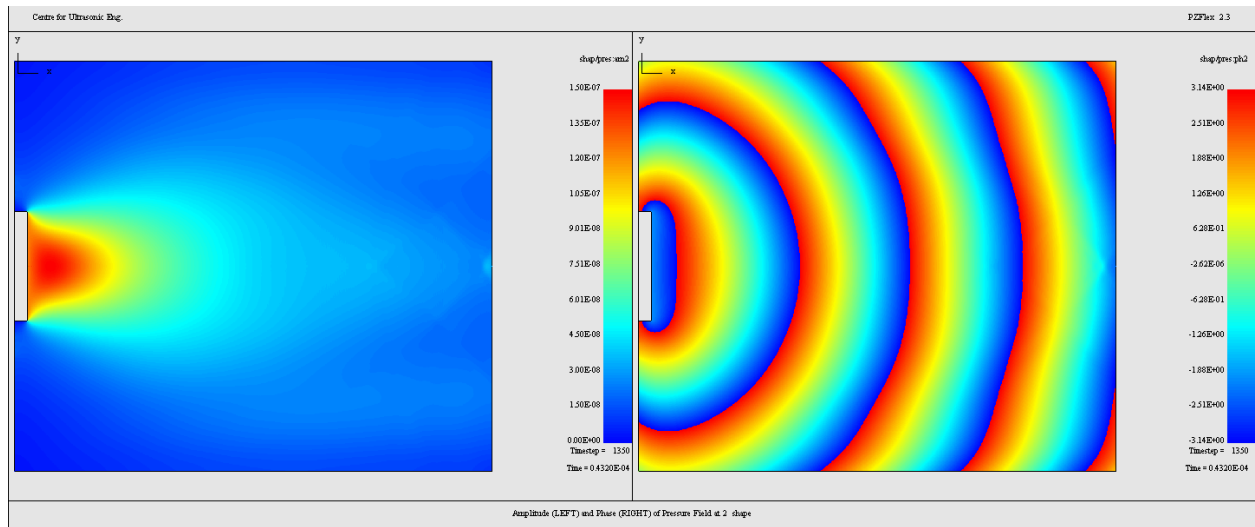


Figure 18: Pressure field at 125 kHz

Figure 18 is the same pressure field as Figure 13.

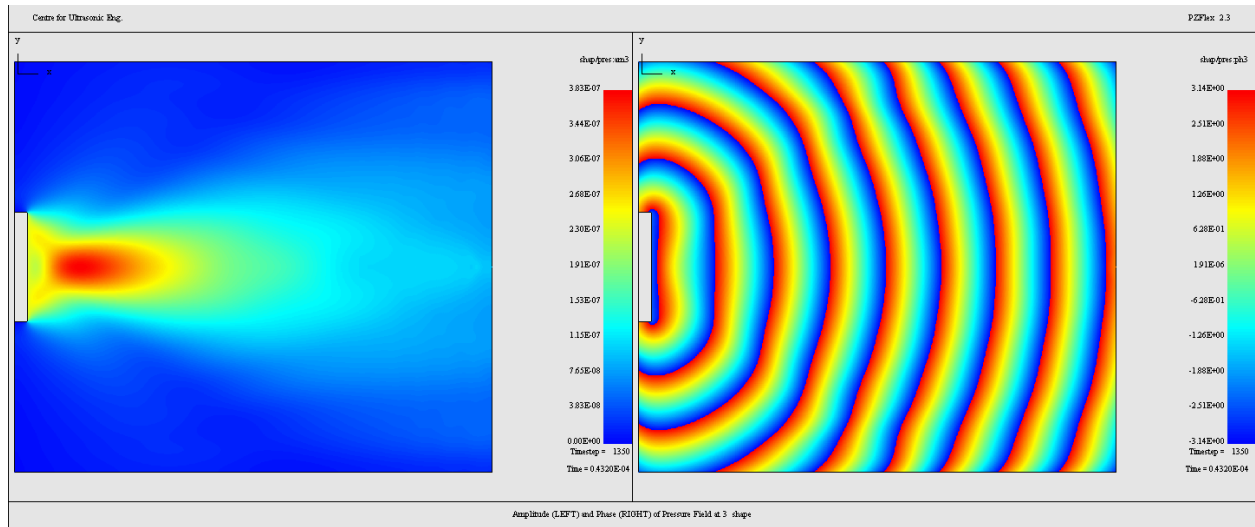


Figure 19: Pressure field at 250 kHz

Figure 19 is the same pressure field as Figure 14.

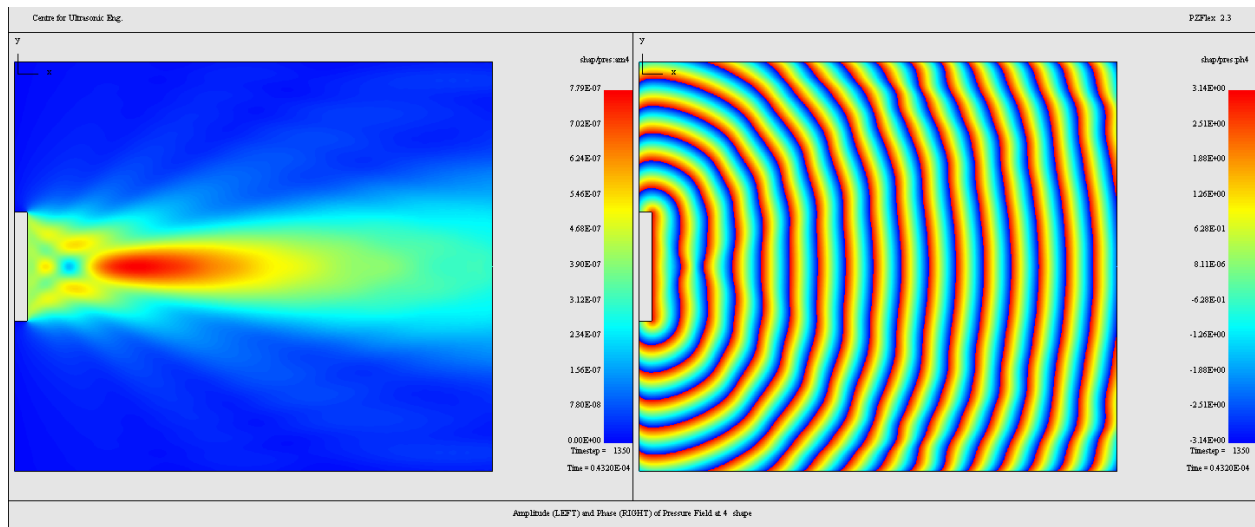
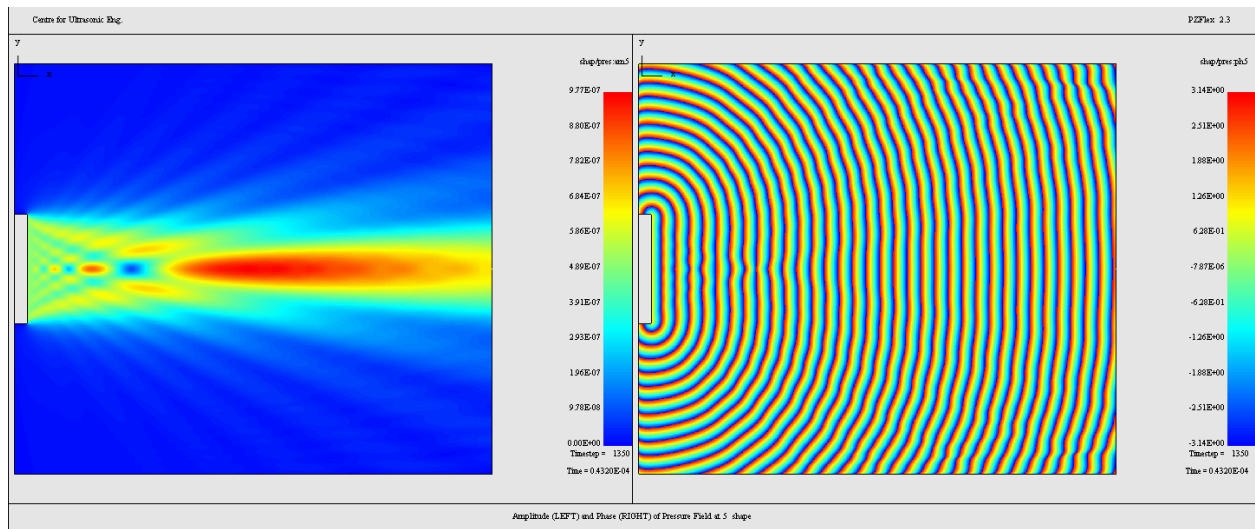


Figure 20: Pressure field at 500 kHz

Figure 20 is the same pressure field as Figure 15.





**Figure 21: Pressure field at 1 MHz**

Figure 21 is the same pressure field as Figure 16.

By changing the frequency at which the pressure field is obtained, the model produces the same plots as changing the radius of the piston. Decreasing the frequency implies increasing wavelength: the piston size is kept at a constant 4 times the wavelength of a 1 MHz signal and so at lower frequencies (or larger wavelengths), the device is effectively becoming smaller.

Transient wave is definitely the better method of analysis as it can save time and still yield the appropriate information from the model.

## 2.11 Piezoceramic Disk (Hard-code)

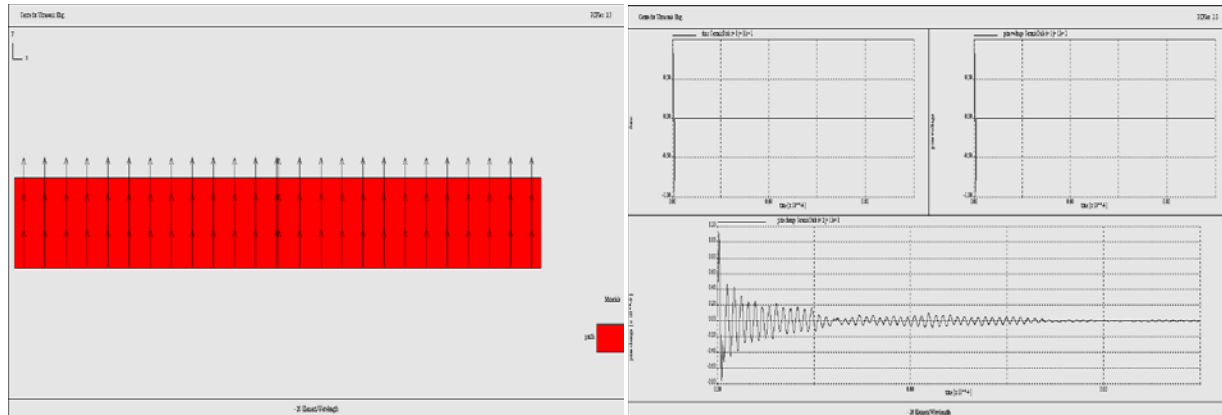


Figure 22: Ceramic disk analysis

Modelling the ceramic disk is very simple due to the axisymmetric that can be applied about the y-axis to create the full 3D disk. There is however, a new section of code that needs to be applied to model the ceramic disk and that is the electric field application.

### 2.11.1 Electric Field Application

As it is a piezo device, a voltage needs to be provided across the device for it to undergo mechanical deformation. This is achieved by attaching electrodes to the top and bottom surface of the device. In this section, the basic code to connect a positive and ground electrode to the device will be reviewed.

```

symb ascale = 2.          /* ratio of model electrode size to
                             /* actual electrode size

piez
  wndo $i1 $i2 $j1 $j2

  defn uppos $ascale
  node $i1 $i2 $j2 $j2 /*top surface of ceramic

  defn lowneg $ascale
  node $i1 $i2 $j1 $j1 /*bottom surface of ceramic

  bc uppos volt func
  bc lowneg grnd

end

```

Code 24: Setting up electrodes and connecting them to the piezo device

The 'ascale' symbol is used whenever symmetry is applied or in 2D models to account for the dimension 'into the page'. The command to use for applying an electric field to a piezoelectric device is PIEZ. The WND0 subcommand is used to define the nodes that construct the piezo device which in this case is the entire model. Using DEFN, a positive and negative electrode is assigned a name and scaled. Immediately after each electrode definition, they are applied to a specific set of nodes (NODE subcommand) with the usual I and J coordinates. The electrode boundary conditions (BC subcommand) are then applied to the model using the electrode name and the boundary option. Depending on the boundary option, a further



parameter will need to be entered i.e. VOLT option selected, FUNC should be declared to define the time history.

```
pout
hist func                /* The input function TIME HIST - 1
histname electrode vq uppos /* Saves voltage and charge (vq) for uppos electrod
                             /* in TIME HIST - 2 & 3
end

shap
freq 60.0e3
freq 160.0e3
freq 690.0e3
end
```

**Code 25: Recording different data in time histories and calculating device shapes**

For piezoelectric models, it is crucial to store voltage and charge data for the electrodes on device as the combination will allow an impedance profile to be retrieved. It is also good measure to also keep track of the input function. The SHAP command used previously to plot the beam profiles can be used here for calculating the mode shapes of the piezoelectric device at the specified frequencies. The SHAP commands are usually added after the first model run as they should reflect the electrical resonant frequencies obtained from the impedance profile of the device.

```
grph
  nview 2
  mirr x
  ttl 1
Displacement Shapes at 60kHz
  plot shap 1 0          /* plot requested shape at 0 degrees
  plot shap 1 180        /* plot requested shape at 180 degrees
end
t
```

**Code 26: Plotting the mode shapes at 0° and 180° phase**

The code used to plot the mode shapes at different phases is shown in Code 26.

Note: Impedance profiles can be easily calculated the 'Insight' tool in the task bar. It allows the user to select the Flex History file to open which is where the specified data is stored.

## Section 3 – Wizards

### 3.1 Introduction

PZFlex offers wizards to help set up the most common models so that the user does not need to hard-code everything. After execution of the model, results can be viewed by the provided interface allowing impedance profiles, movie generation and beam profiles to be obtained at a click of a button. All that is required from the user are the general parameters regarding the model i.e. materials used and physical dimensions. There are several different wizards that will be examined to investigate the types of wizards and their basic operation.

All of the available wizards can be selected from the 'wizards' pull down menu.

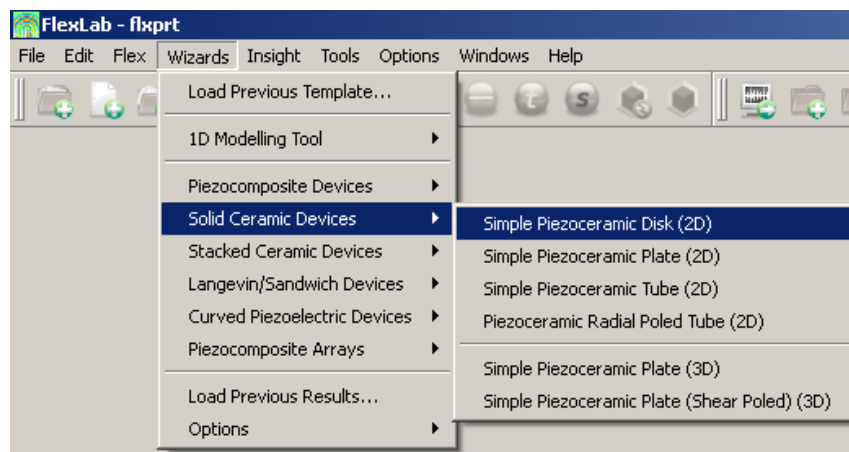


Figure 23: Wizards pull down menu

Most of the wizards are self explanatory: the simple examples to follow will give an insight into what can be achieved by the wizards.

## 3.2 Piezoceramic Disk

**Piezoceramic Disk Wizard Tool**

**Define your transducer**

1 Piezoceramic: PZT5H (Generic)

2 Thickness (mm): 3

Device Diameter (mm): 30

Lower Freq. (kHz): 0

4 High Freq. (kHz): 1581.9

5

Layer	Thickness (mm)	Material
Matching 2		None
Matching 1		None
Backing 1		None
Backing 2		None

6 ☐ Device in fluid? ☐ Add circuit 7

☐ Mesh Quality

Figure 24: Piezoceramic Disk Wizard 1st Interface

Figure 24 is the screen that is displayed when the 2D piezoceramic disk is selected. This offers the user to input the necessary parameters to generate the model. Each of the parameters are clearly labelled, making it straightforward to use:

1. Piezoceramic: Lists and allows user to select the appropriate material for the disk
2. Thickness: The thickness of the ceramic disk – alters frequency response of device
3. Device Diameter: Sets the diameter of disk
4. Lower/High Freq: The frequency range which the device is stimulated at
5. Backing/Matching Layers: Allows the addition of backing/matching layers to the device with constraints to control the thickness and material of the layers
6. Device in Fluid: Modelled under a specific fluid type – quicker ringdown
7. Add circuit: Specific circuitry can be designed and will be automatically connect to the device

**Piezoceramic Disk Wizard Tool**

**Define your transducer**

Piezoceramic: PZT5H (Generic)

Thickness (mm): 3

Device Diameter (mm): 30

Lower Freq. (kHz): 0

High Freq. (kHz): 1581.9

Layer Thickness (mm) Material

Layer	Thickness (mm)	Material
Matching 2		None
Matching 1		None
Backing 1		None
Backing 2		None

**Piezo Device**

☒ Device in fluid? Water at 25C ☐ Add circuit

☐ Mesh Quality

**Specify Outputs**

**Default Outputs**

By default the wizard will calculate Impedance, Admittance and GB-Plots

☒ Average Pressure/Displacement

☒ Transmitting Voltage Response

☒ Virtual Hydrophone

☒ **Mode Shapes**

Mode 1	Mode 2	Mode 3
60	160	690

Frequency (kHz)

☒ **Virtual Vibrometer**

The Virtual Vibrometer allows you to examine the front face displacement and pressure distribution for 3D models.

For 2D models, a slice through the model allows you to see the displacement and pressure distribution.

☒ **Beam Profiles**

Can calculate Radial or Field Plots

Frequency (kHz)

Field 1	Field 2	Field 3
60	160	690

**Figure 25: Post-processing options**

With these main settings covered, clicking on the 'Next' button will attach a further set of parameters to the interface as shown in Figure 25. The new options are all associated with post-processing: by default, impedance, admittance and GB plots are always calculated. Further options permit mode shapes and beam profiles to be calculated provided that the device has been modelled in fluid.

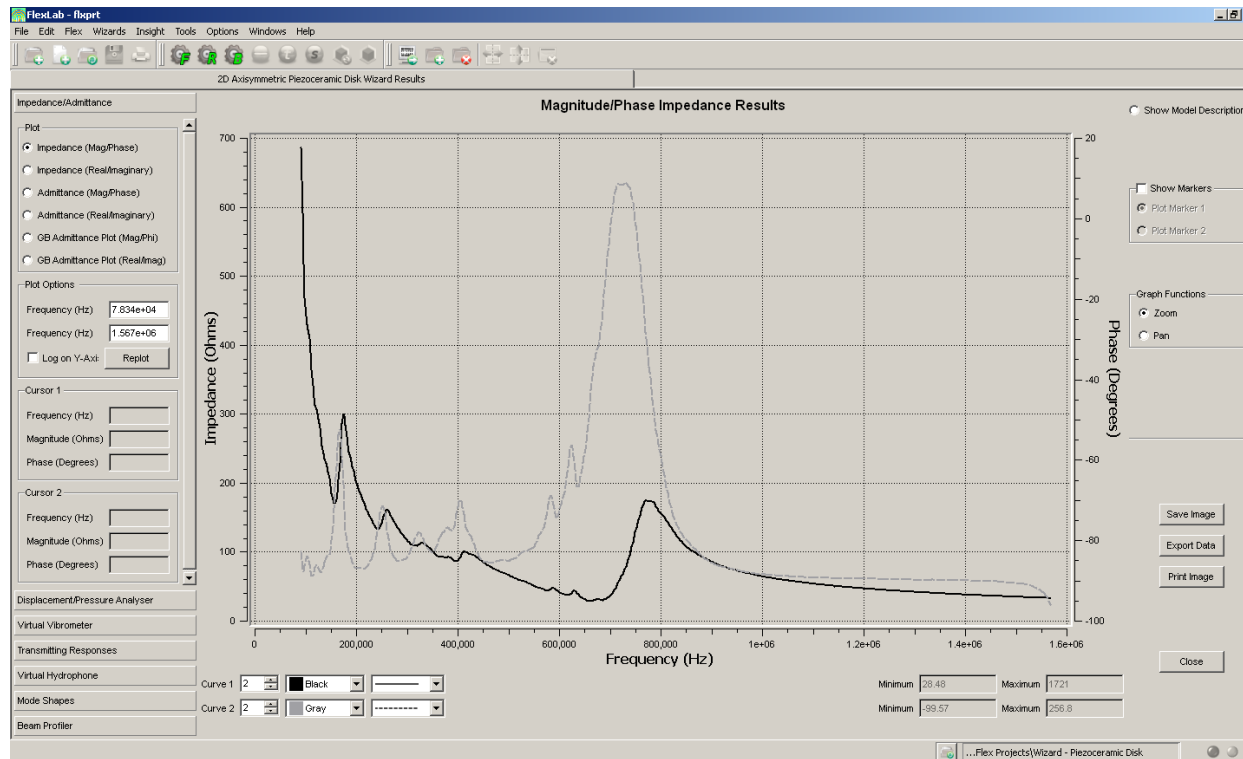


Figure 26: Post-processing screen

As mentioned before the wizards are simple to use and understand. Along the left hand side, there are various tabs to select the results that are of interest. Each tab will reveal more options related to the calculated property and sometimes will require parameters to be entered before generating the graphical results. The most common properties to analyse are the impedance profiles, mode shapes and beam profiles. The impedance will be the first graph generated on screen as shown in Figure 26.

To obtain modes shapes, click the 'Mode Shapes' tab and select the frequency which the mode shape will be analysed at. Either a movie of the device can be generated or the mode shapes at specific phases can be shown.

The beam profiler requires the frequency of analysis to be selected and dimensions of the field to generate the profile in. There are resolution settings available for the user to decide if a quick or a highly accurate profile is needed.

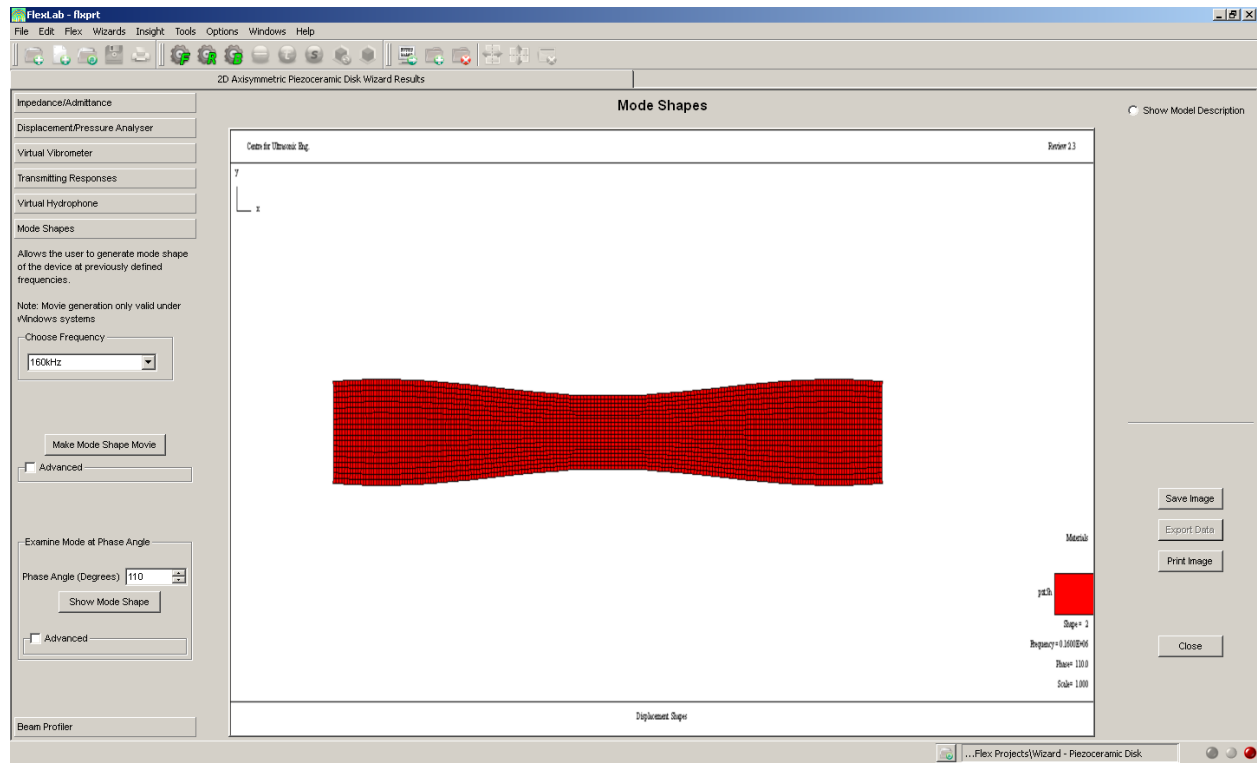


Figure 27: Generating mode shape at phase of 110°

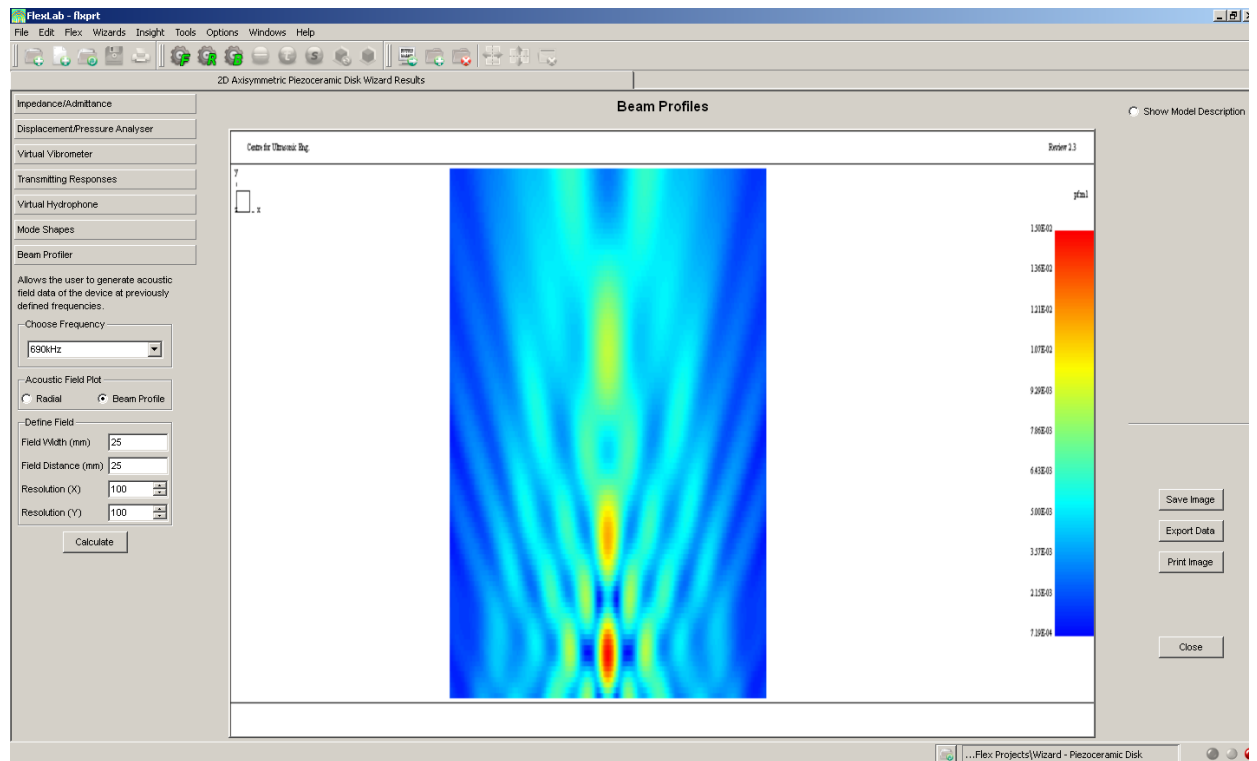


Figure 28: Generating beam profile at 690 kHz

### 3.3 2D and 3D Piezocomposite Structure

**2D Composite Wizard Tool**

**Define your transducer**

Piezoceramic: 3203HD

Epoxy Filler: Hard Set - Vantico HY1300/CY1301

Thickness (mm): 5

Kerf (mm): 1.5

Pillar Width (mm): 1.5

2D model approximation for 1-3 & 2-2 Piezocomposite

Device Depth (mm): 15

Device Width (mm): 15

No Backing or Matching

Lower Freq. (kHz): 0

High Freq. (kHz): 972.96

Layer	Thickness (mm)	Material
Matching 2		None
Matching 1		None
Backing 1		None
Backing 2		None

**Piezo Device**

☒ Device in fluid? Water at 25C ☐ Add circuit

☐ Mesh Quality

Figure 29: 2D Composite Device interface

The wizard offers the users the standard parameters associated with the composite structure: kerf, pillar width, epoxy filler material, ceramic material and width/depth of device. The interface will be familiar as it offers same standard features such as adding backing/matching layers, modelling in fluids and adding circuitry. Also, the post-processing options are identical to most if not all wizards.

This remaining part of this section will explore some of the effects of changing the pillar size whilst the kerf remains constant.

NOTE: Displayed results will be in the order of 0.75 mm, 1.5 mm and 3.0 mm pillar width

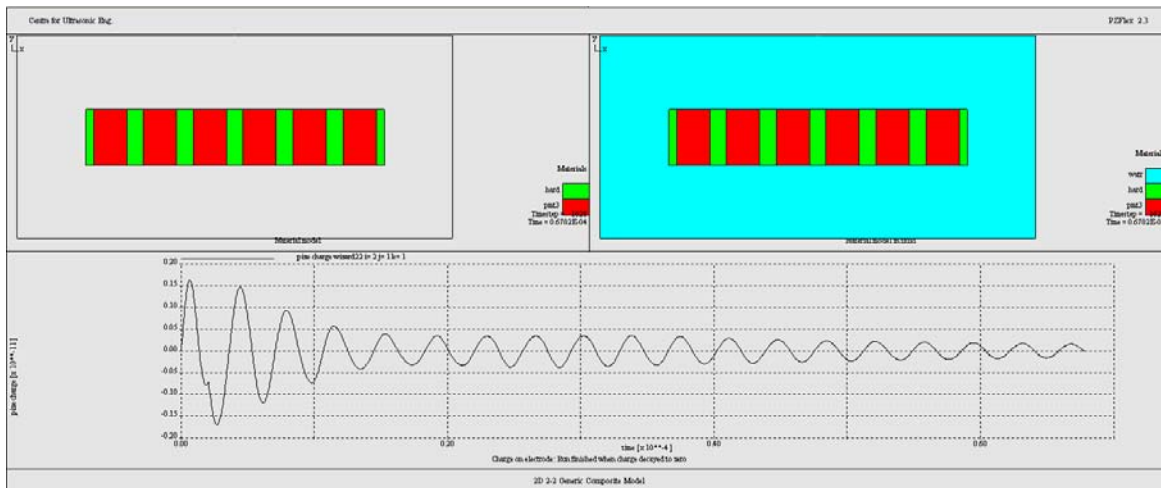
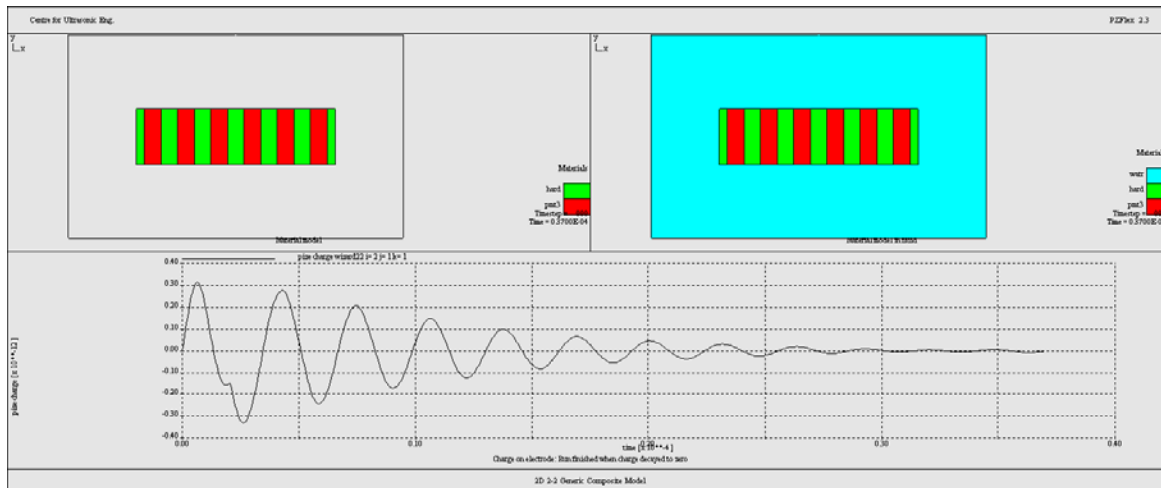
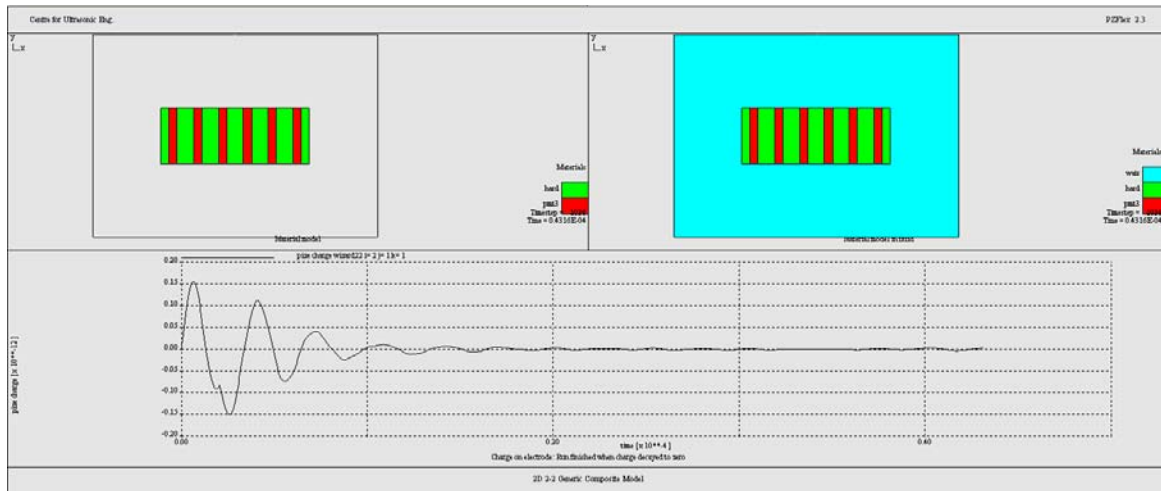


Figure 30: 0.75, 1.5 and 3 mm pillar width (top to bottom)



Figure 30 shows each device modelled in water. The difference in the ring down time is most noticeable: as the pillar width increased, the device resonated for a longer period of time. This is due to there being more piezo material, resulting in a stronger signal before reaching a steady state.

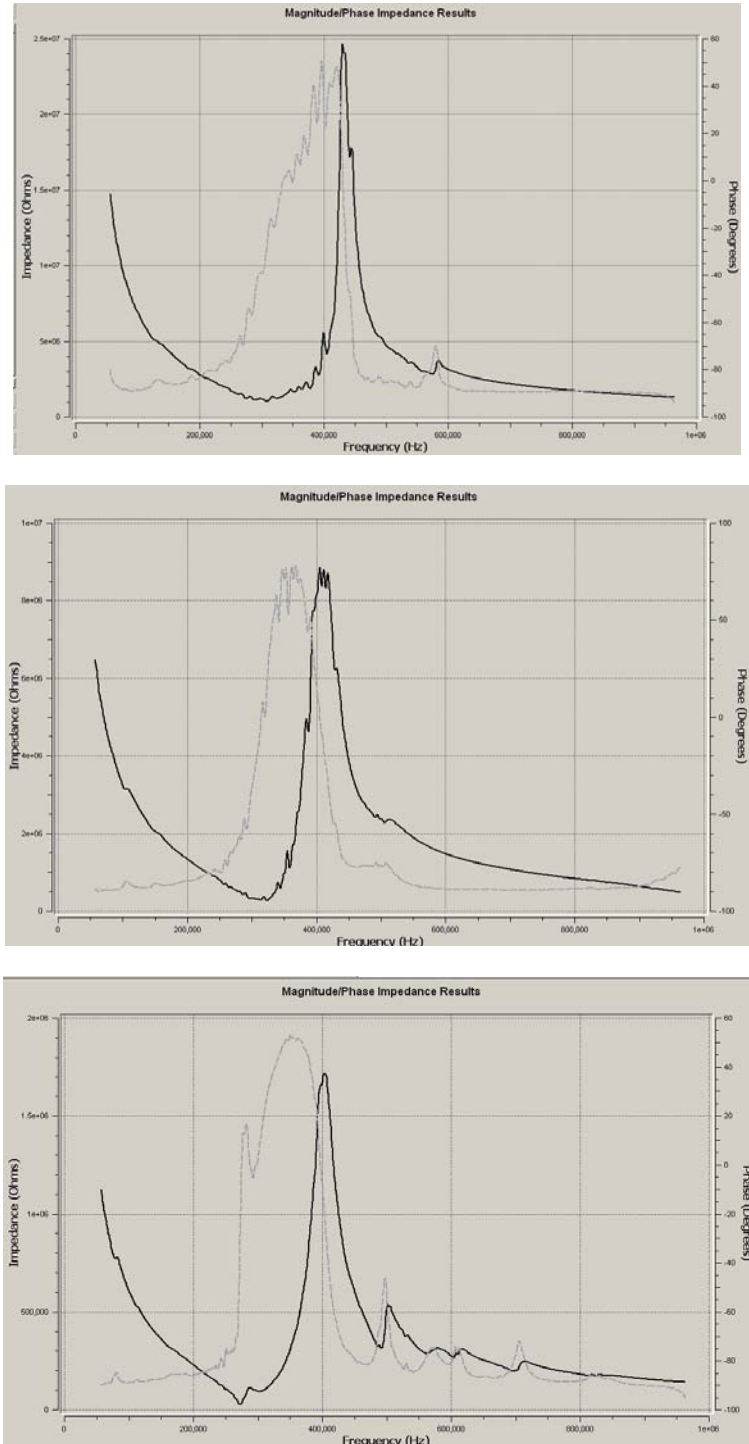
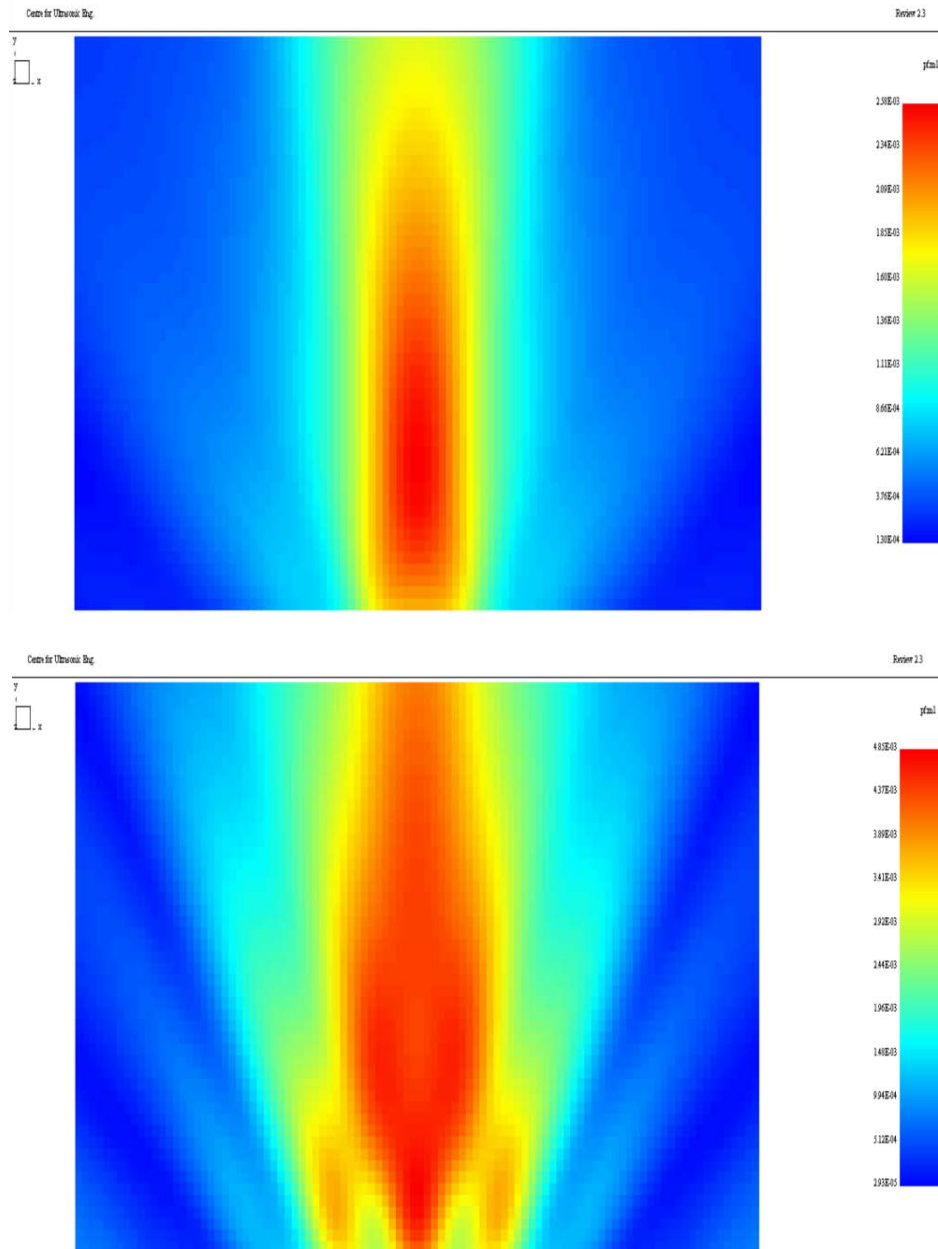
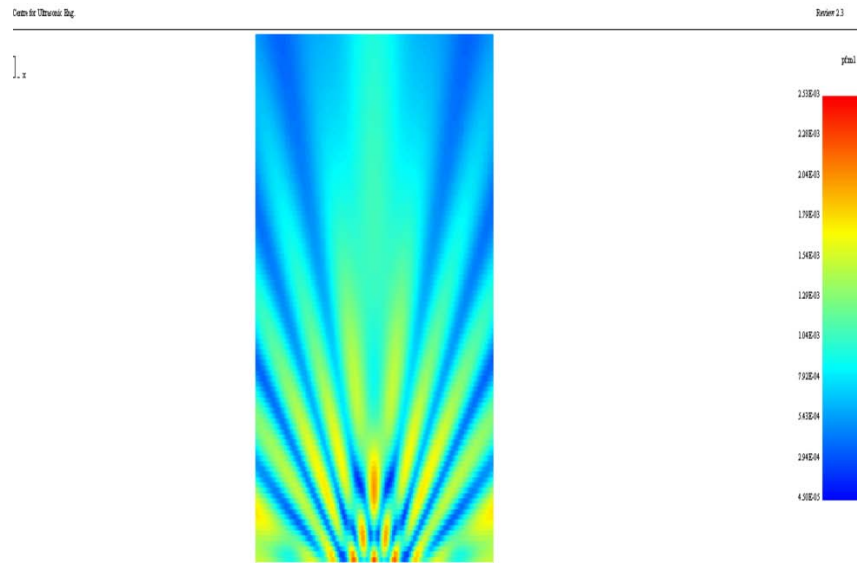


Figure 31: Impedance profiles

The impedance profiles are fairly similar with a few subtle changes as pillar width increases. The electrical resonant frequency fluctuates around the 300 kHz region, exhibiting no linear pattern with increasing pillar width. Small 'spike' interruptions in the profile are a result of the various modes that exist in the device. Other than the thickness mode, the other modes remain fairly insignificant until pillar width reaches 3.0 mm: an obvious 'spike' occurs after the main peak. This is an effect of increasing the pillar width as the lateral mode will now have a greater effect. The impedance at the electrical resonant frequency also steadily decreases as the pillar width increases.





**Figure 32: Beam profiles from each device**

The beam profiles become more complicated as the pillar width increases. This relates back to piston source section where the source radius was varying by different sizes and creating more complex interference patterns.

### 3.3.1 2D vs. 3D

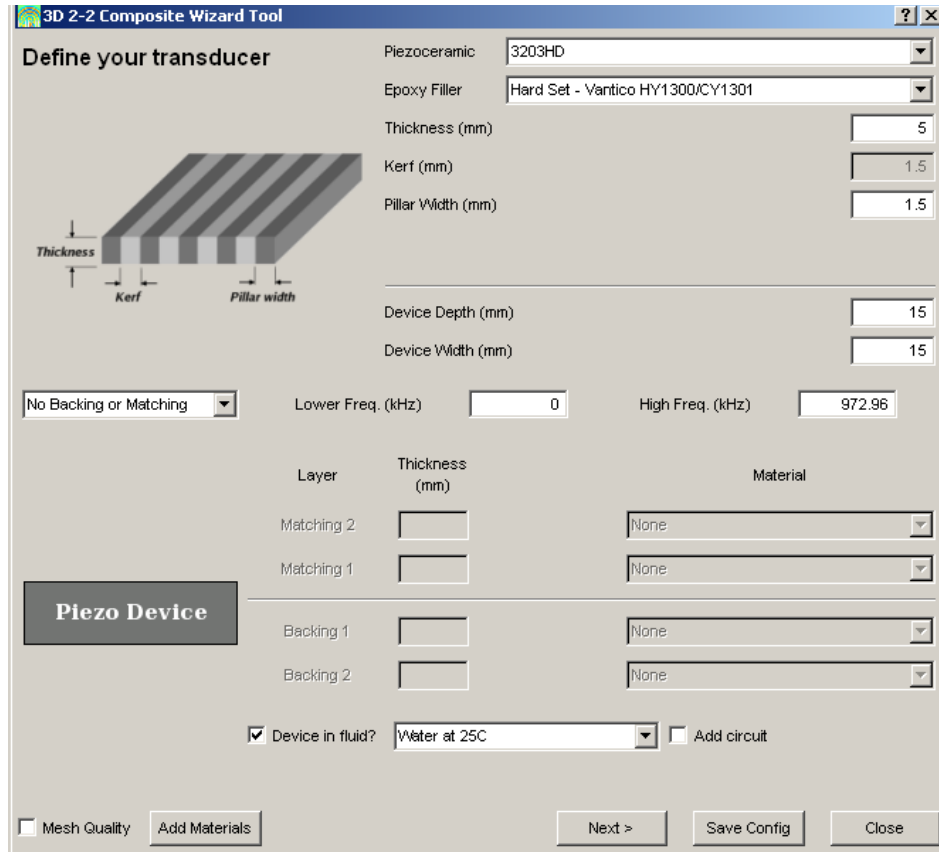
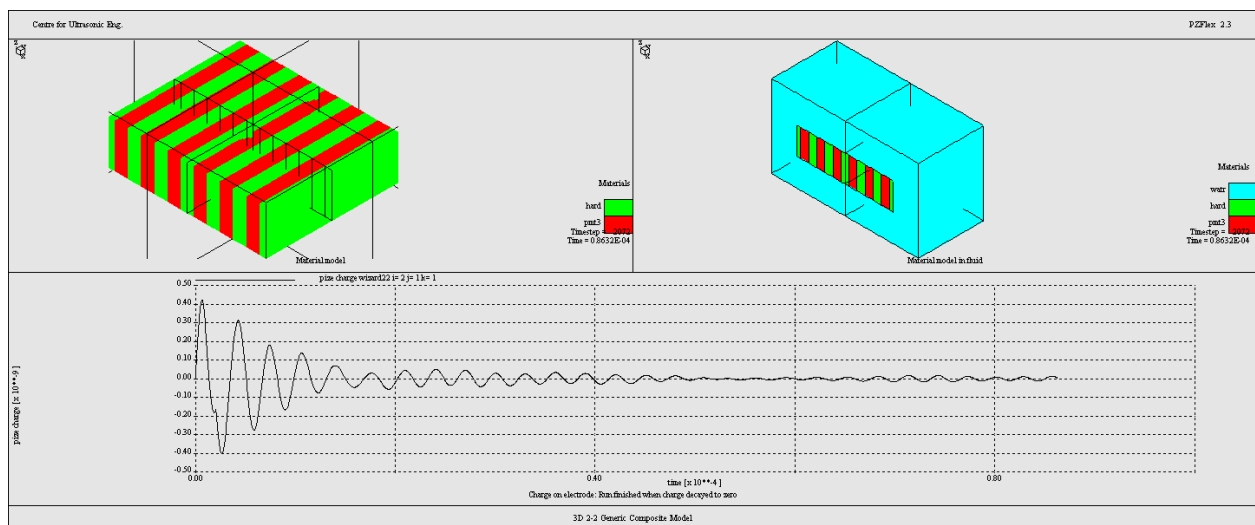
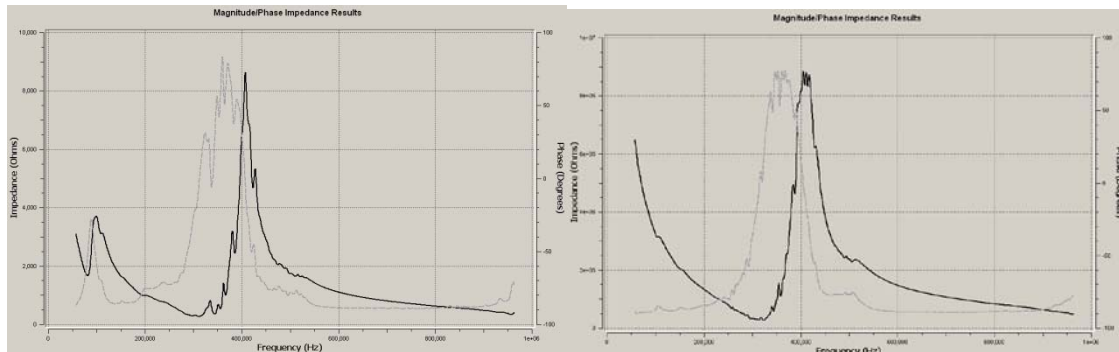


Figure 33: 3D 2-2 Composite Interface

The interface is the exact same as the 2D model. It can be assumed that the device depth parameter is not used to construct the 2D model but may be used for calculating the device's characteristics.

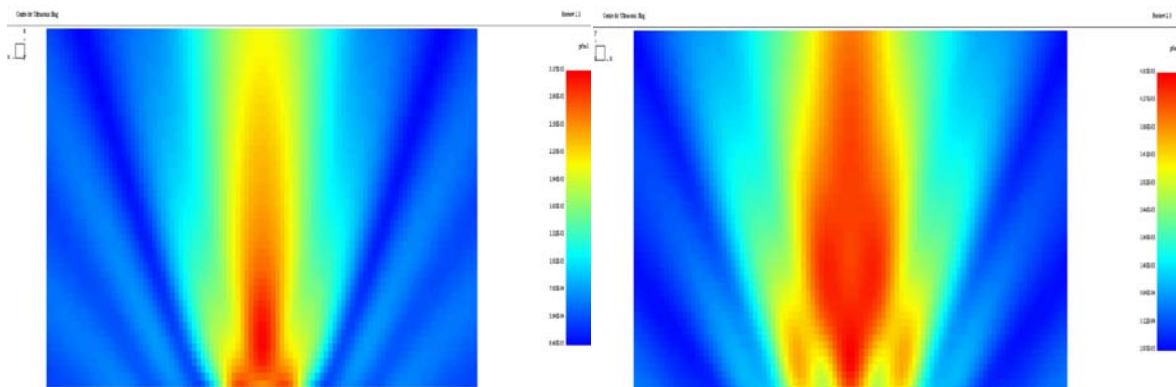


Even though the 3D model is of essentially the same device, there are aspects that the 3D model takes into account that the 2D model leaves out.



**Figure 34: Impedance profiles from the 3D (left) and 2D model**

The 3D model produces an impedance profile that is not as smooth as the 2D model shown in Figure 34. Other resonant modes become more of apparent in 3D models due actual modelling of nodal points outside of the plane in the 2D model: some modes are further amplified and new modes can be introduced. Inter-pillar and intra-pillar resonances are considered in the 3D model yielding more realistic results.



**Figure 35: Beam Profile from the 3D (left) and 2D model**

The 3D beam profile again differs from the 2D model: the high pressure region of the beam in the 3D model does not extend as far and is narrower. This is due to the 2D model using symmetry boundaries to replicate a 3D model and generate better results: the model essentially extends infinitely in the third dimension and so will differ from the more accurate 3D model where the third dimension is finite.

In general, 3D models are more accurate but are time-consuming to execute and can be complex to hard code. Depending on the geometry of the model, 2D model can generate results that are just as accurate as a 3D model.

### 3.4 Tonpilz Transducer

The tonpilz transducer is used for underwater applications such as sonar. Active material is sandwiched between a light, stiff radiating head mass and a heavy tail mass to create an underwater transmitter or receiver [4].

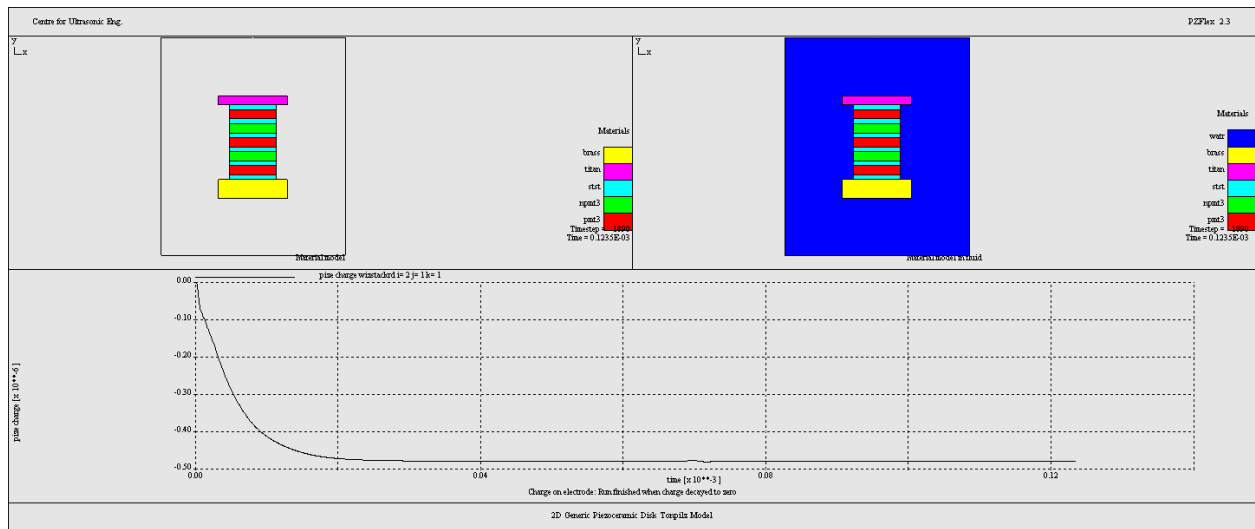
**Define your transducer**

Piezoceramic: 3203HD  
 Electrodes: Stainless Steel  
 Layer Thickness (mm): 2  
 Electrode Thickness (mm): 1  
 Pre-stress Stack (MPa): 5  
 No. of layers in stack: 5  
 Device Diameter (mm): 10  
 Tail/Head Diameter (mm): 15  
 Lower Freq. (kHz): 0  
 High Freq. (kHz): 486.48  
 Layer Thickness (mm): 2  
 Material: Titanium  
 Tail Mass: 4  
 Material: Brass  
 Device in fluid? Water at 25C  
 Add circuit

Buttons: Mesh Quality, Add Materials, Next >, Save Config, Close

Figure 36: Tonpilz Wizard interface

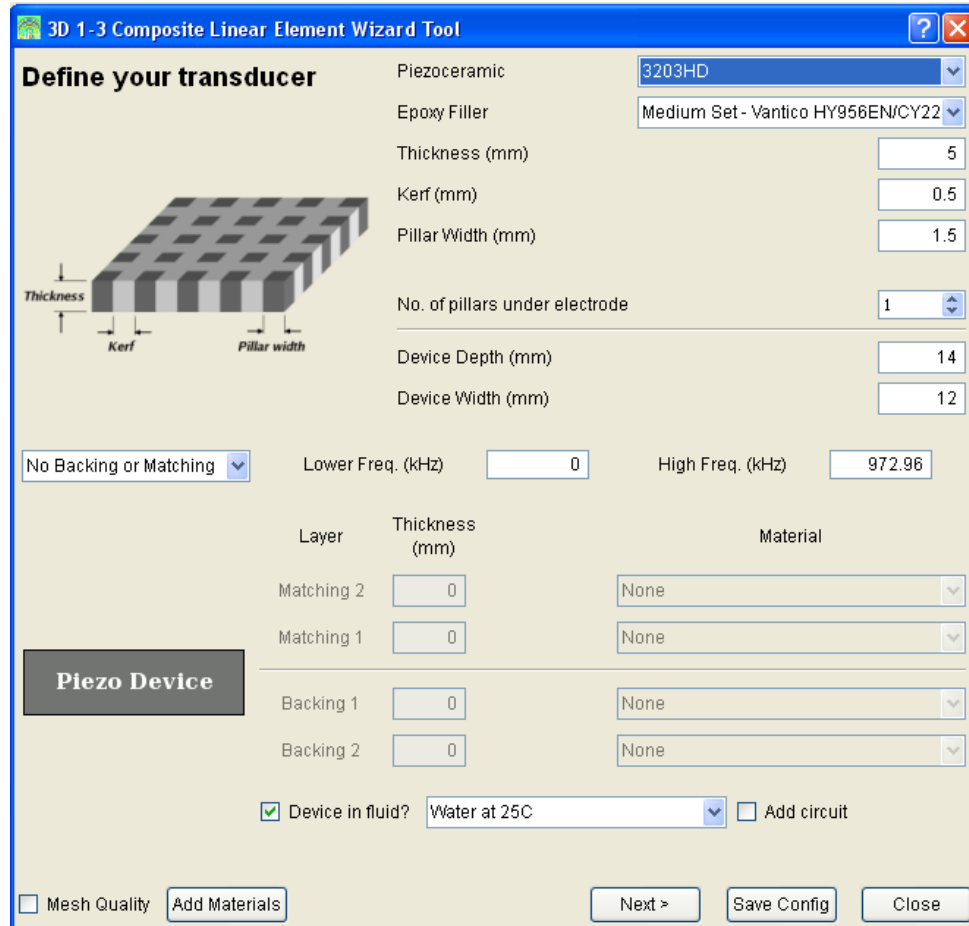
With the diagram of the device at the side of the interface, most of the input settings are made clear. The number of layers considers only the layers of active material and does not include electrode layer if it is used. The electrodes will be automatically added in between the active material and the piezo material will be poled accordingly. The pre-stress stack setting applies a compression force to combat the tensile force experienced when operating under high voltages.



**Figure 37: Tonpilz model with settings shown in Figure 36**

Running the model will generate the usual selected outputs such as the impedance profile and mode shapes.

### 3.5 2D Array Element Transducers



**3D 1-3 Composite Linear Element Wizard Tool**

**Define your transducer**

Piezoceramic: 3203HD  
 Epoxy Filler: Medium Set - Vantico HY956EN/CY22  
 Thickness (mm): 5  
 Kerf (mm): 0.5  
 Pillar Width (mm): 1.5  
 No. of pillars under electrode: 1  
 Device Depth (mm): 14  
 Device Width (mm): 12

No Backing or Matching (dropdown)  
 Lower Freq. (kHz): 0  
 High Freq. (kHz): 972.96

Layer	Thickness (mm)	Material
Matching 2	0	None
Matching 1	0	None
Backing 1	0	None
Backing 2	0	None

**Piezo Device**

☒ Device in fluid? Water at 25C ☐ Add circuit

☐ Mesh Quality

Figure 38: 3D 1-3 composite transducer wizard interface

A 3D 1-3 Composite wizard is also available, eliminating the need to hard code the complicated model. The interface resembles the other wizards with parameters that should be familiar. To adjust the number elements in the transducer model, the 'Device Depth' and 'Device Width' will need to be altered with regards to the 'Pillar Width' and 'Kerf' i.e. should be multiples of the combined dimensions. The 'No. of pillars under electrode' controls how many elements are electroded together to create in effect a single element i.e. they will be fired simultaneously. Changing this input setting may adjust the general width of the device by appending more elements to electrode together.



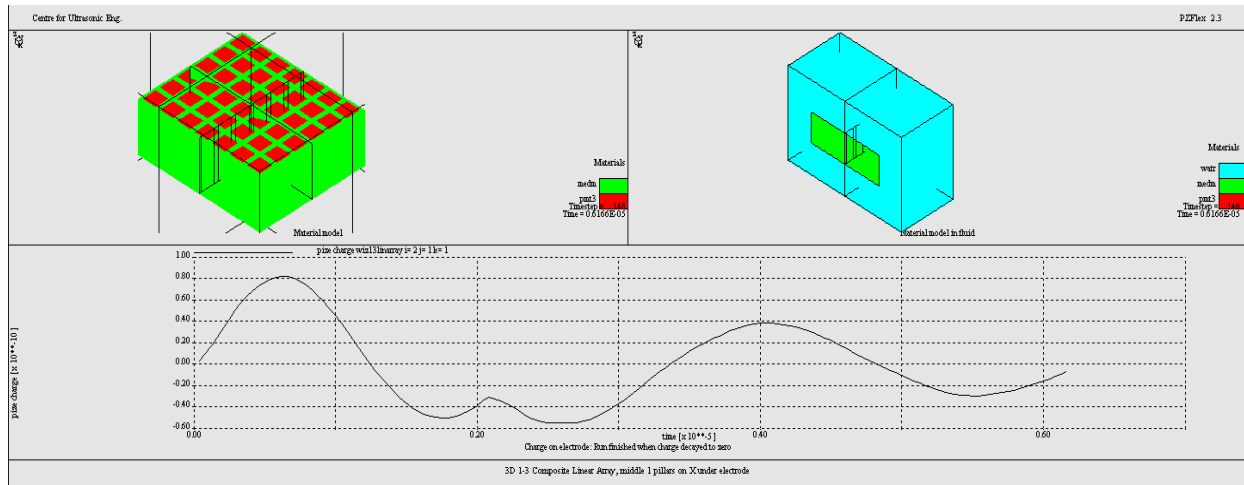


Figure 39: 3D 1-3 composite model running

Due to the method the wizard generates the model, there may be slight issues with generating the expected number elements in each row/column. The best way to account for this issue is simply a trial error process with the device width/depth settings. In practice, this is not a major concern as how they are electroded dictates if the elements are active or not.

## **Section 4 – Phased Delay Laws**

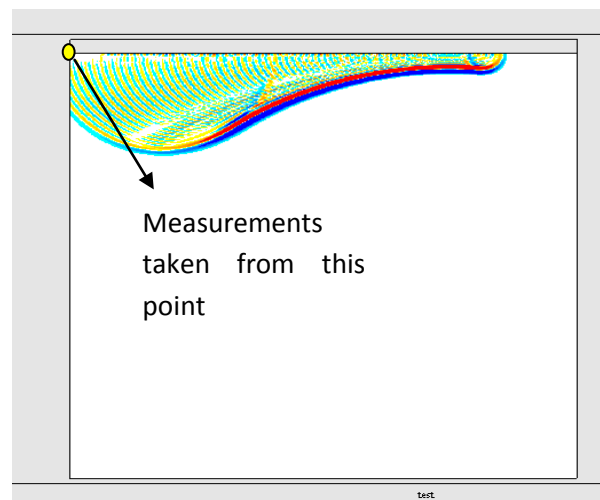
Delay laws are time delays applied to transducers with several active elements (array transducers) to control where the beam is focused. Each separate element has a different path/time to reach the desired focal point. By applying a delay to when each element is active, the difference in each elements path can be compensated for i.e. each element will transmit a wave, arriving at the focal point at the same time to constructively interfere.

To gain a better understanding, the following sections will analyse some of the methods in generating code to calculate delay laws for different scenarios.

### **4.1 Stainless Steel Block**

The delay laws were to be calculated for a simple a 2-2 piezocomposite device to generate a focus in a stainless steel block. A PZFlex input file was used to implement the program for the delay calculations.

For this simple scenario of focussing the device in a stainless steel block there needs to be a few parameters first defined before the delay laws can be calculated. This includes: wave speed in the material; focal point and general transducer information (number of elements and dimensions). Assumptions that were made when implementing the delay code were: the transducer was situated at the middle of the steel block and focal point dimensions were taken from the top left corner of the steel block.



**Figure 40: Model of a steel block with delay laws applied to transducer**

One the main tasks of the program was to obtain the distance which the wave from each element covers to reach the focal point. With this information, the longest time taken to reach focal point can be calculated and consequently, be used to calculate the delay law for the remaining elements. To re-iterate, all the waves must meet at the same time at the focal point. The mathematics behind this delay law program consists only of trigonometry and the speed-distance-time formula so it should be straightforward to understand.

Firstly, the exact position of the first element was calculated as it allowed the use of a loop to find the position/dimensions of all the other elements due to its linear layout. With the position of first element established, trigonometry was then used to work out the direct distance between the focus and element. This was simply repeated for the remaining elements.

With the distance information stored in an array, the time taken to cover the distance (flight time) was calculated. And using SYMB #get, the maximum time taken was retrieved from the array which all the delay laws were calculated from. Simply by using this value and subtracting the flight time, the delay applied to each specific element could be stored into a text file and read into an existing model.

```

symb #read DelayLaws.txt

symb i_elem_start = 2          /* start indice element 1
symb i_elem_end = $n_elem * 2  /* start indice element 2

symb JIND = 1
                                /* dummy delay to test code

plod
    vctr vct 0 1 0              /* set vctr for multiple use

do loop I $i_elem_start $i_elem_end 2

symb delay = $ELEMENT$JIND

symb istrt = $I
symb istop = $istrt + 1

plod
    pdef pld$I func 1. $delay    /* delay read in from file
    sdef pld$I vct $i$istrt $i$istop $j_array $j_array
    end

symb JIND = $JIND + 1

end$ loop

```

#### Code 27: Reading data into actual model file

With the text file created, the data could be accessed in a separate Flex input file by using the #read command. The PDEF sub-command under PLOD section of code is where the delays should be applied to in the model: the fifth parameter allows a time-shift/delay to be applied before the driving function . As long as the delays are labeled in a logical fashion, a do loop can be used to apply the delay laws to all the elements efficiently.

## 4.2 Wedge and Stainless Steel Block

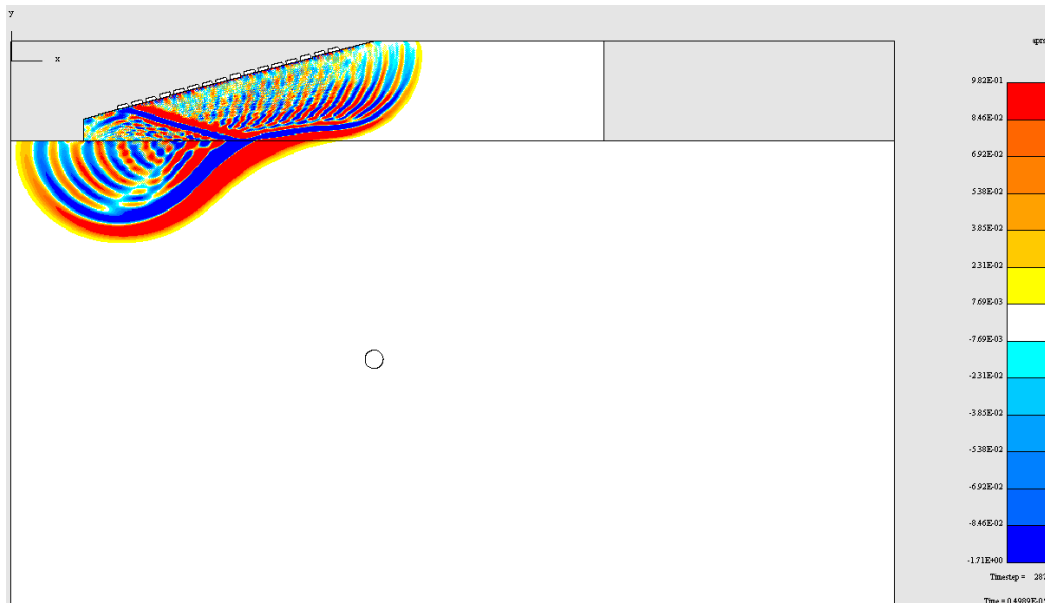


Figure 41: Model of 16 elements firing into a wedge and block of material

In order to calculate the delay laws, the wave path travelling from each element must again be determined. However, this is slightly more complex as there are now two types of material which will cause the wave to refract at the material boundary. Therefore, the path of each wave is not a simple straight line towards the focal point.

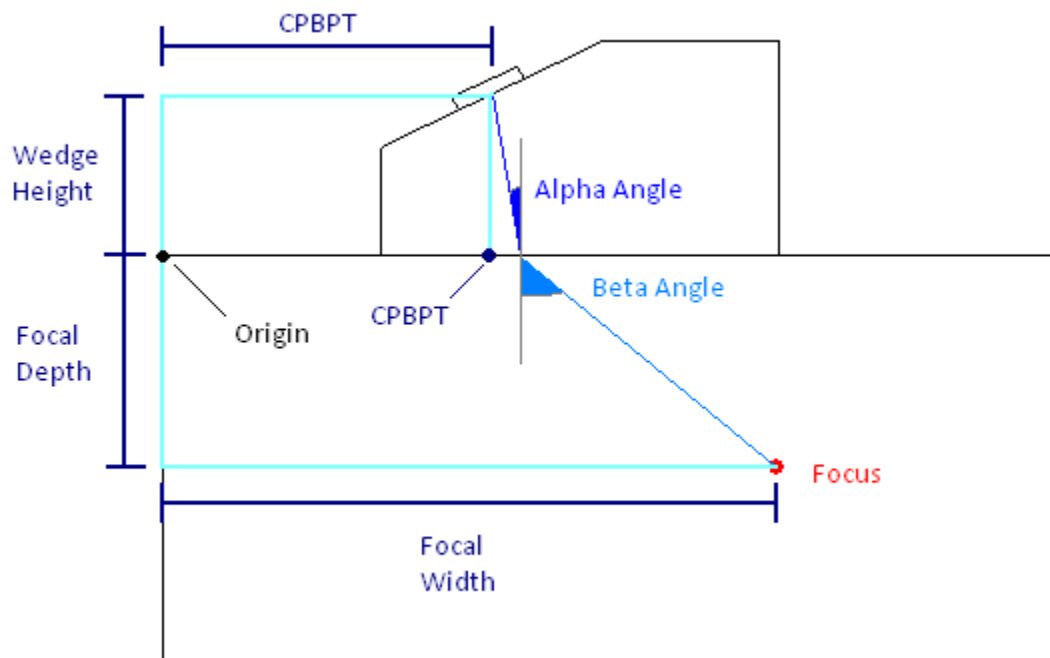


Figure 42: Diagram of model with available information in darkest blue text

Another issue is that there is limited information about the model: the focal point and the position of the transducer are known. The first task here will be to find the angle which the main beam needs to be incident at the boundary in order to refract and arrive at the focal point. The assumption at this point is that the transducer is a single element situated at the middle of the shoe (angled part of the wedge). Using trigonometry, equations were derived to solve one of the angles (Alpha). The following symbols will be used in the equations:

- $X_F$  – Focal width
- $Y_F$  – Focal depth
- $X_1$  – CPBPT distance
- $Y_1$  – Wedge Height
- $X_2$  – Distance between the CPBPT point and where the main beam meets the boundary (MBpt)
- $X_3$  – **Horizontal** distance between MBpt and focus
- $\alpha$  – Alpha angle
- $\beta$  – Beta angle
- $V_1$  – Wave velocity in wedge
- $V_2$  – Wave velocity in block

$$1. \tan(90 - \alpha) = \frac{Y_1}{X_2} \Leftrightarrow X_2 = \frac{Y_1}{\tan(90 - \alpha)}$$

$$2. \tan(\beta) = \frac{X_3}{X_F} \Leftrightarrow X_3 = X_F \tan(\beta)$$

$$3. X_F = X_1 + X_2 + X_3$$

$$4. X_F - X_1 = X_2 + X_3 = \frac{Y_1}{\tan(90 - \alpha)} + Y_F \tan(\beta)$$

$$5. 0 = -X_F + X_1 + \frac{Y_1}{\tan(90 - \alpha)} + Y_F \tan\left(\frac{V_2}{V_1} \sin(\alpha)\right)$$

There is one value of  $\alpha$  that will satisfy equation 5. The equation can be differentiated to find  $\alpha$  but may be a little complicated with the trigonometric functions. Instead, code was written to substitute in all possible angles up to the critical angle for  $\alpha$  and the angle which, yielded a value closest to zero, would be the angle of incidence ( $\alpha$ ).

With  $\alpha$  known,  $\beta$  was calculated. The angles only apply to the main beam i.e. from a single element located at the center of the shoe. Each element of the 2-2 piezocomposite transducer will have a slightly different angle of incidence as they will be offset from the center of the shoe. In order to calculate the various angles, a theoretical focus point must be established. If the wedge and the block were all same the material, the ultrasonic beam would not refract and simply continue to focus to a point with straight paths leading back to each of the elements: this is the theoretical focus point. Knowing this point will allow the angle of incidence of each element to be determined and as a result, allow all the necessary distances of the wave paths to be calculated.

After obtaining the distances, the program follows the same process of calculating the delay laws as in '4.1 Stainless Steel Block'.

#### 4.2.1 Batch File

Usually a batch file is used to run several simulations of the same model with varied settings without the need to re-write code each time. One of the main advantages of using a batch file is that it can be controlled from the one file. Using this aspect of a batch file, both the delay law and the wedge model file can be modified and executed from the one file. The batch file has become essentially a control file.

Both the delay law and wedge model file use the same variables for either calculations or generating the model which can be transferred to the control file. Any changes to the parameters in the control file will reflect in both the delay law program and wedge model.

```
symb #submit pzflex WedgeDelayLawsBatch

symb #msg 1
Delay Laws Calculated - 't' or term to continue
t

symb #submit pzflex WedgeDelayLawModelBatch
```

Code 28: Code for executing external files

The SYMB #submit command is used to execute any external files. The first parameter is the file type that will be run which is a 'pzflex' input file. For executing review files, use keyword 'review'. The last parameter is simply the name of the file without any file extensions appended at the end (NO .flxinp or associated file types).

```
symb #read symb.wedgeBatch /* read in controller variables
```

Code 29: Included in the files to be executed by control file

Code 29 should also be included in the delay law and wedge model file to have access to the parameters set in the control file. Running the control file should now be the exact the same as running the delay law and wedge model separately.

NOTE: Control file was written as a review file and not a flex input file so execute the file by using the 'Run Review' icon. Issues with the other files are not shown in the control file's print file so make sure the other files are working 100%.

## Section 5 – ‘How to’ s

### 5.1 Arrays

#### 5.1.1 Creating an array

```
data
  open zerocross 2 $arraysize 1 0 f
end
```

Code 30: Creating an array to store data

The DATA >> OPEN is used to create an array of any size to store: integers, floating point numbers and characters. In Code 30, an array named ‘zerocross’ is created with ‘2’ dimensions. The following three parameters are the size of the IJK indices. As it is only a 2D array, there will be ‘arraysize’ number of I-indices and ‘1’ J-index creating a single column of ‘arraysize’ data elements. The last parameter selects the type of data to be stored: in this case, ‘f’ represents floating point data.

#### 5.1.2 Setting values in an array

To set a value in an array is fairly straight forward. Using the SYMB command and ‘#set’, the value of ‘Delay’ is stored in an array called ‘FlightTime’ at the location: J – 2 – 1 which are the index positions of the IJK vectors respectively.

```
symb #set $Delay FlightTime $J 2 1
```

Code 31: Setting a value in an array

#### 5.1.3 Returning values from an array

```
symb #get { TempDelay } array FlightTime $K 2 1
```

Code 32: Returns a value from an array to a variable

The SYMB command is again used to retrieve a data from an array. After the ‘#get’ function, a variable is declared within curly brackets to store the retrieved data. The function type is then declared using the keyword ‘array’. There are also other useful commands such as ‘datamax’ which automatically retrieves the maximum value. For a list of the various keywords used in the #get command refer to the manual. The last commands are simply defining the name of the array and the index position to obtain the data from.

### 5.1.4 Creating Files to output data

```
symb #msg 3 > DelayLaws.txt /*3 Lines of message
symb *****
symb DELAY LAWS
symb *****

do Loop3 K 1 $Elements

symb #get { TempDelay } array FlightTime $K 2 1

symb #msg 1 >> DelayLaws.txt
symb ELEMENT$K = $TempDelay

end$ Loop3
```

Code 33: Creating and text file and writing to file using a loop

The first line in Code 33 creates the text file to store the data and the following 3 lines appear in the text file. This coincides with the number 3 in the first line of code. The key operator to creating the text file is the '>' sign which writes the declared number of lines to the file defined. Using a do loop, we access information stored in an array and using the '>>' operator, the following line of code is appended to the file.

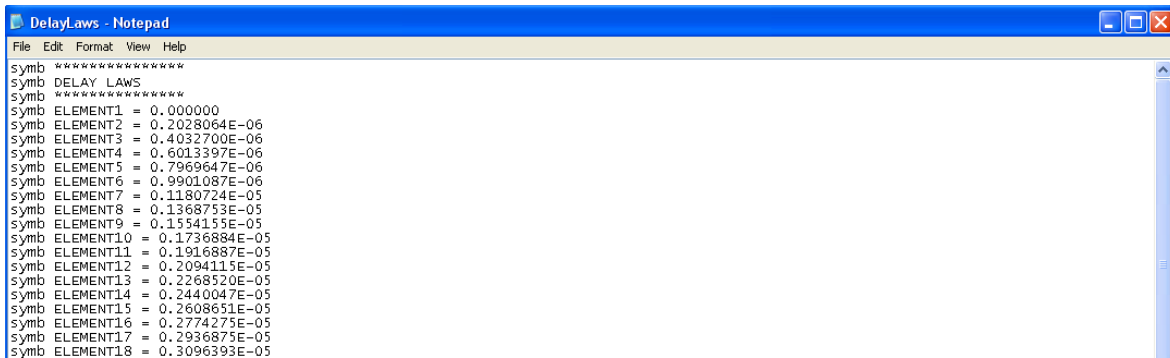


Figure 43: Showing created text file with stored data



## 5.2 Do Loops

```
do Loopio I 1 $arraysize 1

symb points = -$focalWidth1 + $cPBPT
symb #set $points zerocross $I 1 1
symb aang = $aang + $steps
goto Loopio if $aang le $critical

end$ Loopio
```

Code 34: Syntax for a do-loop

Loops are extremely useful in any programming language. Simply start the loop with the DO command followed by the constraints controlling the loop. Give the loop a name ('Loopio') and set up a variable to be used to control how many times to run the loop. This variable will only exist in this loop. The following 2 parameters are the values which the variable will start and finish on. The last parameter controls the value used to step through the range of values i.e. incrementing by 1 each loop cycle from 1 to 'arraysize'. The code is then inserted within the loop to be executed repeatedly. At the end of the loop, it always finishes with 'end\$' and the name of the loop. A useful tool to use is the 'goto' command which allows the loop to finish prematurely if a criterion is met. In Code 34, the criterion is simply when the variable 'aang' has become less than 'critical'.

## 5.3 If Else Structure

```
if ( $cPBPT2 le $focalWidth1 ) then
symb BP2 = $cPBPT2 + $Opposite2
else
symb BP2 = $cPBPT2 - $Opposite2
endif
```

Code 35: An If-Else Structure

The 'if-else' structure is another common tool in many programming languages and they all more or less follow the same syntax. The IF command is used in conjunction with a condition in brackets followed by the keyword 'then'. If the condition is true then, the code within this section is executed. If the condition is false, the code within the ELSE section is executed. The structure is terminated by 'endif'. More conditions can be added by using the ELSEIF command, creating an 'if-elseif-else' structure. The 'elseif' follows the same syntax as the 'if' line of code. These structures can be nested within one another i.e. 'if-else' structure within the code of another 'if-else' structure.

## 5.4 Using the FUNC command

### 5.4.1 Introduction

The FUNC command is used for selecting the type of driving function to load the model. Functions include: Blackman Harris; chirp; Gaussian, sine wave; step; wavelet and time delay. PZFlex will even allow your own functions to be imported from data files.

To grasp how the FUNC command works, the sine wave function will be used to show how changing different parameters will have an effect on the input. For each type of function, there will be a set of parameters governing its characteristics i.e. frequency, amplitude and phase. Using SINE as an example, the available parameters for the user to control are:

***func sine <frequency> <amplitude> <phaseshift> <nperiod/ncycles>***  
***<valueadd> <rampcycle> <tdelay>***

Not all parameters need to be addressed when using the sine function as there are usually default values stored. 'valueadd' can be used to offset the function on the amplitude axis; 'rampcycle' allows the user to specify how many cycles it takes to reach maximum amplitude in a smooth fashion and 'tdelay' is simply a time delay applied to the function.

NOTE: All waveforms obtained from time histories of the input function using the INSIGHT tool in PZFlex

## 5.4.2 Continuous Sine Wave

```
func sine $freqint 1. 0. 0.
```

Code 36: General syntax of a sine function

Beginning with the FUNC primary command, we then specify the type of function that will be used: in this example we are using SINE. The parameter to define next is the frequency which will take on the value given to the symbol 'freqint' (1 MHz). The amplitude of the wave is '1'. There is '0' phase shift applied to the sinusoid and by setting the number of periods/cycles to '0', a continuous wave is generated over the simulation period of the model.

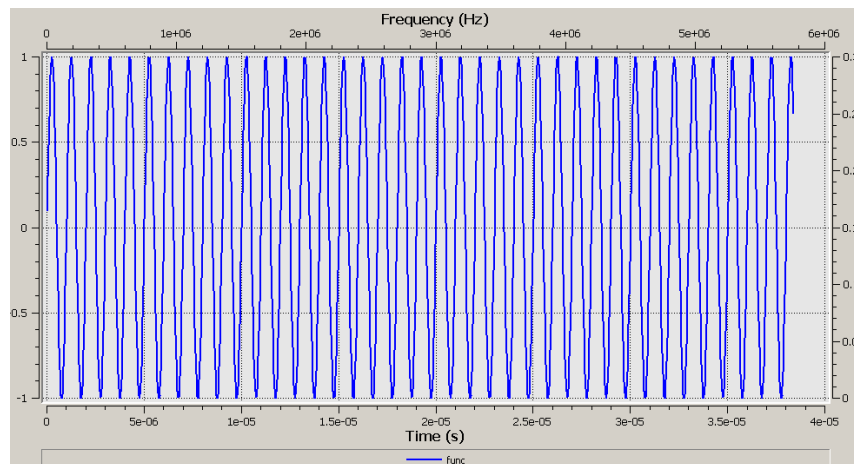


Figure 44: Continuous input wave function

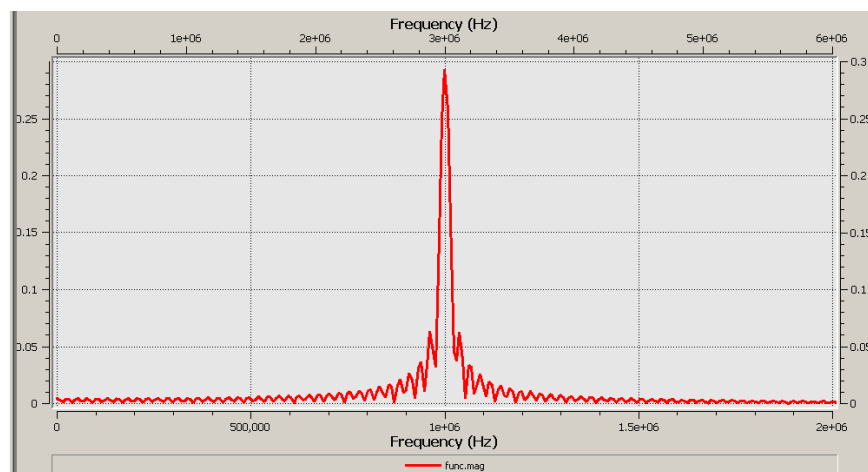


Figure 45: Fast Fourier Transform form input to show fundamental frequency of 1MHz

### 5.4.3 Sine Impulse

A commonly used method of characterising a system is to use an *impulse* as the input function. This allows frequency data up to the frequency of the impulse to be extracted which is highly useful when obtaining beam profiles. All the parameters of the sine function remain the same as in section '5.4.2 Continuous Sine Wave' except from the number of periods/cycles.

```
func sine $freqint 1. 0. 1.
```

Code 37: Changing continuous sine wave to an impulse

The value of '1' is now set for the number of cycles and so only one period of a 1 MHz sine signal should be the input function.

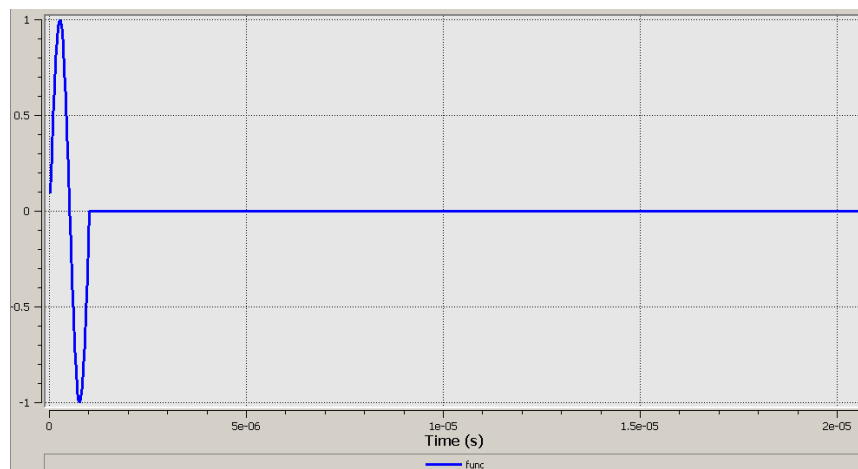


Figure 46: Impulse function consisting of a single sine wave cycle

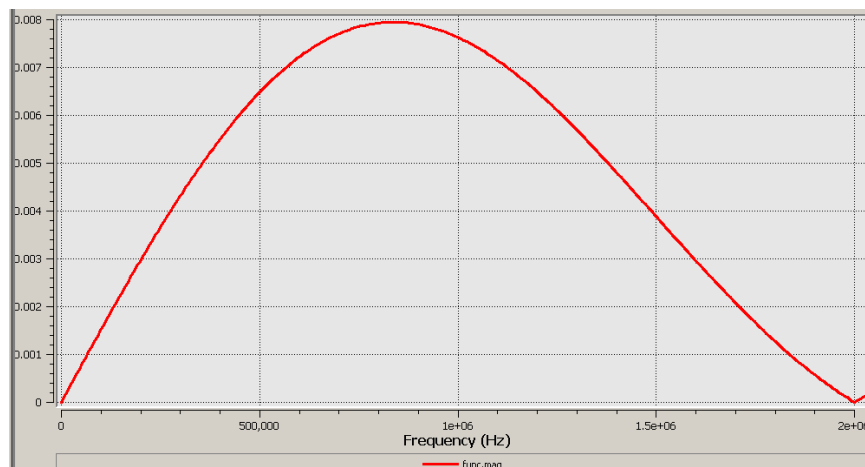


Figure 47: FFT of the sine wave impulse

#### 5.4.4 Three Cycle Burst with Phase Inversion

In this example, both the 'phaseshift' and 'ncycles' parameter are tweaked to show its effect.

```
func sine $freqint 1. 180. 3.
```

Code 38: Modified phase and cycles parameters

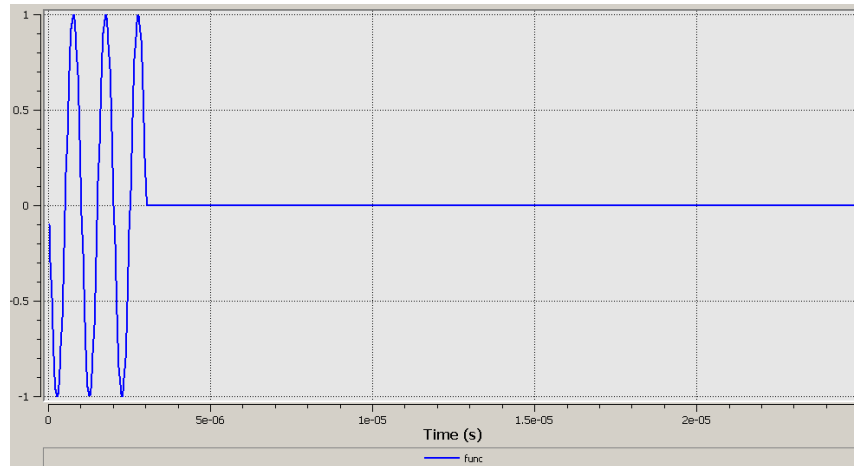


Figure 48: 3 cycles of a sine wave with a 180 degree phase shift

The sine wave now begins half way through a cycle and repeats for 3 whole cycles.

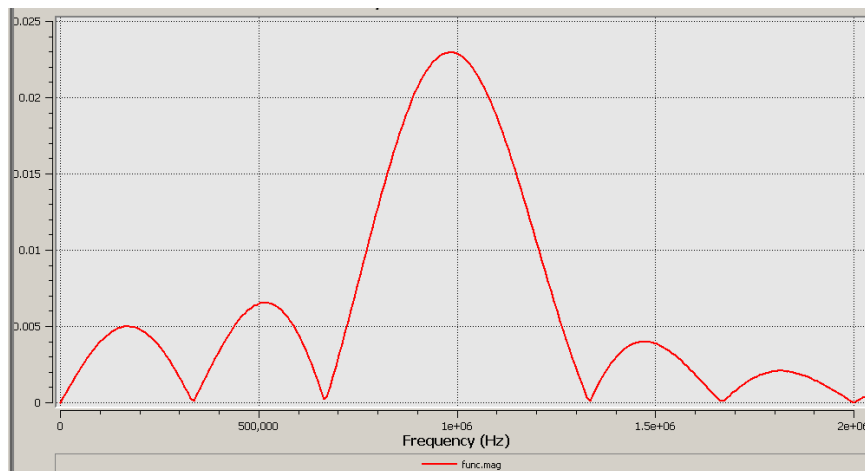


Figure 49: FFT of the 3 cycle burst in Figure 48

### 5.4.5 Ramp-Up 3 Cycle Burst with Time Delay

Continuing with exploring the different parameters available for the SINE function, we will try using the 'rampcycle' and 'tdelay'.

```
func sine $freqint 1. 0. 3. 0. 3. 3.e-6
```

Code 39: Testing further parameters in SINE function

The code above should yield a 3 cycle sine wave burst with peak amplitude of '1'. The ramp-up will occur over the 3 cycles, reaching peak amplitude in the 3<sup>rd</sup> cycle and will begin after a 3 microseconds time delay.

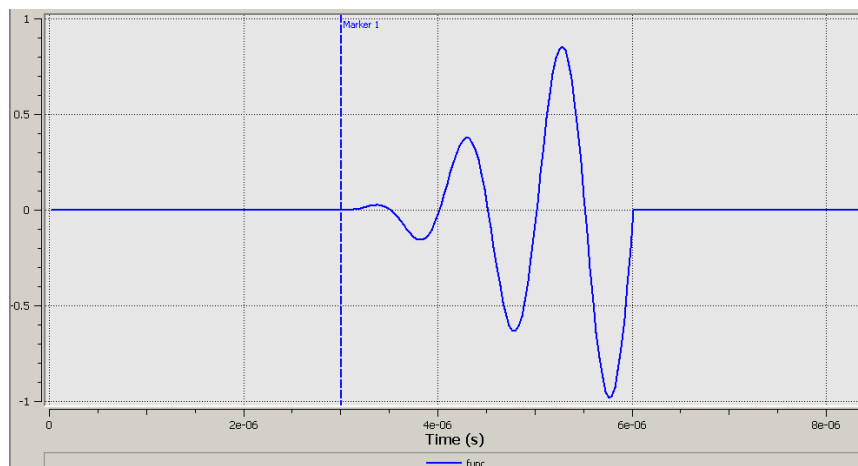


Figure 50: waveform showing ramp-up of sine wave and time delay in use

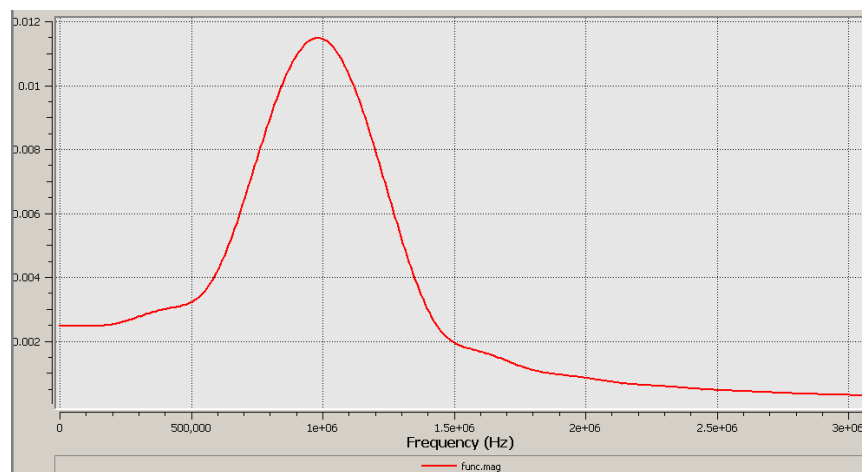


Figure 51: FFT of ramp-up SINE function

### 5.4.6 Blackman Harris

When using a different function, all that is required is to identify the parameters that will change the fundamental properties. For the BLAK function there are only 3 parameters: centre frequency, amplitude and time delay. Time delay does not need to be stated in the line of code as a default value of zero has been stored. The Blackman Harris is a window function applied to a sinusoid. A window governs the amplitude of the signal and is used to reduce spectral leakage.

```
func blak $freqint 1.
```

Code 40: Defining Blackman function

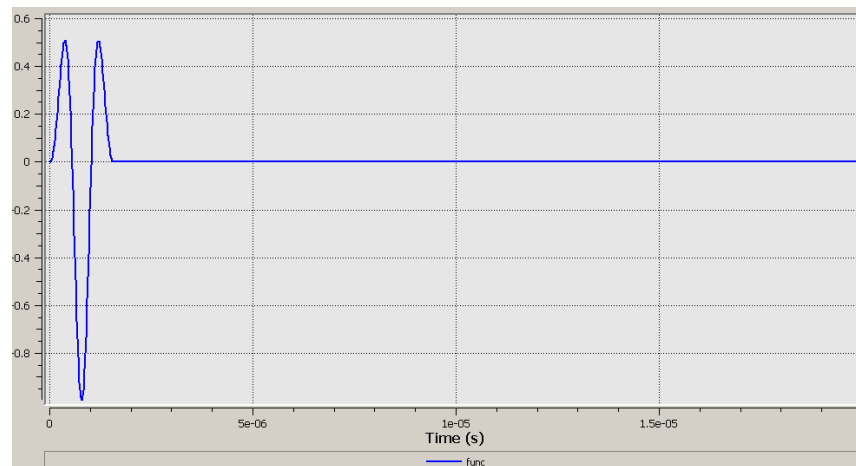


Figure 52: Blackman Harris input signal

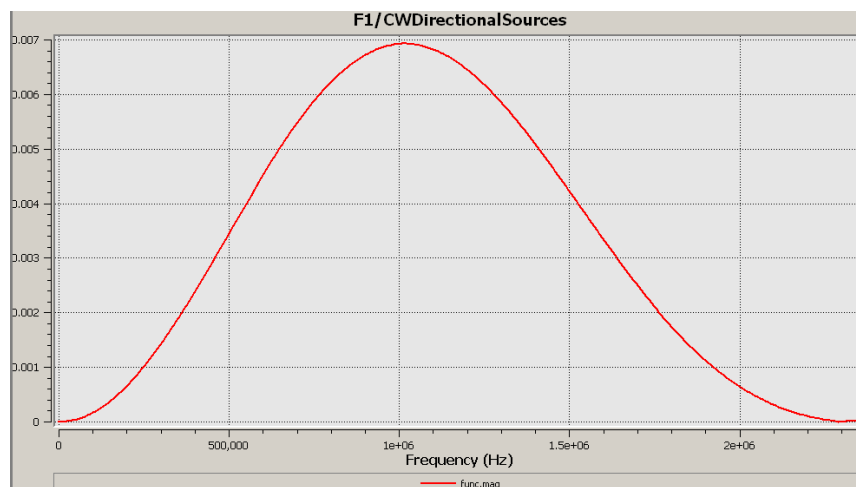


Figure 53: Frequency content of the Blackman Harris signal

### 5.4.7 Wavelet

Wavelets are of great use when it comes to signal processing. They have specific properties that when they are convolved with an unknown signal, information can still be extracted. Again, using the wavelet function is similar to all the others: enter the function's name followed by the appropriate parameters which, in this case, are frequency and amplitude.

```
func wvlt $freqint 1
```

Code 41: using the wavelet function

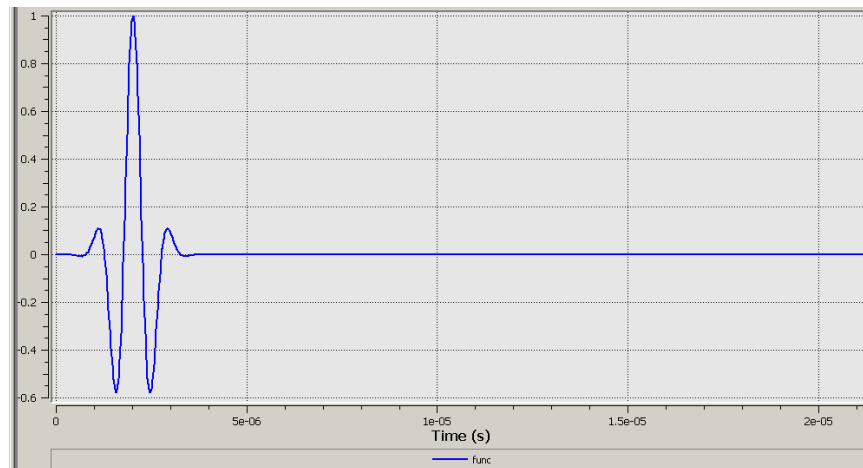


Figure 54: Wavelet waveform

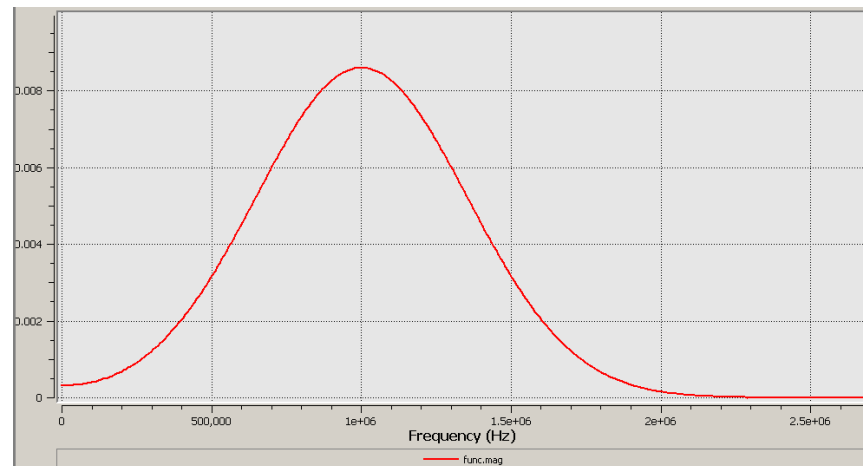


Figure 55: FFT of wavelet



### 5.4.8 Chirp

A Chirp signal is a waveform which changes frequency linearly as time progresses. An up-chirp will steadily increase in frequency and a down-chirp will steadily decrease in frequency. A chirp signal is a great tool for generating a wideband response due to its ability to cover a wide range of frequencies.

```
func chirp 1. 0. 500e3 1.5e6 10e-6
```

Code 42: Declaration of a chirp signal

The chirp function has a few more parameters to set than usual: amplitude; time-shift; starting frequency; ending frequency and time duration.

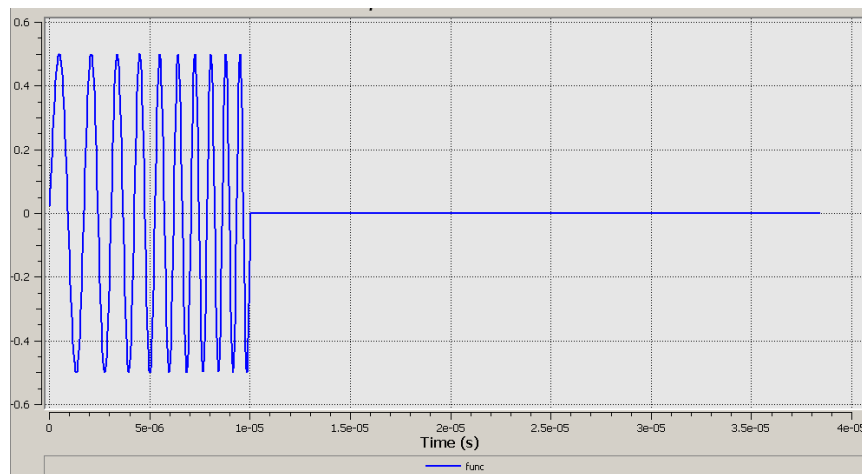


Figure 56: Up-chirp signal beginning at 500 kHz to 1 MHz

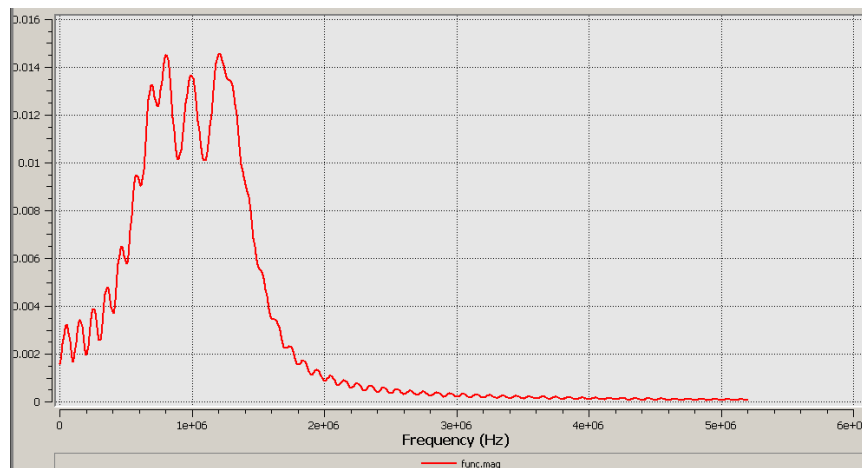


Figure 57: FFT of the up-chirp in Figure 56

## 5.5 Using the PLOD command

### 5.5.1 Introduction

The PLOD command is used to apply a pressure load to a specific set of nodes in the model. Generally, it is used in conjunction with the FUNC command where the user has defined a specific driving function to be applied.

PLOD is the primary command which allows the use of all the subcommands associated with it. The list of subcommands include: PDEF, SWEP, SPOT, VECT, SDEF, SDF2, CYLN, SPHR, SNGL, GCON, LDEF, CHEK, DACT and PRNT. Like any other subcommand in PZFlex, there is a set of parameters that controls its functionality. Since PLOD is concerned with applying a pressure load, one of the main set of parameters common to most of the subcommands is the location of where the load is applied, therefore, nodal coordinates will usually be entered.

How to use PLOD will be demonstrated through basic example code and, by applying different parameters to the various subcommands, their effects can be highlighted. The examples will only focus around the PLOD function only but may refer to other related sections of code.

### 5.5.2 Wave Applied from Sides of Model

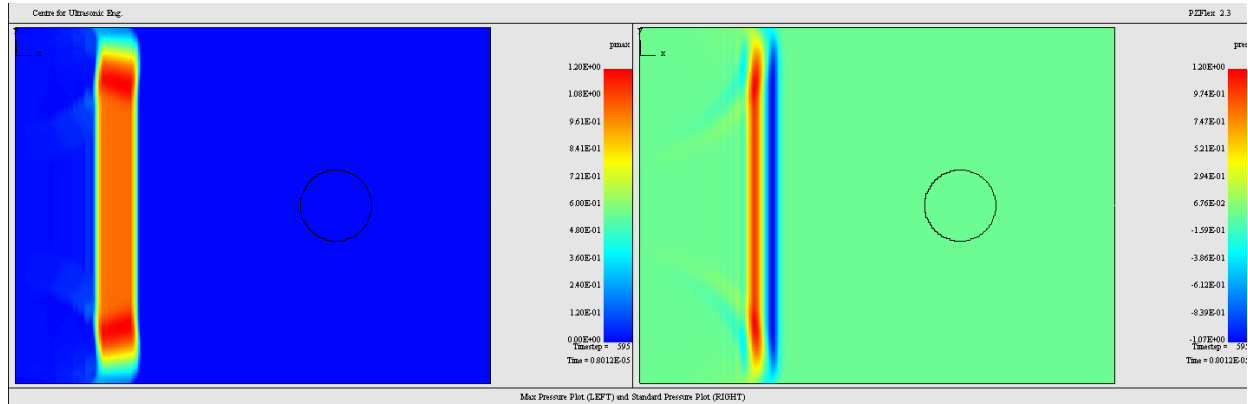


Figure 58: Pressure loaded into the left hand side of the model

The simplest example to begin with is to apply a pressure wave to one side of the model as shown in Figure 58. A single cycle sine wave has been used as the driving function defined using the FUNC command.

```
plod
  pdef pld1 func
  vctr vct1 1. 0. 0.
  sdef pld1 vct1 $i1 $i1 $j1 $j3
end
```

Code 43: Applies pressure load to the model as shown in Figure 58

**pdef <plodname> <pressureTHname>**

PDEF is used to specify a pressure history (data) to be loaded into the model. For this example, only the first two parameters are required to be entered: name of the pressure load ('pld1' – user controlled) that will be applied and the name of the pressure time history assigned to this pressure load. When the FUNC command is used for the driving function, 'func' should be entered as the pressure time history which is the 2<sup>nd</sup> parameter.

**vctr <vectorname> <vectorX> <vectorY> <vectorZ>**

VCTR defines the direction of the positive pressure vector. This is purely the direction which the user wishes to apply the wave to generate positive pressure values. The VCTR subcommand requires a name for the vector and a value 0 or (+/-) 1 for each of the X, Y and Z parameters respectively: these values are with regards to the XYZ axis at the origin of the model. For this example, the pressure load will be applied to the left side of the model therefore, the wave should propagate left to right in the positive x-direction which is set by the value of '1' after 'vct1', the name of the vector. The remaining Y and Z

parameters are set to '0' as no components of the wave are travelling along these axes. The Z parameter should only take the value of '1' for 3D models.

***sdef*** ***<plodname>*** ***<vectorname>*** ***<ibegin>*** ***<iend>*** ***<jbegin>*** ***<jend>***

With the VCTR and PDEF command set, the SDEF command can be used to apply the pressure load to a specific surface. Both the VCTR and PDEF will always precede the SDEF command. The SDEF subcommand uses the previously declared pressure function name and vector name which is the reason why they need to precede this command. After entering the name of the pressure load and vector respectively, the IJK coordinates which dictates the set of nodes the pressure load is applied to should be entered in the usual fashion. The two I-coordinates define where the pressure load should start and finish along I (or X) direction and similarly for the J-coordinates. All the necessary parameters have been entered to generate the pressure wave and should look **similar** to Figure 58.

### 5.5.3 Wave Applied from different side of Model

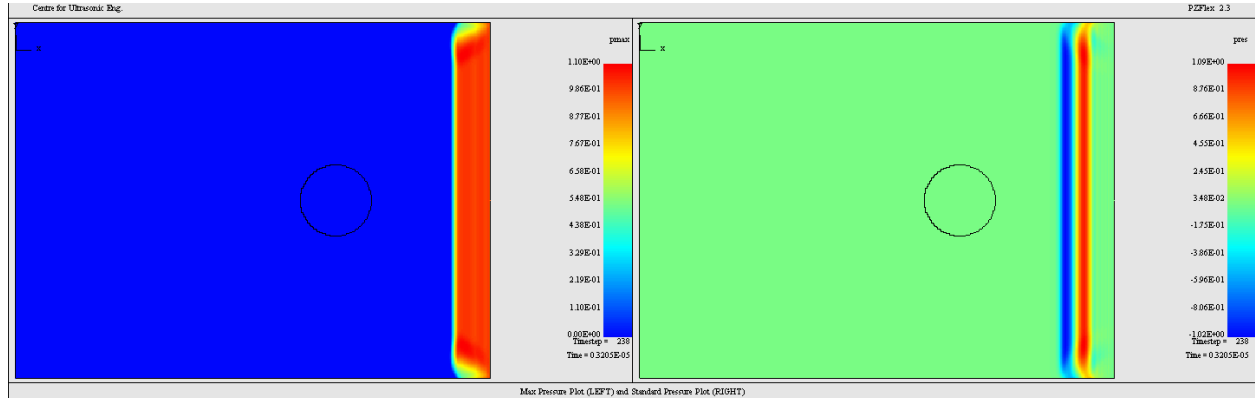


Figure 59: Pressure wave applied to right side of model, propagating in the -x direction

Changing the surface and direction of the applied pressure wave is a straightforward task of manipulating the parameters of the subcommands used in the 5.5.2 Wave Applied from Sides of Model.

```
plod
  pdef pld1 func
  vctr vct1 -1. 0. 0.
  sdef pld1 vct1 $indgrd $indgrd $j1 $j3
end
```

Code 44: Changing vector and IJ coordinates

If the wave is to be applied from the right, for positive pressure values, the vector will need to be set in the negative X-direction which is achieved by a negative value of '-1' after the vector name. The wave is loaded through the height of the model again so there is no change in the J-coordinates: all that is required is to set the I-coordinates to start and end on the last key point along the I-axis.

NOTE: Be careful of Boundary Conditions – If applying pressure load to edges of model, set boundary conditions to FREE.

## 5.5.4 Generating a Spherical Wave

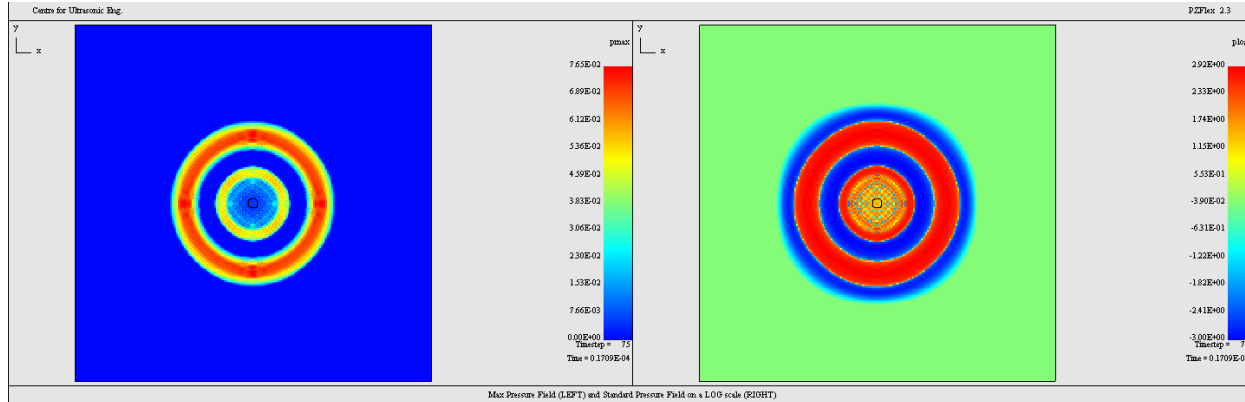


Figure 60: Generating a Spherical wave

A spherical wave propagates through space like the outer surface of a continually growing sphere. To implement the spherical wave, there are two PLOD subcommands that should be employed: SPOT and SDF2.

```
plod
  pdef pld1 func
  spot spot1 $x2 $y2
  sdf2 pld1 spot1 watr watr2
end
```

Code 45: Use of SPOT and SDF2

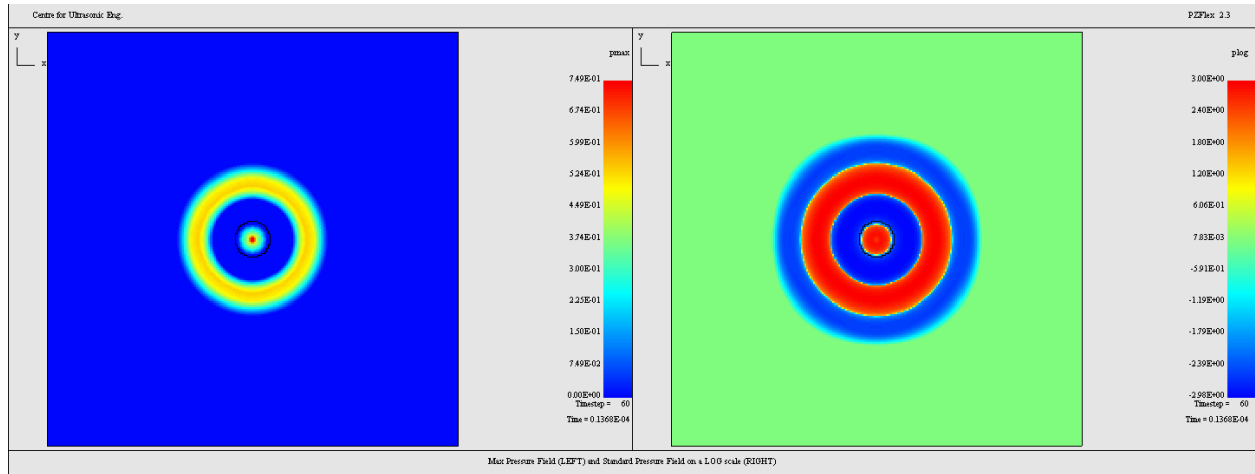
The PDEF command remains the same due to the use of the same driving function. The SPOT subcommand is a tool to identify a surface of nodes to apply a normal pressure loading by simply inputting a known nodal position close to the surface. In this example, the SPOT is situated at the centre point of the small circle (at key point x2 and y2). To help visualize how it works, if the location of the SPOT was a light source, the first surface that the light reaches is the side of the surface where the pressure load is applied at 90° (angle of incidence).

***sdf2 <plodname> <direcname> <matname1> <matnam2> <ibegin> <iend> <jbegin> <jend>***

The SDF2 subcommand differs from the SDEF command slightly with regards to the parameters: its functionality is still the same. The SDF2 command is able to identify a surface/boundary between two materials. The first 2 parameters are the usual applied function, 'pld1', and the direction vector which has been obtained by the SPOT command. The following two parameters are the two materials that form the boundary which in this case is 'watr' and 'watr2'. For more complex models where there are several boundaries, there are further IJK parameters to specify roughly where the boundary lies.

### 5.5.5 Using SPHR

The SPHR and CYLN sub commands can also be found in the SITE primary command. The reason they also appear in the PLOD command is that it allows the user to easily apply pressure wave to the defined sphere or cylinder created. This especially useful when the wedges of the spheres/cylinders are utilized to create more complex boundary shapes.



**Figure 61: Generating a pressure load to the declared spherical boundary using SPHR**

The SPHR command is similar to the SDEF command: it applies a specified function to a boundary in a direction defined by previous commands such as VCTR and SPOT. When the SPHR command is used in the SITE definition and a pressure is to be applied to its surface, it should also be used in the PLOD section of code as the parameters are almost identical making it much easier to set up.

***sphr*** ***<plodname>*** ***<direcname>*** ***<xcent>*** ***<ycent>*** ***<zcent>*** ***<radius>*** ***<thetabegin>*** ***<thetaend>***  
***<phibegin>*** ***<phiend>***

The first 2 parameters are the exact same as the SDEF command.

```
site
  regn watr
  sphr watr2 $x2 $y2 0 $radius 0
end

plod
  pdef pld1 func
  spot spot1 $x2 $y2
  sphr pld1 spot1 $x2 $y2 0 $radius 0
end
```

**Code 46: Use of SPHR in both the SITE and PLOD section**

The centre of the sphere is then entered as XYZ coordinates followed by its radius. The rest of the parameters are related to the angle which the sphere can begin and end with respect with the X and Z axis. This allows the pressure load to be easily applied to both 2D and 3D wedges of the sphere which otherwise would be difficult. Unfortunately, this is unavailable in version 2.3 of PZFlex.



## 5.5.6 Using CYLN

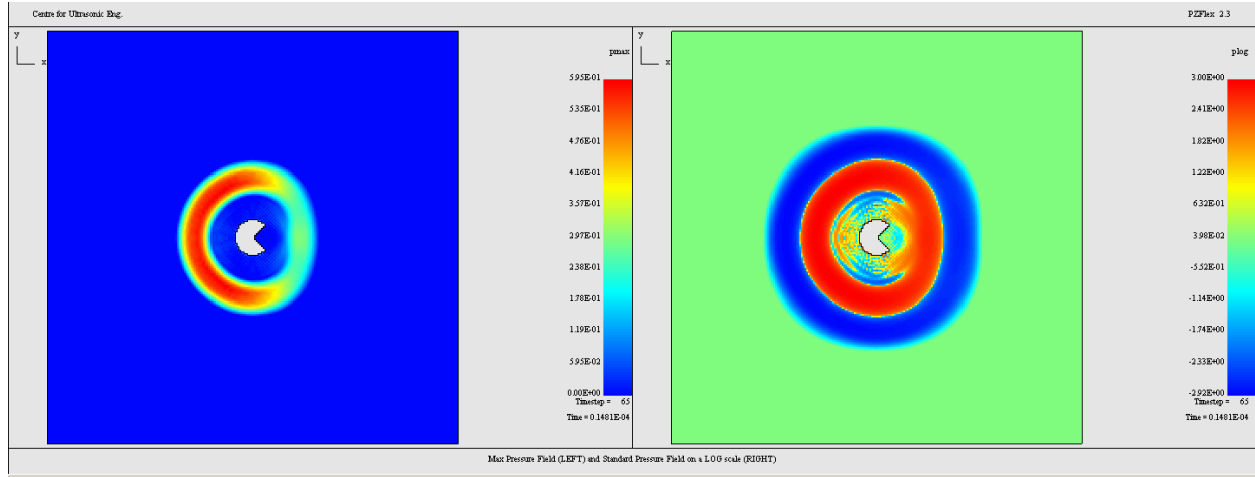


Figure 62: Applying pressure load using CYLN to 'pac-man' boundary

**CYLN** <plodname> <direcname> <axisname> <iaxis> <cbegin> <cend> <center1> <center2>  
<radius> <thetabeg> <thetaend>

The parameters of the CYLN sub-command are slightly different under the PLOD command than the SITE command: there are fewer commands that need to be input as shown in Code 48: Changing 'iaxis', 'cbegin' and 'cend'. The CYLN command requires a pressure load name and a direction vector to apply the load just like the SPHR command. The 'axisname' is used if there is a user defined axis which can be used to align the cylinder. The 'iaxis' parameter defines the axis which the length of the cylinder is positioned against. In this example, the Z-axis (into the page) is the orientation of the length of the cylinder. The 'cbegin' and 'cend' constraints determine the length of the cylinder along 'iaxis'. The 'center' parameters dictate the central position of the circular face of the cylinder. The radius of the circular face is then controlled by the 'radius' parameter. Finally, the last two constraints define the angle at which the circular face begins and ends, allowing wedges to be created.

```
site
  regn watr
  cyln void stnd z 0 1 $x2 $y2 $radius $radius 0 0 45 315
end

plod
  pdef pld1 func
  spot spot1 $x2 $y2
  cyln pld1 spot1 stnd z 0 1 $x2 $y2 $radius 45 315
end
```

Code 47: Use of CYLN in both the SITE and PLOD section

The CYLN subcommand only requires one radius value under the PLOD command and there are no parameters for inserting a hole at the center of the cylinder.

To gain a better understanding of the 'iaxis' parameter, another example will be explored where the length of cylinder will be along the Y-axis.

```
site
  regn watr
  cyln void stnd y 25e-3 75e-3 0 $x2 $radius $radius 0 0 45 315
end

plod
  pdef pld1 func
  spot spot1 $x2 $y2
  cyln pld1 spot1 stnd y 25e-3 75e-3 0 $x2 $radius 45 315
end
```

Code 48: Changing 'iaxis', 'cbegin' and 'cend'

The cylinder will begin at 25 mm and extend to 75 mm along the Y-axis. When the Y-axis has been selected for 'iaxis', the centre co-ordinates will change to a Z-coordinate and X-coordinate respectively.

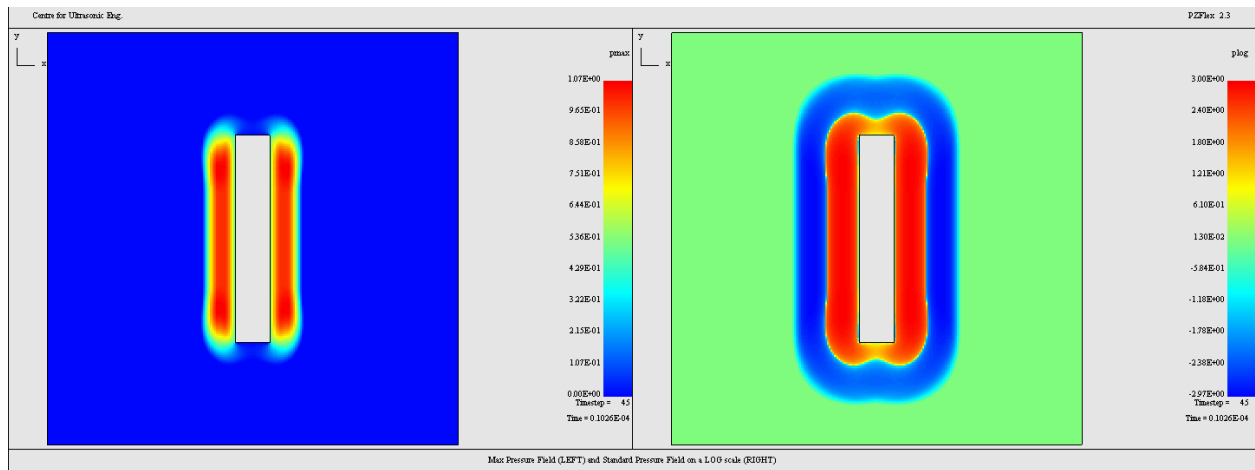


Figure 63: The cylinder oriented along the Y-axis

The angle constraints do not appear to have an effect due to analysis on one plane.

## References

- [1] Fundamental of Acoustics, 3<sup>rd</sup> Edition, L.E.Kinsler, p1
- [2] Passive and Active Acoustic Characterisation of Bubbles Relating to Bioprocess Monitoring, Strathclyde 2008, K.G.Macpherson, p41
- [3] [http://www.physikinstrumente.com/en/primages/pi\\_dipoles\\_d4c\\_O\\_eps.jpg](http://www.physikinstrumente.com/en/primages/pi_dipoles_d4c_O_eps.jpg), visited 26/08/10
- [4] [http://en.wikipedia.org/wiki/Plane\\_wave](http://en.wikipedia.org/wiki/Plane_wave), visited 20/07/10
- [5] <http://en.wikipedia.org/wiki/Tonpilz>, visited 16/08/10