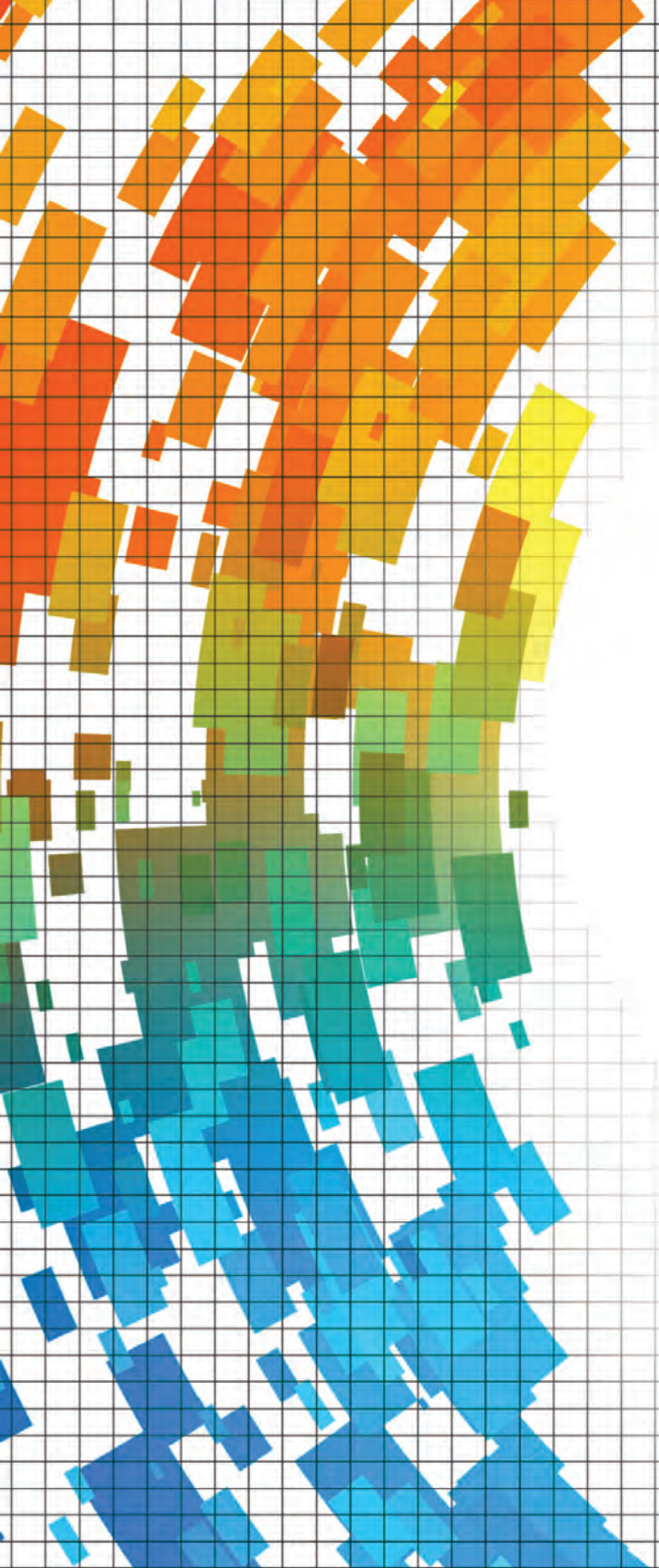


PZFlexx®
Virtual Prototyping

User Manual

User Manual

Version 2014

PZFlex

Weidlinger Associates, Inc.
399 West El Camino Real, Ste. 200
Mountain View, CA 94040—2607
http://www.pzflex.com

CONTENTS

INSTALLATION

PZFlex is available on CD-ROM and via Internet download from the PZFlex web site, www.pzflex.com. For a CD-ROM, contact Weidlinger to arrange for shipping. To download the software, you need a valid username and password, which are provided when you purchase a license to use PZFlex.

PZFlex is available for Windows Operating Systems and SuSe/RedHat distributions of Linux. We recommend installing the 64-bit rather than 32-bit version if possible.

Downloading PZFlex

1. Go to: www.pzflex.com.
2. Click on “Support” >> “Client Login”
3. Enter username/password.

Click the software version appropriate for your computer system. The download process begins automatically.

The latest documentation is included with the PZFlex package. It can also be downloaded separately from the Download page.

Installing PZFlex on Windows Systems

PZFlex is distributed as a self-extracting executable. For our purposes, we assume the installation file name to be `pzflex-2014.x86_64.setup.exe`. If there are access constraints on the system, you may need to log in as “administrator” to install the software. If you do not have access privileges as administrator, have the system administrator install the software.

Double-clicking the installation program icon runs the installation automatically. We recommend that you select the default configuration. You will be asked to confirm the path name to the installation folder. The default (recommended) installation directory is `c:\Program Files\WAI`. Although we use this name throughout the manual, you may substitute a different folder name. The installation program completes installation of the software in the designated installation folder.

When the installation is complete, the system will prompt you to reboot. Respond “Yes,” to reboot the system.

Warning
Uninstalling previous PZFlex software removes all file names that were placed on the system during the previous installation process. Any changes that were made to these files are lost when the program is uninstalled. The software installation includes a set of program file examples; we recommended that you avoid modifying the original versions of these files. Instead, create a copy of the examples folder in a separate workspace, where you can modify files to meet your specific needs.

Installing a Dongle Driver—Windows

A dongle is a software enabler key; that is, it permits only authorized use of particular software. It is a piece of hardware that is plugged into a USB port of the computer. To use a dongle you must first install a dongle driver. After installing PZFlex and rebooting, install the dongle driver via the following path from the Start menu (bottom left of screen):

Start > All Programs > Weidlinger Associates, Inc > PZFlex > License Manager > Install HASP HL Driver

Place the Flex.key file that comes with the dongle in the `c:\Program Files\WAI` directory. PZFlex will now run on the system when the dongle is connected to an active USB port. This procedure can be performed on any number of systems. PZFlex functions, however, only on the system for which the dongle is plugged in. Dongle users may skip the following “Obtaining a Flex.key File” section.

Obtaining a Flex.key File—Windows

If you are not using a dongle, you need a license key file, Flex.key, to run PZFlex. PZFlex is licensed to run on one or more designated computers. When the program is run, it checks information about the computer system it is being run on against the list of validated systems identified in the Flex.key file. If the Flex.key file is not found, or if the computer system is not validated, the program terminates with an error.

To obtain a Flex.key file, go to:

Start > All Programs > Weidlinger Associates, Inc > PZFlex > Licence Manager > Generate Licence Info

Email the newly created file, `c:\Program Files\WAI\info.txt`, to flex_support@wai.com, requesting a Flex.key file. You will receive the Flex.key file as an attachment in a return email. Place it in the `c:\Program Files\WAI\Flex.key` directory.

The following utility may be useful in ensuring that the Flex.key file is placed correctly:

> All Programs > Weidlinger Associates, Inc > PZFlex > Licence Manager > Install License File

Running the Software—Windows

Installation is now complete. PZFlex, Review, and Build can all be executed through the FlexLAB integrated development suite. FlexLAB is accessible either through the Programs section of the Start menu or through a desktop shortcut. Further information on running and using FlexLAB is embedded in the “Help” menu, accessible when FlexLAB is running, or by browsing the documentation folder. Starter guides for PZFlex and Review are available in the FlexLAB “Help” menu and in the documentation folder.

Installing PZFlex on Linux Systems

PZFlex is currently available for SuSe 11 and Red Hat Enterprise Linux 4 & 5 systems. The program is provided as a single installation file obtained on CD-ROM or as an Internet download. The software installation package is named `install_pzflex_rhel5.run` (or similar, depending on OS). The file is provided as a self contained installation package. Once the installation file is available on the Linux network, you can log onto the Linux system as root (superuser). Enter:


```
install_pzflex_rhel5.run
```

The installation will then proceed to uninstall the old PZFlex installation, and the secondly install the new version.

Installing a Dongle Driver—Linux

The new Linux installation process has the dongle driver built in and should install automatically. A dongle is a software enabler key; that is, it permits only authorized use of particular software. It is a piece of hardware that is plugged into a USB port of the computer. To use a dongle, you must first install a dongle driver.

Should you have issues after installation of the main software package, please contact PZFlex to obtain the dongle driver RPM package (named `akausb-rhel3-1.rpm` or similar) from the PZFlex CD or website and enter:

```
rpm —ivh akausb-rhel3-1.rpm
```

Place the Flex.key file that comes with the dongle in the `c:\Program Files/WAI` directory. PZFlex will now run on the system when the dongle is connected to an active USB port. This procedure can be performed on any number of systems. PZFlex functions, however, only on the system for which the dongle is plugged in. Dongle users may skip the following "Obtaining a Flex.key File" section.

Obtaining a Flex.key File—Linux

If you are not using a dongle, you need a license key file, Flex.key, to run PZFlex. PZFlex is licensed to run on one or more designated computers. When the program is run, it checks information about the computer system it is being run on against the list of validated systems identified in the Flex.key file. If the Flex.key file is not found, or if the computer system is not validated, the program terminates with an error.

If you do not have a valid Flex.key file, you can obtain one by filling out a Hardware Installation Sheet (available from Weidlinger) and faxing it to the number indicated on the sheet. A Flex.key file will be generated and emailed to you. Place the Flex.key file in the directory `/opt/WAI` to enable the software to locate it properly and to permit PZFlex to run on the computer.

Running PZFlex—Linux

PZFlex, Review, Build, and FlexLAB can be run from any terminal window with the commands "pzflex," "review," "build," and "flexlab." We recommend that you use FlexLAB to control all Flex jobs.

> Note: Firewalls
> The Flex.key is associated with the network card or modem on the computer. Interrogation of the card validates the key. On some systems, the Flex.key triggers an alarm from the software-based firewall. If this occurs, modify the firewall settings to allow PZFlex access to the computer's network card or modem. Be assured that PZFlex never uses the interrogation to send or receive information over the Internet.

Establishing User Preferences Using Defaults Files—Windows and Linux

When PZFlex or Review is run, it reads default files to establish user preferences. The `Flex.defaults` file is read by PZFlex, and the `Review.defaults` file is read by Review. You can alter these files through the interface in FlexLAB (accessed from the desktop shortcut or the `Start/Program Files/WAI` menu) by selecting from the `Options/Flex/Flex` defaults menu. The code is installed with "global" files provided in the `c:\Program Files\WAI\Flex\PZFlex\Programs` folder. You can copy these files to your directories and alter them to override the default settings. You can also edit the files to establish general default preferences for any user running the programs on the computer system.

Executable Names

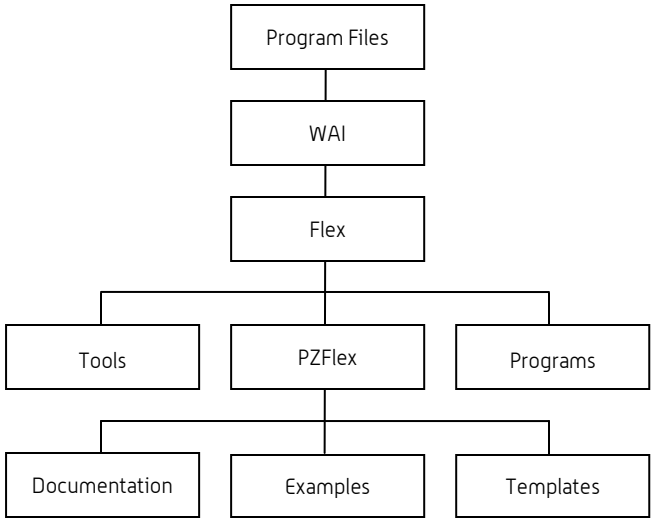
With version 3.0, PZFlex has adopted a new naming convention for executables. If the number “64” appears in the name, the executable is designed for 64-bit operating systems; “32” indicates that the executable is written for 32-bit systems. If the letter “s” appears in the name, the executable uses “single precision” arithmetic; the letter “d” indicates “double precision” arithmetic. For example, `pzflex64s.exe` is the PZFlex executable for 64-bit operating systems with single precision arithmetic. Note that 64-bit executables cannot be run on 32-bit operating systems.

To simplify options, batch files with the names “pzflex,” “review,” and “build” call the new executables automatically. The default options are the “64” option for users with 64-bit operating systems and the “32” option for users with 32-bit operating systems. The default precision is “s” for all installations.

You can point FlexLAB to different executables using the `Options/Flex/Executables` menu. DOS Prompt/Terminal users can enter the name of the alternative executable directly.

The message “Flex.key file not found” means that the Flex.key file was not placed in the proper folder. Check to make sure that the Flex.key file is in the `c:\Program Files\WAI` directory.

The message “Code not validated for this system” means that the Flex.key was found but that the validation information it contains does not match the system it is being run on. Contact flex_support@wai.com for assistance.



Software Directory Structure (default for Windows OS)

The directory structure under Linux is the same except that “Program Files” is called `/opt`.

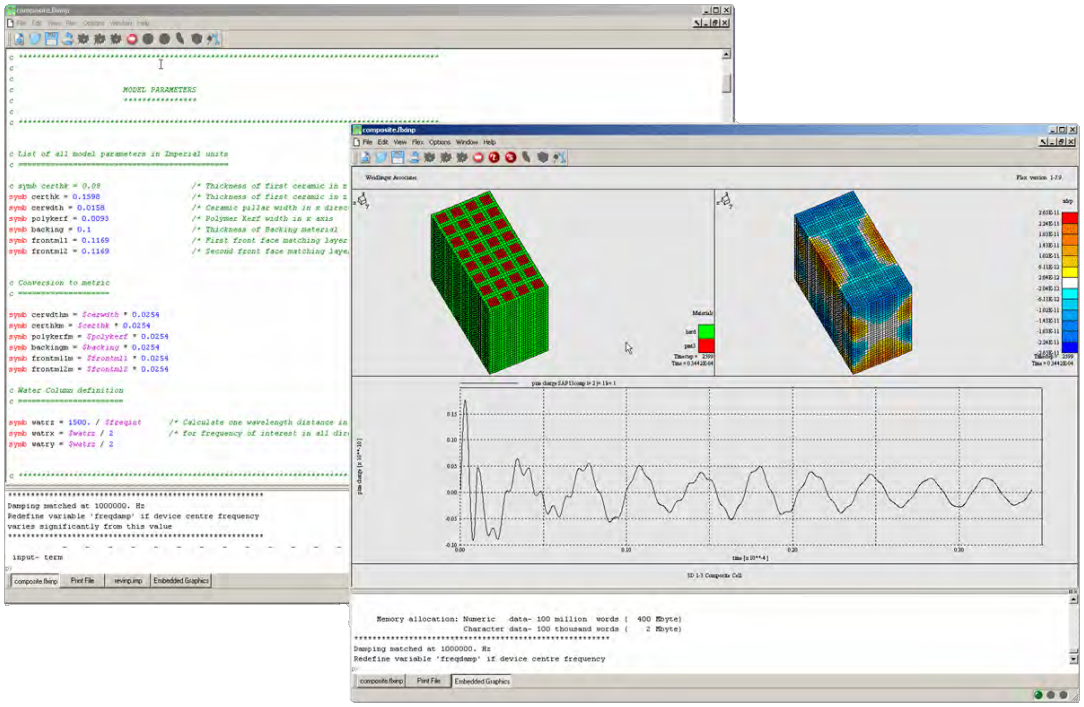
All executables for PZFlex, Review, Build, and FlexLAB are stored under “Programs.” The Flex.key file is stored under “WAI.”

“Documentation” and “Examples” contain further subdirectories. Documents stored in these subdirectories are either text files for use in FlexLAB or PDF documents that can be read with Adobe Acrobat Reader. Acrobat Reader is not installed with PZFlex; it can be downloaded free of charge from www.adobe.com.

FlexLAB

FlexLAB is an integrated editing program that allows you to create PZFlex input files, run your PZFlex job, monitor the print file history, and view the results graphically, all in the same environment.

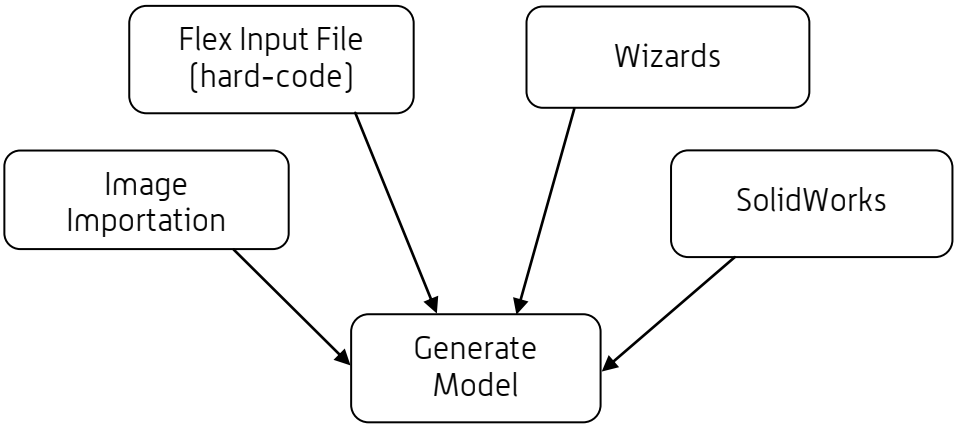
Below are two examples:



FlexLAB is a common interactive environment for use with any of the FLEX programs: PZFlex, EMFlex, NLFlex, and so on. As such, many of the commands in FlexLAB refer to “Flex” as the executable.

Modelling Options

FlexLAB offers a variety of ways of generating models:



Flex Input File

Flex input files are the standard files for hard-coding any model. You are required to learn how to code in Fortran and understand how to use the commands specific to PZFlex. There is a small learning curve but the advantage of learning to hard-code a model will reveal the powerful capabilities of FlexLAB. *(See Using PZFlex—P118)*

Wizards

Wizards uses an intuitive interface to quickly generate common models and results. *(See Wizards—P40)*

Image Importation

The image importation tool allows an image file to be loaded into PZFlex and converted to a format that can be processed by FlexLAB. Further coding is required but for complex 2D models, this simplifies the process of creating the model greatly. *(See Material Image Import Tool—P34)*

Solidworks

The Solidworks interface provides a means of porting models/structures designed in Solidworks over to FlexLAB. *(See Solidworks Interface Guide P108)*

Starting FlexLAB

Once you have installed PZFlex, you can access FlexLAB from a desktop icon, a taskbar icon, or a folder option in the “Programs” section of the PZFlex “Start” menu.

PZFlex Executables

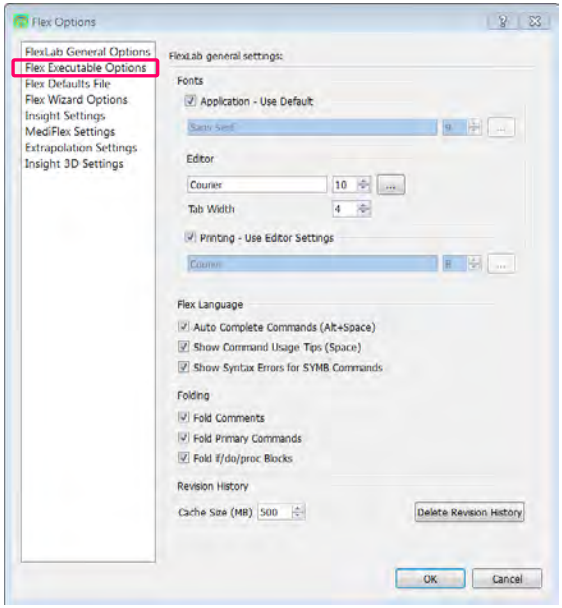
The PZFlex installation package automatically forms a new directory structure for the PZFlex executables and stores them in:

`C:/Program Files/WAI/Flex/Programs` (Windows)

or

`/opt/WAI/Flex/Programs` (Linux)

To use a version of PZFlex that exists in a different location, you can alter the path names to the program executables using PZFlex “Options.” Access PZFlex Options through either the “Options” toolbar button or the “Options” menu; select “Executables” on the left side of the screen:



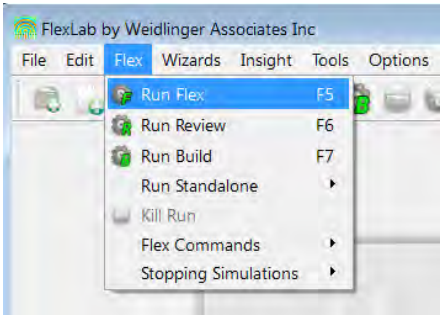
Note
Changing the path names of the executables is not recommended unless you are fully conversant with the software.

Running FlexLAB

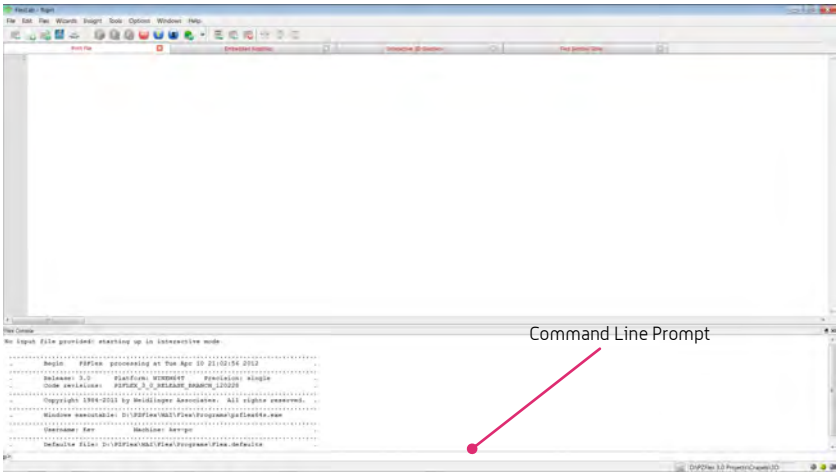
After you select and start FlexLAB, you will see a blank project screen:

To open PZFlex, Review, or Build, click one of the program options on the toolbar:

You can also access the same options via the PZFlex menu:

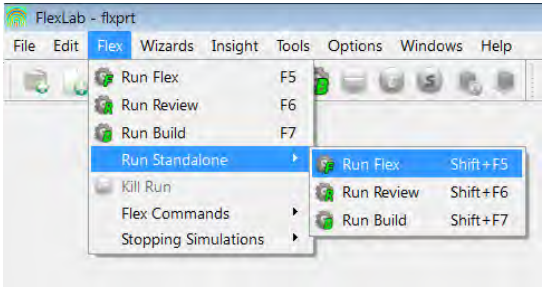


This opens a command line prompt window at the bottom of the screen:



You can begin to enter PZFlex commands manually into this prompt window the same as you would from a standard DOS/Linux prompt window.

You also can run any of the programs in a standalone manner, i.e., you can start PZFlex or Review without sending it an input file and then run it manually. This feature is useful when you are viewing older history files for reference but do not want to overwrite an existing project. To use this feature, use the standalone menu on the PZFlex submenu:

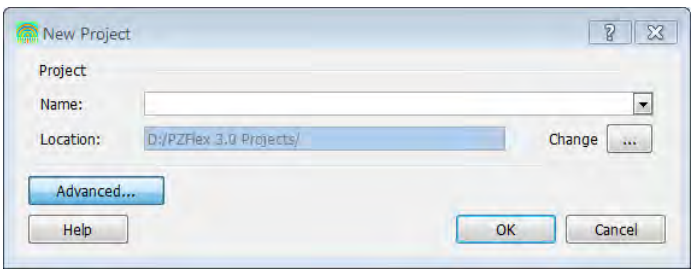


Starting a New Project

To begin a model in PZFlex, you generally need a PZFlex input file, a materials file, and possibly a Review file (for post-processing). Most users keep all of these files in a single

directory. The “New Project” feature allows you to name and open all the required files in a single directory.

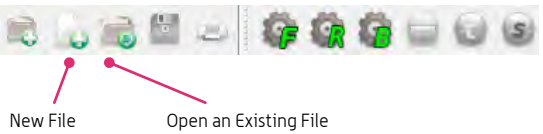
Selecting “New Project” initializes the following window:



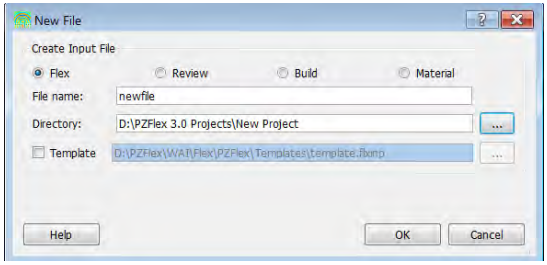
Specify the name of the files and the directory in which they are to be stored. Once you select “OK,” FlexLAB creates a PZFlex input file, a materials file, and a Review input file from the templates provided. You can use the “Advanced” button to restrict or expand this option.

Starting a New Input File

It is simplest and most efficient to build up your input file in a separate text-based PZFlex input file and then execute it in FlexLAB. To open a new text file, either click the “New File” button or select “New File” from the “File” menu bar:



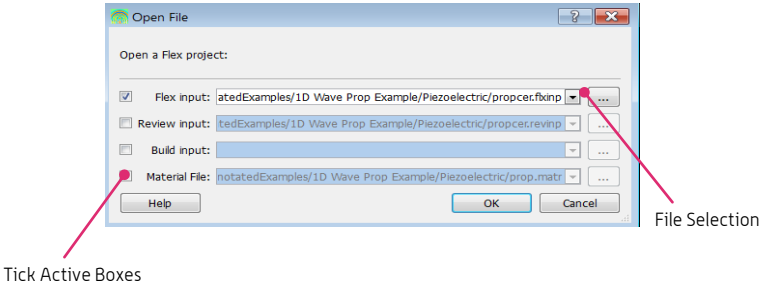
FlexLAB will ask you the file name and what application the file is being created for, i.e., Build, PZFlex, or Review:



When you enter the base name of the file, FlexLAB automatically adds the appropriate suffix, identifying it as an input file for the selected program. Selecting the template option allows you to choose template options for either PZFlex or Review input files. You can either define your own templates or use the ones provided.

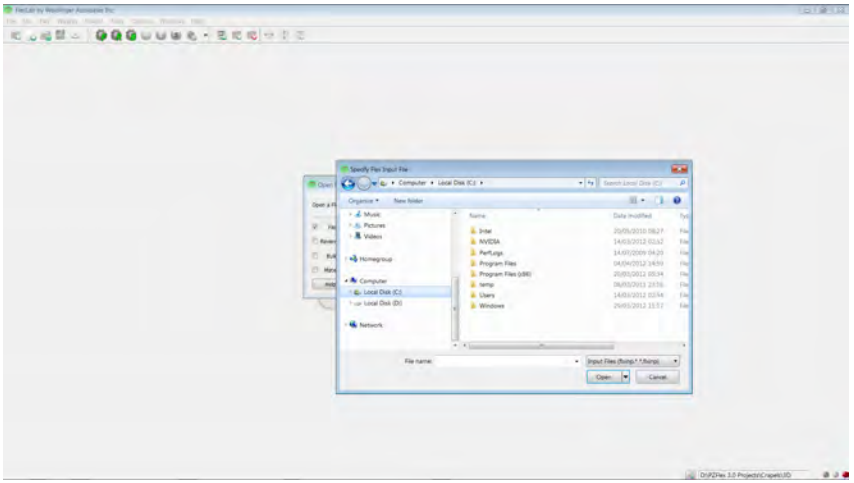
Opening an Existing Input File

To open an existing file, use either the toolbar icon or “Open File” from the “File” menu.



You will see this window:

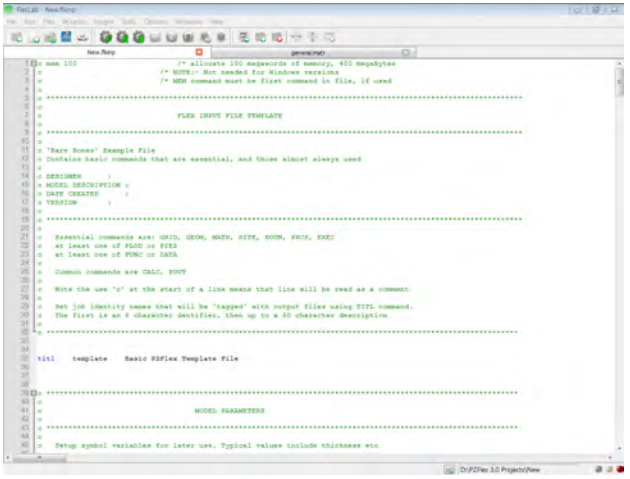
To open the file, click the box on the left side of the window. You can then either enter the directory path for the input file or use the “File Selection” button to open the file-searching option:



FlexLAB allows you simultaneously to open one PZFlex input file, one materials file, one Build input file, and one Review input file.

Viewing Input Files

Occasionally, you may wish to open other input files to copy and paste previously written code. Use the “View” option in the “File” submenu to open write-protected versions of previous files; you can then copy and paste from these without accidentally altering them. To distinguish between active files and viewing files, FlexLAB displays viewing files on a grey background:



Saving Input Files

Files are saved the same way as in other Windows-based programs—by using either the toolbar icon or the “Save” option in the “File” menu.

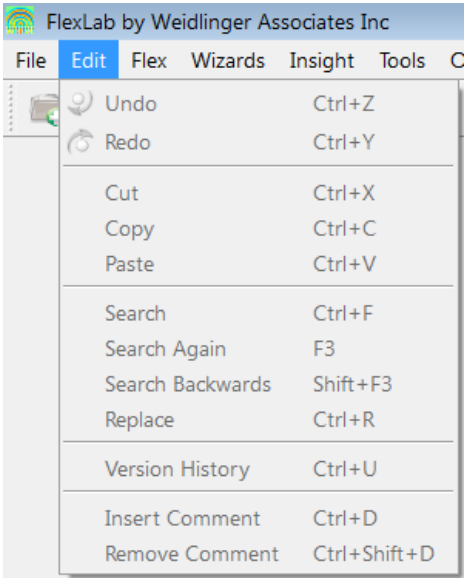
Printing Input Files

The “Print” option is currently available for printing only text files. You can access it through either the toolbar icon or the “File” menu.

To print the graphical images, you must output the images to TIFF or PostScript format using standard PZFlex or Review commands and then use another Windows program to print them.

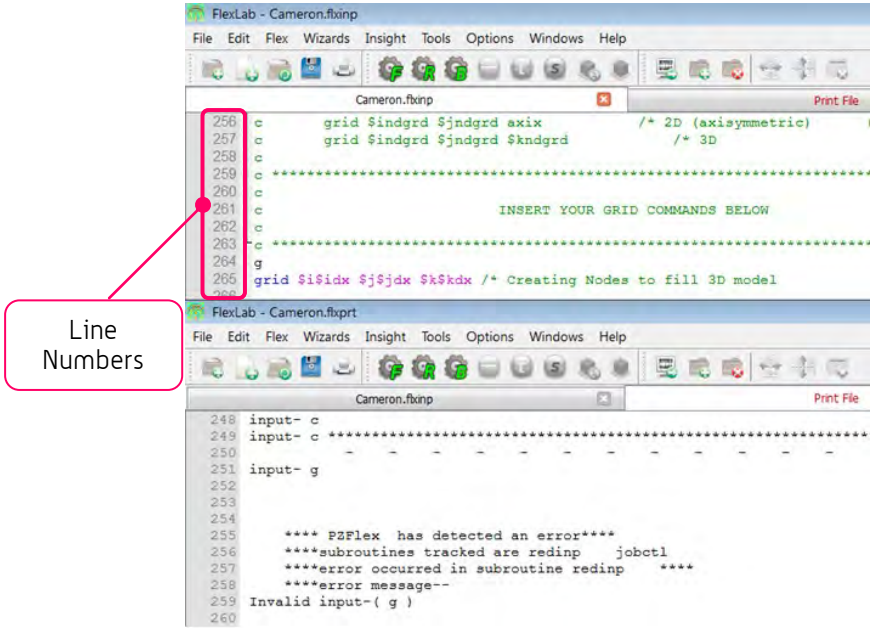
Editing Input Files

FlexLAB uses the same text-editing commands as nearly all Windows-based editing applications, e.g., copy, cut, paste, find, and replace. They are accessed either from the “Edit” menu or by using key commands.



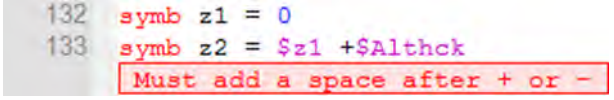
Line Numbers

In the Flex input file, each line of code is designated a number making it easier to identify errors in code when the simulation stops abnormally. The Print File shows exactly where the simulation runs to before it terminates. Line numbers also appear in the Print File, allowing you to easily locate the issues with the code.



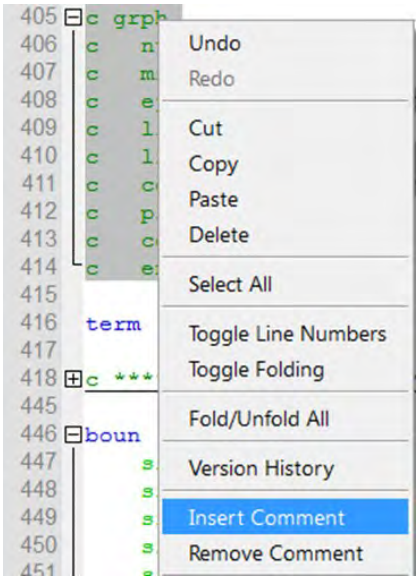
Error Checks

When there are syntax errors in the code, PZFlex will highlight it immediately rather than waiting for the user to discover this when trying to run the flex file.



Comment Toggle and Collapsible Code

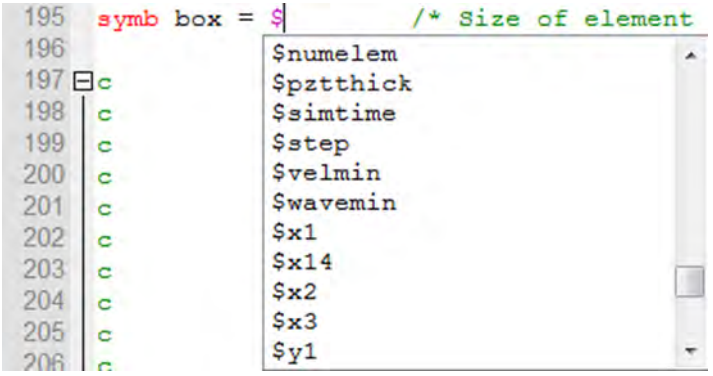
Comment toggle allows several lines of code to be instantly transformed into comments using the 'Insert Comment' tab in the right mouse button (RMB) menu. This is useful for debugging your code as you can quickly remove without deleting the code from the flex file to give you a better understanding of how the code is affecting your model. The commented lines of code can be reinstated using the 'Remove Comment' tab in the RMB menu. You can also use the keyboard shortcut once you have highlighted an area of text : **CTRL + D** to comment and **CTRL + SHIFT + D** to uncomment.



Code can also be collapsed by clicking on the negative symbol next to the line number. For models with vast amount of code, it can be useful to collapse lines of code that you are confident that works and do not need to be changed. This means you cannot accidently change or delete the code. The code can be easily expanded back to its original form by clicking on the positive symbol.

Auto-complete SYMB

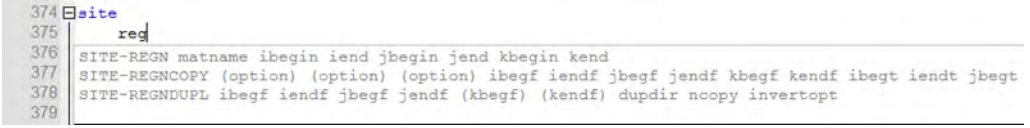
When the user calls on an existing symbol using the '$' operator, pressing CTRL + Space will list all the symbols that have been declared in the flex file and allow you to select the symbol to be used*. Therefore, you do not need remember all the symbols used in the code and it saves you time scrolling back and forth to find the symbol you want to use.



*Only Applies when using the SYMB command

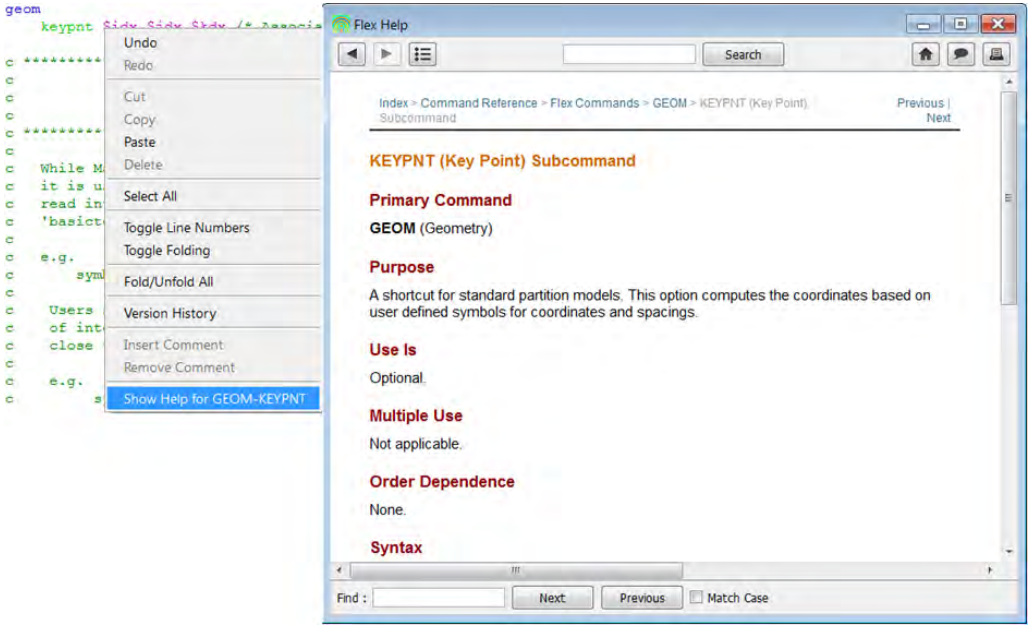
Tooltips

Tooltips helps the user to understand what inputs a particular sub-command requires to function properly as well as the sub-commands that are available. For example, if you are using the REGN sub-command within the SITE primary command, when you begin typing 'REGN', a textbox listing the necessary inputs for REGN appears. This allows the user to simply enter what is needed without having to remember the number of different inputs and the exact location/sequence of these parameters.

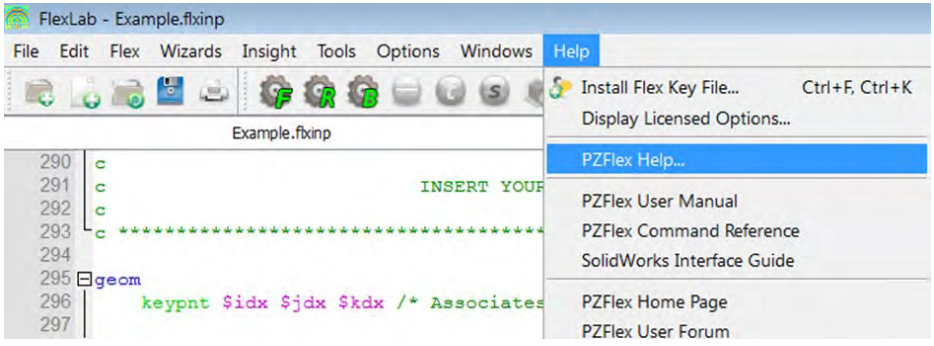


HTML Help

Any command that you are unfamiliar with or require more information about, right clicking the command and selecting “Show Help for (command here)” will open the HTML help file displaying all the information related to the command.



The PZFlex User Guide and Command Reference have also been converted to HTML which allows you to search for anything within these documents. This can be accessed through the Help tab above the PZFlex icon bar.



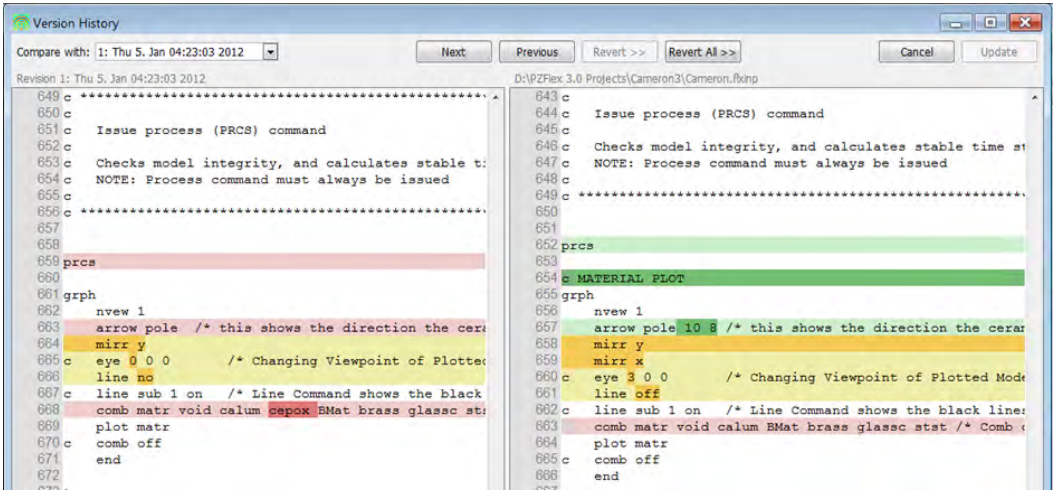
Column Selection

Using **ALT + Left Mouse Drag (Windows) / CTRL + Left Mouse Drag (Linux)**, you can highlight columns of code/data instead selecting whole lines.

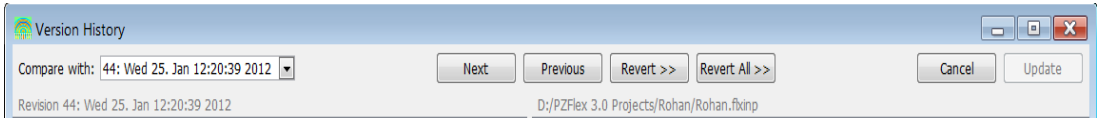
```
symb i1 = 1
symb i2 = $i1 + nint ( ( $x2 - $x1 ) / $box )
symb i3 = $i2 + nint ( ( $x3 - $x2 ) / $box )
symb indgrd = $i3

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

Version History

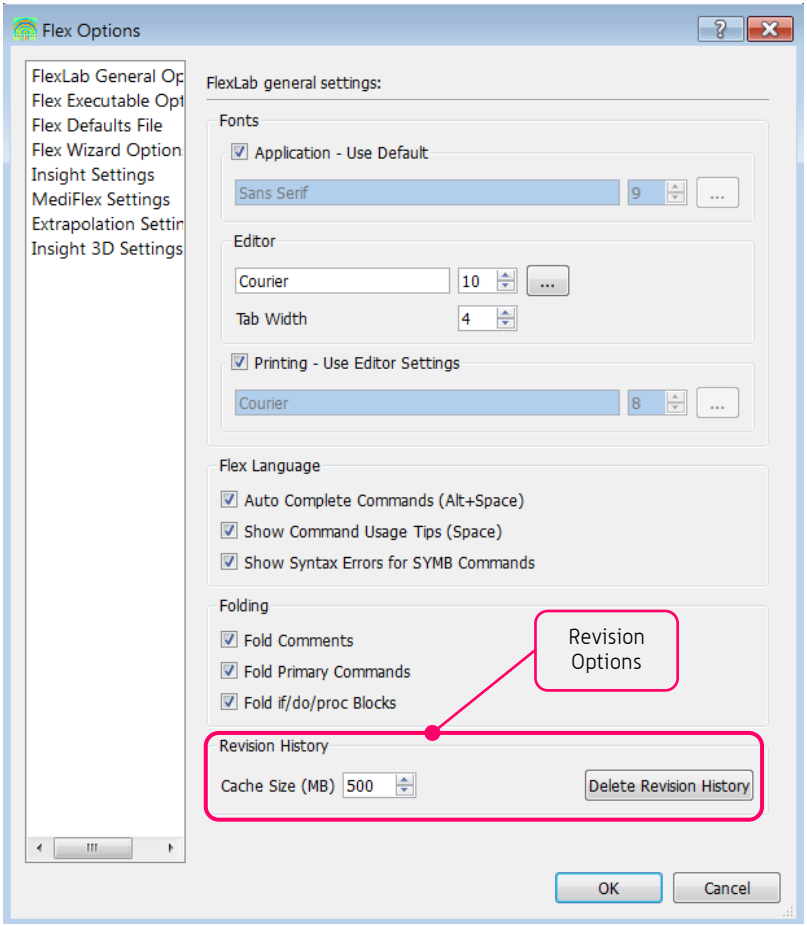
The version history feature allows you to revert back to an older version of the code. This is useful for debugging errors and testing the effects of new pieces of code or alterations. The version history feature opens a new window where the latest version of your code can be compared to an earlier version. Differences in the code files are highlighted by a variety of colours, making them easy to identify: green indicates newly added code; yellow indicates changes in existing code and red indicates removed code.



Each time your code runs, its save the version of code in execution. Using the drop down menu in the top left-hand corner, each version of code can be selected; using the next and previous buttons allows you to navigate through the changes that have been made and revert them back. Once all changes have been restored, the update button finalises all the modifications.

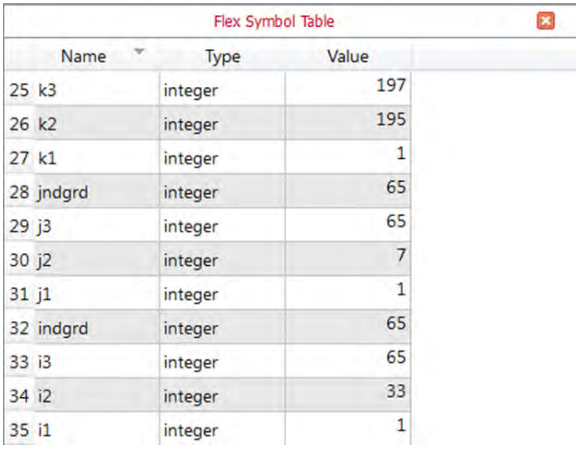


The number of previous versions of code that can be stored is dictated by the cache size which can be defined by the user in Options > Settings.



Flex Symbol Table

During the simulation run, it is possible to observe the values that are taken on by the SYMB variables used in the model. This helps the debugging process; if the model is producing unexpected results the symbol values can be checked to confirm they have been set correctly. Using the 'term' command, the simulation can be paused to allow you to view and double check the variable values.



	Name	Type	Value
25	k3	integer	197
26	k2	integer	195
27	k1	integer	1
28	jndgrd	integer	65
29	j3	integer	65
30	j2	integer	7
31	j1	integer	1
32	indgrd	integer	65
33	i3	integer	65
34	i2	integer	33
35	i1	integer	1

Syntax Highlights

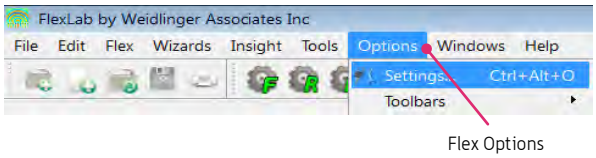
Syntax highlighting speeds up the modeling and debugging of input files by color-coding commonly used commands or commented sections. FlexLAB uses the following colors:

Dark green	Comments in the code, typically preceded by the "c" or "/*" comment symbol defined in PZFlex.
Dark Blue	Primary commands in PZFlex.
Light Green	Secondary commands in PZFlex.
Black	Standard font for variable names and command line options.
Light Blue	Number strings.
Red	"Symb" command.
Magenta	Symbol variable names.

Setting up PZFlex/Review Defaults

FlexLAB has a dialogue window that allows you to define your preferred working environment. You can specify such things as default axis orientation, color maps, foreground/background color options for the graphics window, and model viewpoint. To access the defaults section, open the “Flex Options” window, by clicking the “Options” menu.

The following window will appear:



Highlight “Flex Defaults,” on the left. You can now modify any of the default settings and click “OK” to save the settings for further use.

You can also set preferred defaults for certain graphics and symbol-related options in the `Flex.defaults` and `Review.defaults` files. The file names, which are case-sensitive, must be created exactly as shown. The defaults file is useful if you frequently need to change baseline code defaults for your models.

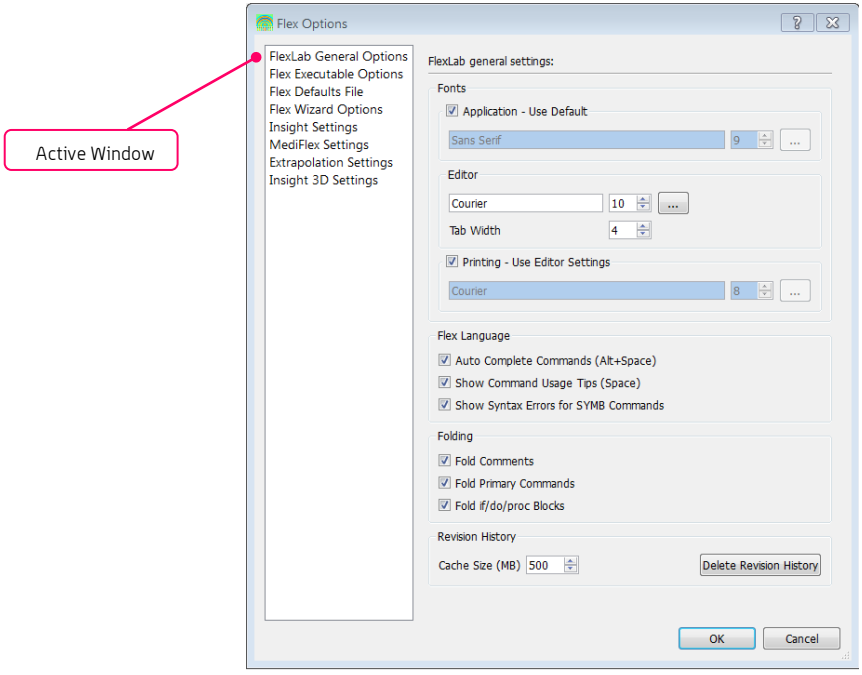
Accessing Default Files

When the program is initiated, it looks in the current directory for the appropriate defaults file. If it finds the file, it reads it and sets the requested defaults. If the program does not find the file, it looks in your home directory for the defaults file preceded by a period, e.g., `Flex.defaults`. If it finds a defaults file there, it reads it and sets the requested defaults. If it still does not find the defaults file, it applies the baseline code defaults. If the `standard_procedure_file` option is included in the defaults file, all procedures on the file are read and made available for use on your current job. All versions of the program (i.e. PZFlex, Review, and so on) contain a sample defaults file. Any of the following lines can be used to set the options:

```
background_color          gray          /* any of: black, white, gray or rgb values
foreground_color          black         /* any of: black, white or rgb values
material_color_table 4                  /* color table number: from 1 to 6
data_color_table     2                  /* color table number: from 1 to 6
view_point       0.0  0.0  0.0          /* x, y and z values of view point
vert_point      -1.0  0.0  0.0          /* x, y and z values of vertical point
eye_point       -1.0 -2.0 -3.0          /* x, y and z values of eye point
number_views_and_type     4    1        /* two integers. first is from 1 to 8
swap_2d_xy_axes           on            /* any of: off, on
reverse_2d_x_axis         on            /* any of: off, on
reverse_2d_y_axis         off           /* any of: off, on
postscript_color          on            /* any of: off, on
postscript_date           on            /* any of: off, on
postscript_format         portrait      /* any of: portrait, landscape
element_grid_lines        on            /* any of: off, on
line_width                .75           /* line thickness in points (72 points/in)
symbol_echo               value         /* any of: off, on, value
symbol_print              off           /* any of: off, on, none
job_processing            batch         /* any of: batch, interactive
text       varname        textstring    /* set text variable varname = textstring
memory_character          30            /* default character allocation (10**3 words)
memory_numeric            13            /* default numeric allocation (10**6 words)
standard_procedure_file   pathname      /* pathname of procedure file
```

Setting General Options

As with the PZFlex defaults, you can modify simple options for FlexLAB. Selecting the "General" setting on the left side of the screen brings up this window:



This shows the default settings of the general window for the file-saving and for the text editor font. Certain features such as auto-complete, tooltips, error checking and collapsible comments can also be enabled/disabled in this tab. The cache size for the Version History can also be adjusted in this option tab.

Flex Executable Options

The settings that determine the operation before, during and after a simulation run can be adjusted here. The settings mainly involve how to saves your files and the monitoring capabilities of the program. 3D graphics is enabled in this section which will be discussed in more detail later.

Flex Wizard Options

The wizard options gives enable you to change the number CPUs used when running a wizard project. Other settings governs how certain data is displayed and general appearance of the wizard interface. (See later in *Wizards—Using Wizards* section on P53)

Extrapolation Settings

The appearance of models in the extrapolation toolkit can be altered along with the default outputs that the toolkit can provide. (See later in *Extrapolation Toolkit—Using Extrapolation* section on P69)

Insight Settings

Insight is a post processing tool that allows the saved data from the model simulations to be accessed and viewed graphically. The general options for Insight can control the type of units displayed, number formats, font types for axes and impedance plots. (See later in *Insight—Using Flex History* section on P78)

Insight 3D Settings

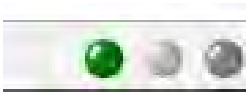
The Insight tool has 3D functionality and the settings here cover 3D graphics and tools that are available such as the spliner widget and mode shape plots. The usual general appearance changes regarding background colours, fonts and legends can also be set here. . (See later in *Insight—Using Flex Dataout* section on P91)

MediFlex Settings

The HIFU toolkit settings govern the CPU settings, model visualisation, general appearance and functionality of the GUI. (See later in *MediFlex Toolkit—Using MediFlex Toolkit* section on P107)

Running PZFlex Projects in FlexLAB

Once you have completed your PZFlex input file, you can run the model simulation by clicking the PZFlex toolbar button (shown above) or by selecting “Run Flex” from the PZFlex menu. FlexLAB then runs the PZFlex program using the text file shown in the PZFlex input window. The traffic-light colors at the bottom right of the screen indicate the program status:

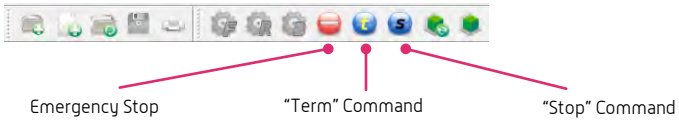


Green means that the simulation is running.

Yellow means that one of the tools is running but is in a temporary halt phase, i.e., it has reached a “term” or “t” command in the text file.

Red means that none of the simulation tools (Build, PZFlex, or Review) is running.

When a simulation tool is running, the run commands for Build, PZFlex, and Review are disabled to prevent you from accidentally starting multiple runs in a single command window. In this case, the toolbar looks like this (note the three new red buttons, explained below):



The “emergency stop” button, which is the equivalent of typing “control c” in the DOS/Linux prompt window, terminates the ongoing simulation *without* saving any of the data in the time history files.

The “term” button sends the “term” command. It is used to continue simulations that have been paused by a previous “term” command embedded in the text file.

The “stop” button sends a “stop” command to a simulation that has been paused by a previous “term” command embedded in the text file. This command stops the program and saves any files.

Note

This option does not generate a “kill file” to terminate the job during a simulation run.

Stopping Simulations

Stop Command

The Stop input command is used to terminate the processing of a job in an orderly manner. If a STOP command is encountered in the middle of the input stream, no input commands following the STOP command are processed. If no STOP command is found and the end-of-file for the Job Input File is encountered, a STOP command is automatically generated, resulting in a normal termination of job processing. A restart file is also always written to the Restart Output File upon encountering a STOP command and any time histories requested by the POUT HIST command are also displayed (unless suppressed) and written to the Time History Output File.

The #KILL Option

The #KILL option causes the program to check for the presence of the kill file (Command Override File) for the job at the time it is encountered. If the kill file is found, the current Job Input file is closed, the kill file is opened and the program processes the commands encountered. This option allows the user to choose the point in a Symbol script which provides a proper point for job termination should the need arise to bring the job down before it finishes. The form of the option is:


```
symb #kill
```

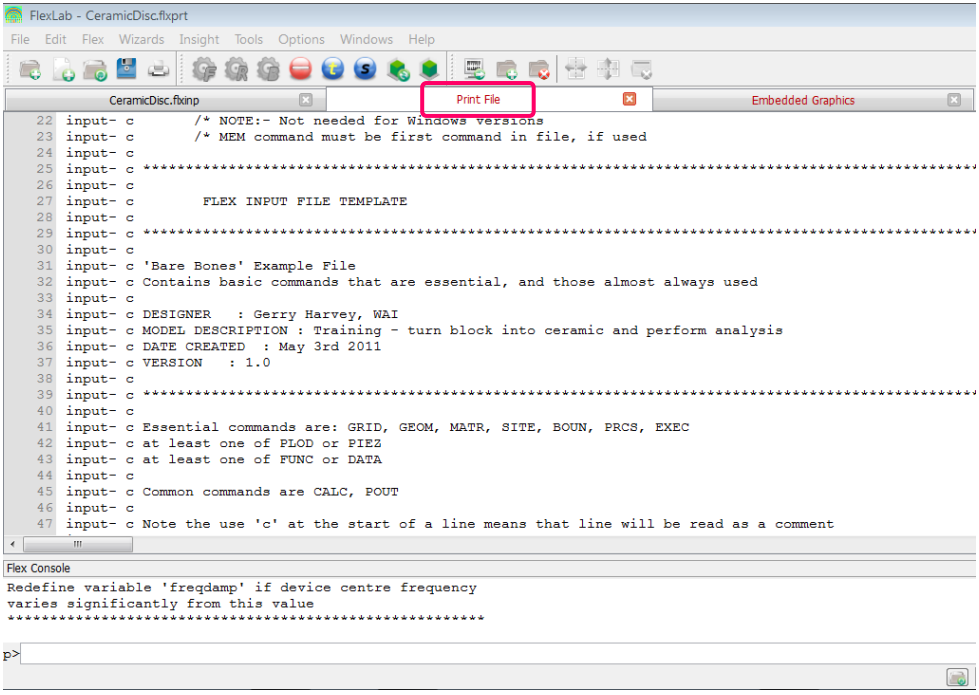
Command Override File

There are times when a **FLEX** job is running that a user needs to override the commands contained in the Job Input File and regain control of the job. Typically, a user might need to bring the program to a graceful exit as soon as possible since the hardware system is going to be shut down soon, someone else needs the system, etc. The functionality to perform this override is provided by means of the `flxkil.name` (kill command) file. Every few minutes or at the end of each time step, whichever is greatest, the local file directory for the job is searched for the `flxkil.name` file. If the file is found, the present **EXEC** command is truncated to the current number of time steps run, the present Job Input File is closed, and the `flxkil.name` file is opened as the new Job Input File.

If the user desires to terminate the job, the file need only contain the **STOP** command. Once the **STOP** command is read, the program will write a restart file and then terminate normally. It should be noted that any valid commands can be placed in the `flxkil.name` file, not just commands designed to terminate the job i.e. plotting commands to show results at that point. Once the job completes, the `flxkil.name` file is deleted automatically.

Print Files

To facilitate the modeling process, FlexLAB displays print files generated by PZFlex, Build, and Review. When you start a job, a window containing the contents of the print file is added to the display:

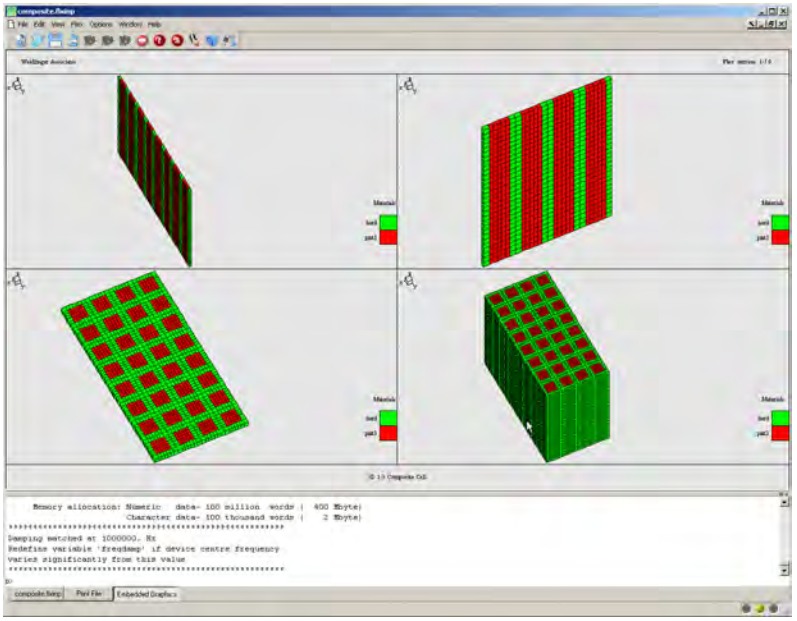


The red circle indicates the additional tab bar added to the window. You can switch among open windows by clicking the desired file. This allows you to check the print file as the model progresses, to make sure that the variables are correct, to see how many elements you are using, and to estimate running time.

Embedded Graphics

Because the graphics window is embedded in FlexLAB, all of the required tools are available in a single window. Therefore, if your model has graphical display commands in the input file, FlexLAB automatically opens a separate window to display the model graphics:

All of the functionality of the PZFlex graphics window is maintained. You can either enter



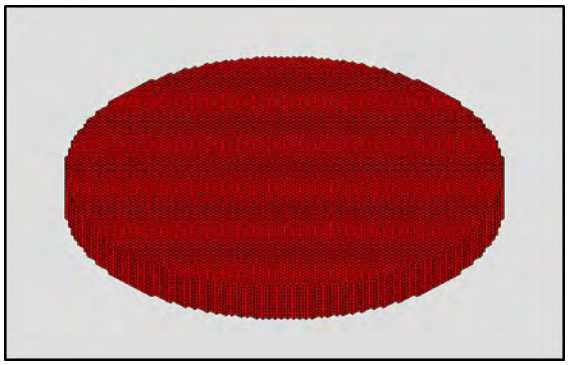
graphics commands in the text file or pause the text file with a “term” command to view the model with a command prompt.

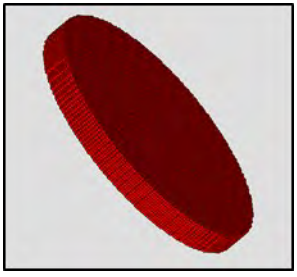
The toolbar has buttons for two common commands, “Redraw” and “Plot Materials”. These options are available only when the simulation program is paused.

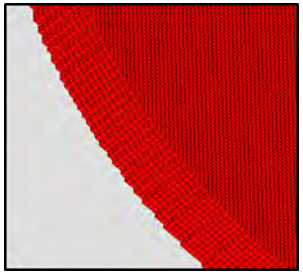
Interactive Graphics

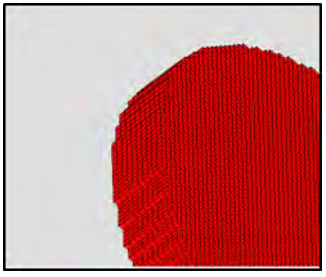
Along with the embedded graphics, Interactive graphics now allows the user to view the model in 3-dimensional space with the ability control the viewpoint through simple use of the right mouse button (RMB).

Original View





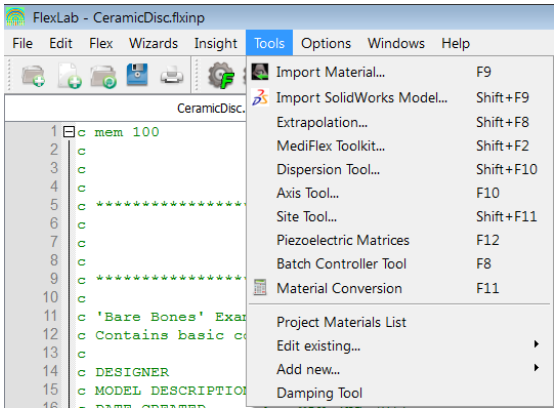




Commands: **RMB = Rotation** **CTRL + RMB = Zoom** **SHIFT + RMB = Pan**

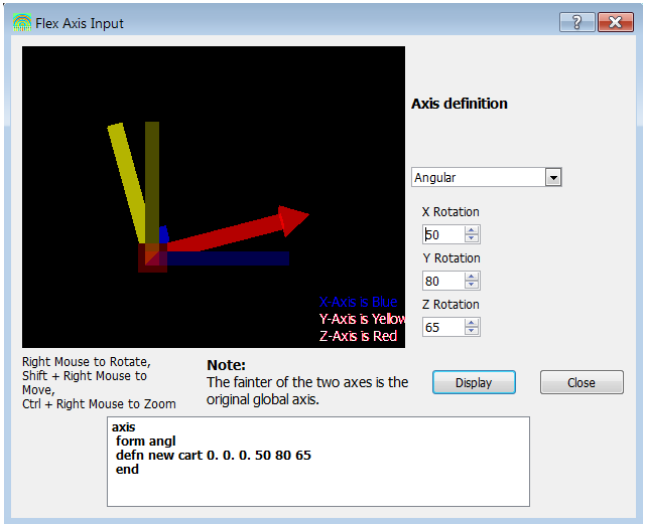
Tools

PZFlex includes a number of Tools that assist in modeling. You will find them in FlexLAB, under “Tools.”



Axis Tool

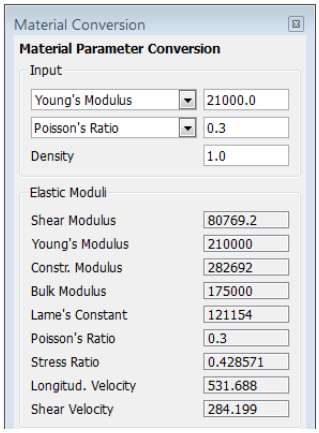
Tracking multiple vectors to represent a new coordinate system can be difficult, especially when the vectors do not follow the principal axes. The Axis Tool graphically displays a newly defined axis in any of the standard PZFlex definition methodologies—angular, coordinate, or vector. A 3D view highlights the new axis in relation to the global axis. The display includes the command syntax, so you can confirm that you have entered it correctly.



Material Conversion Tool

PZFlex accepts material properties for isotropic materials in two primary formats: as longitudinal and shear wave velocity and as bulk and shear moduli. Unfortunately, many manufacturers do not provide data in either format, requiring conversion before use. The Material Conversion Tool handles data in up to nine formats (bulk modulus, shear modulus, longitudinal velocity, shear velocity, stress ratio, constrained modulus, Lame's constant, Poisson's ratio, and Young's modulus).

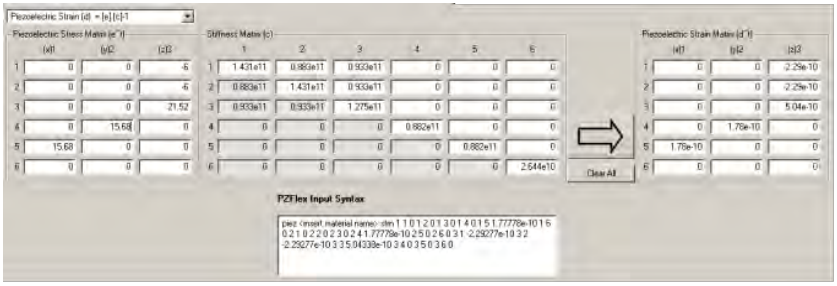
Supplied with density and two other variables, the Conversion Tool converts them to values appropriate for use in PZFlex.



Piezoelectric Matrices Tool

Extensive data are required to accurately simulate piezoelectric material. Three matrices (6x6, 3x6, and 3x3) must be populated; although not all entries are required, approximately 10 parameters are needed. Ceramic manufacturers provide data for these matrices in one of two formats. Basic mechanical properties, for example, are in terms of either stiffness ([c]) or compliance ([s]), and piezoelectric properties are in terms of either constant stress ([e]) or constant strain ([d]). Conversion is possible with simple matrix inversion and multiplication. This cannot be done quickly, however, without computational assistance.

The Piezoelectric Matrix Tool converts matrix data to other formats and prepares the appropriate PZFlex commands to ensure accurate entry of the data in a PZFlex materials file.



Damping Tool

PZFlex includes several advanced attenuation models to simulate the damping in a system. Damping, which varies by material and frequency, can be separated into longitudinal and shear components. It is not feasible for PZFlex to determine the damping characteristics at every frequency with complete accuracy. The Damping Tool enables you to visualize the damping that has been applied to the model and to make any changes to better match the damping of the actual material.

The Damping Tool allows you to enter all the critical variables used in PZFlex attenuation routines such as damping mechanism, longitudinal and shear attenuation base values, and frequency coefficients. It displays in graph form the attenuations against frequency, as well as the velocity variation due to the dispersion in materials with very high attenuation. These potentially critical parameters are often ignored.

Material Image Import Tool

PZFlex quickly models structures composed of regularly shaped components such as cuboids and cylinders. It is more complicated, however, to model structures that consist of irregularly shaped components and scatterers, such as those in tissue. Although the digitization of optical images, followed by manual conversion to a compatible finite-element format, can produce useful material maps, this is a painstaking, inefficient method.

The Material Import Tool takes standard digital images, such as JPGs, and converts them to materials layout files. By manually varying the range of color or grayscale of the materials, you can quickly create complicated models at a resolution equivalent to that of the original digital images. All types of images are useable, including photographs and images created in standard drawing packages.

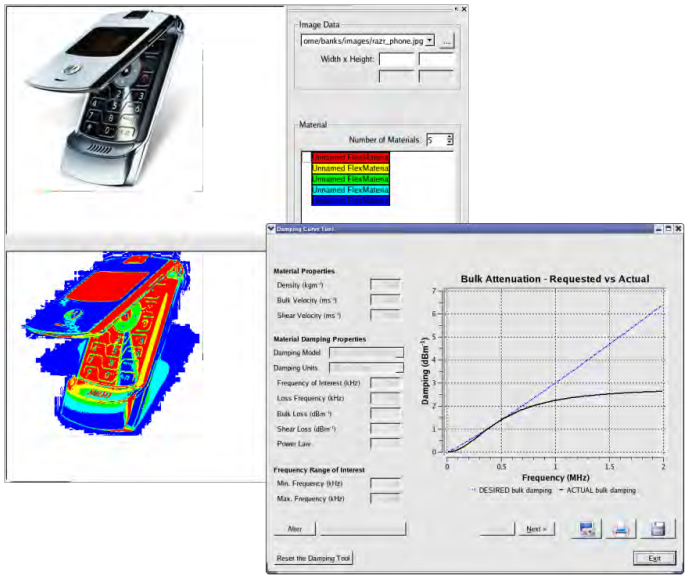
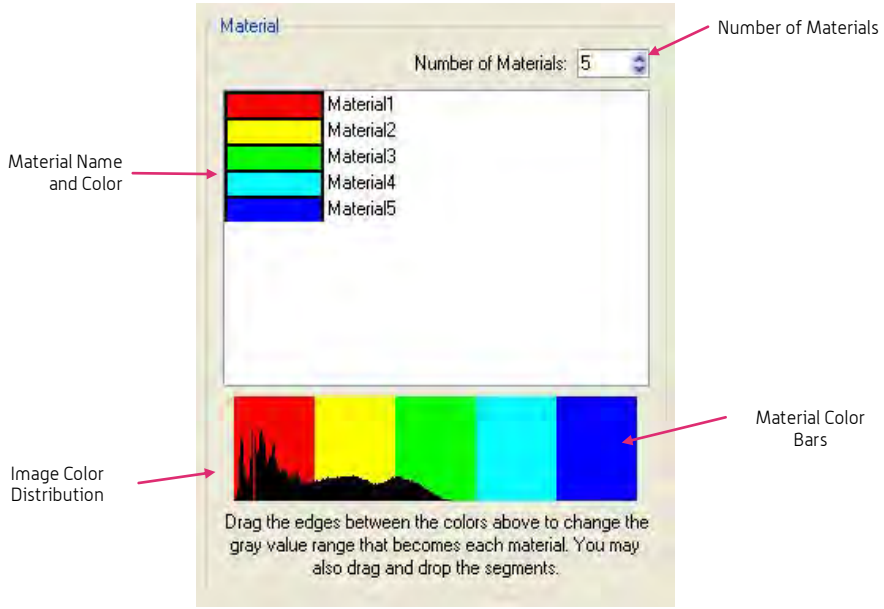


Image Importation Tool

Although PZFlex excels at the rapid building of regularly shaped objects and structures, the construction of models with extremely rough surfaces and heterogeneous media can be difficult for non-expert users. The Image Importation Tool simplifies the process by allowing any digitized image (JPG, PNG, etc.) to be converted to a PZFlex material map based on color or grayscale level. This allows, say, an optical image of a tissue slice to be quickly adapted for use in PZFlex.

The Image Importation Tool is accessed in FlexLAB, under “Tools.”

To read in the source image, select the File Input option at the upper right of the tool. Most digitized image formats such as JPG and PNG are supported. The tool attempts to read any size data (x by y dimensions) embedded in the image file. If it is unable to do so, enter the size in the “width” and “height” boxes. As these dimensions will be used to construct the PZFlex material table file, they should be those you wish to use in the model.



Material Selection

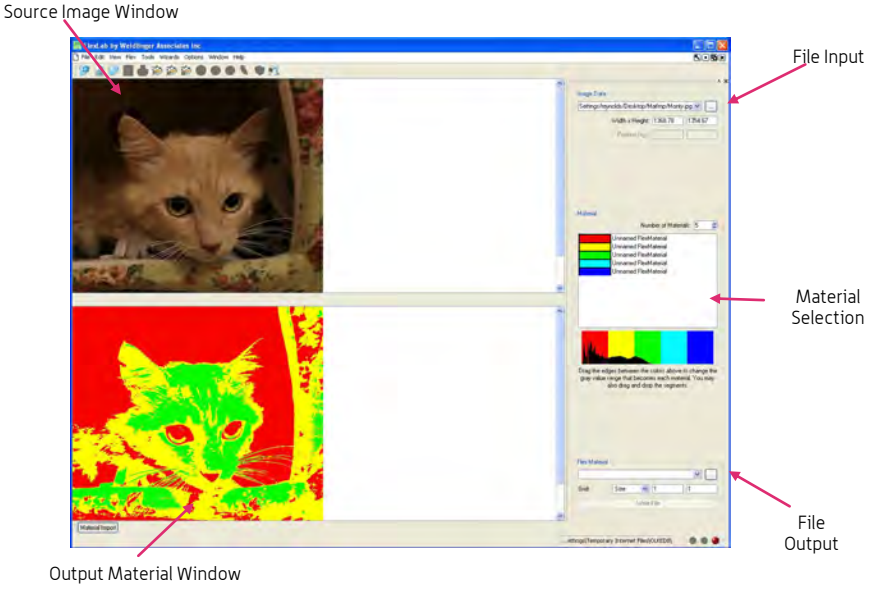
Once the digitized image has been imported, select the color boundaries for each material. The image is initially divided into 5 materials; this can be increased to as many as 12 with the box at the upper right of the materials section. Enter the names of each material next to its representative color. The same name may be used multiple times for materials in non-adjacent color ranges. Only material names are defined at this stage; the material properties will be defined in the PZFlex file.

The bar at the bottom of the material section is interactive, allowing you to manipulate the color ranges and corresponding locations. You should see a black contour surface that represents the color content of the input image. By default, the color bars are evenly spaced across the surface. Placing the mouse on the border of two colors allows you to drag the border to the location of interest. The output file image at the bottom left of FlexLAB changes, showing the result of the dragging.

Once you have manipulated the color ranges to your satisfaction, convert the image to a PZFlex-compatible file format. Select the output discretisation of the file in the output

section at the bottom right of the FlexLAB window. This discretisation need not match the discretisation of the PZFlex file into which it will be imported. Note that selecting a discretisation beyond that of the original image does not improve resolution, but merely creates a larger file. After selection of the size, the output image at the lower left of the FlexLAB window rasterises into a true representation of the output file that will be written. Click “Save File” when you are satisfied with the result.

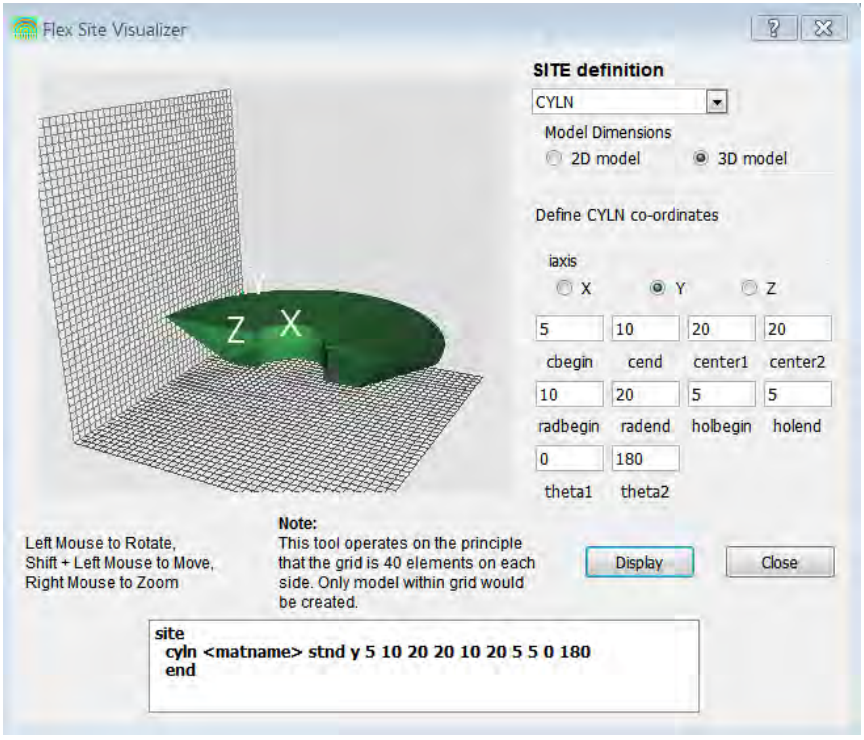
The discretised and segmented model is then written to a PZFlex-compatible file (detailed in the manual under TABL format). This file can be read into a PZFlex input file using the “site tabl in” commands. A template file, called `importer.flxinp`, provides an example of how such files can be read in.



Basic Layout of Material Importation Tool

Site Tool

The site tool shows how certain subcommands function and also offers you the chance to visualise what will be created in your model if used. The tool will allow you to generate, within a 2D or 3D grid, various shapes using the subcommands of the SITE primary command.



For example selecting the CYLN command, you will find that you are able to generate interesting shapes and not just a simple cylinder. Each of the input parameters can be easily tweaked such as ‘theta1’ and ‘theta2’ which you will find, dissects the cylinder at the specified angles. Also in this environment, any parameter that you are unsure of, you can easily change the input values and clicking the ‘Display’ button will show you their effects.

Once you have generated a shape you would like to use in your model, the code segment at the bottom can be copied and used in your flex input file. As the site tool only generates the dummy model in a 40 element sided grid, you are required to scale it to the sizes in your model or replace the parameters with your key points instead.

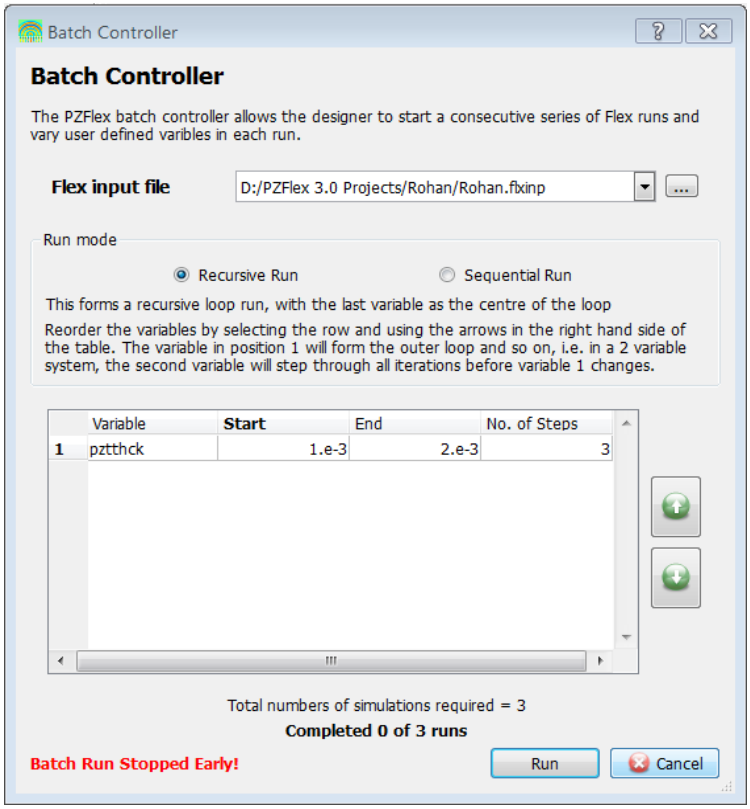
Batch Controller Tool

The batch controller tool offers a simple way to perform a series of flex runs and allows the user to vary defined variables in each run.

In order for the user to control the variables to sweep through for each run, in your flex input file, the variables must be declared as the following:

```
symbx pztthck = 1.e-3
```

Selecting the batch controller tool will open the following window:

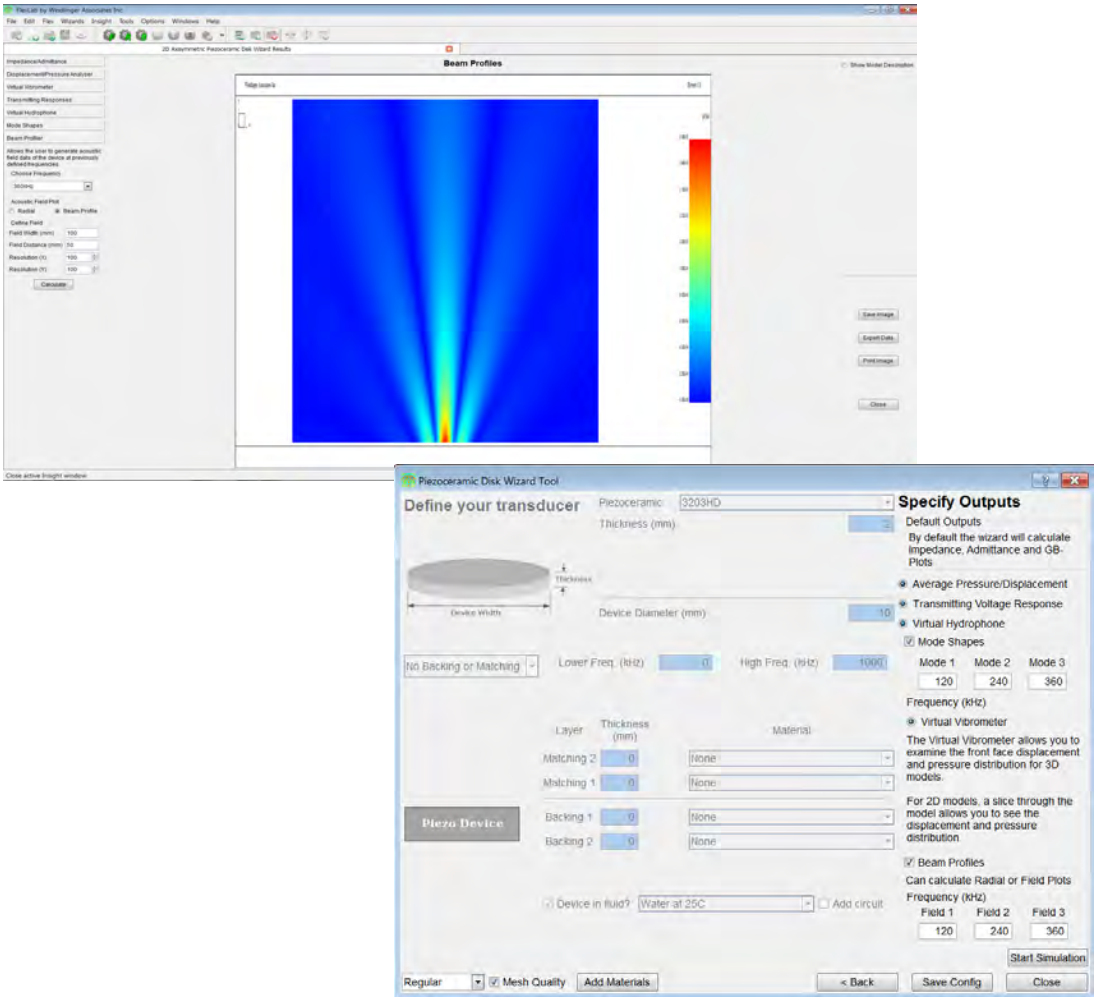


Once you have selected your flex input file, the sweep variable will appear in the window below allowing you select the range and step size which dictates the number of runs in the batch. The 2 run modes allows either a ‘recursive’ manner where all combinations of variable changes are explored or a ‘sequential manner’ where the variables will be altered in order they appear. The runs can be stopped at any time using the cancel button.

Wizards

Sometimes you are interested only in simple answers and common metrics from standard devices, in a format that makes it easy to enter and retrieve data. PZFlex's Wizards (in the FlexLAB toolbar) allow you to simulate a wide range of standard devices, including 1-3 and 2-2 piezoecomposites and bare disk and plate piezoceramics. The Wizards use a simple but elegant GUI and comprehensive suite of templates to produce accurate simulation results in seconds.

Many common devices are included in the Wizards list, which continues to expand to encompass a wider range of industries and applications.

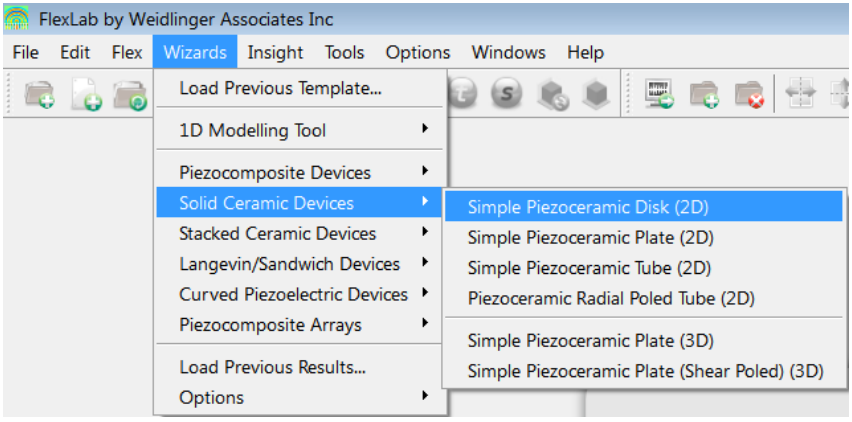


Wizards include:

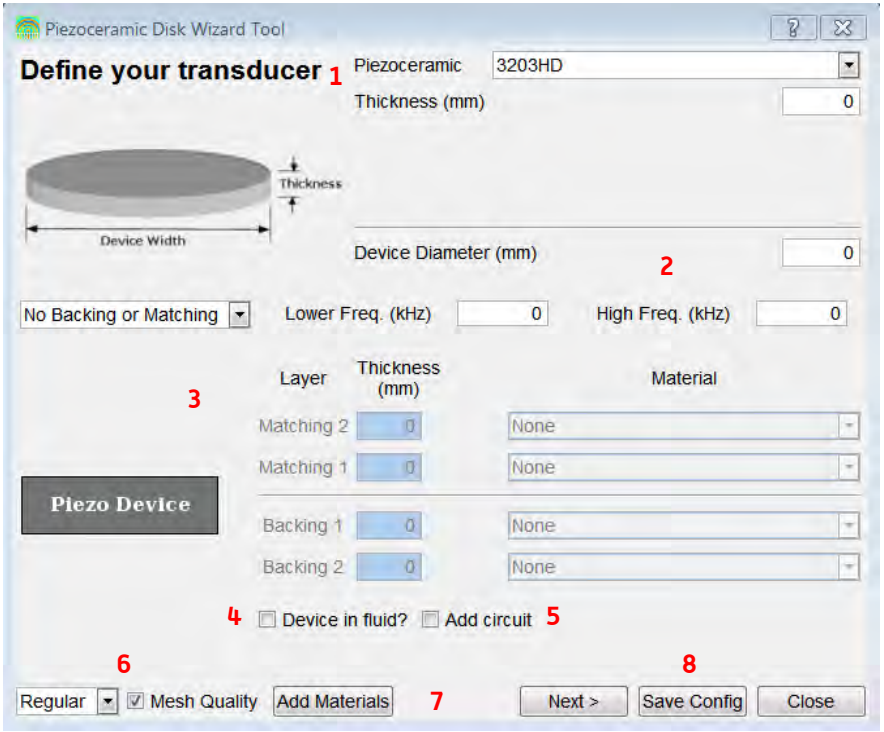
- Piezocomposite devices in 2-2 and 1-3 arrangements
- Solid Caramic devices with variable shapes
- Stacked Ceramic devices in few shapes and forms such as bolted tonpilz.
- Langevin/Sandwich devices
- Curved Piezoelectric devices

Once the wizards are activated, a new window appears allowing users to simply input their desired parameters to create the model. With the simulation executed, an intuitive post-processing menu allows you to select the desired results from the model such as mode shapes, transmit voltage responses and beam profiles.

Using Wizards— Simple Piezoceramic Disk (2D)



The wizards are accessed in the top menu tab marked ‘Wizards’; there are a wide selection available that all clearly named. For this example, we will select the Simple ‘Piezoceramic Disk’ wizard within the ‘Solid Ceramic Devices’ group which will open the following interface:

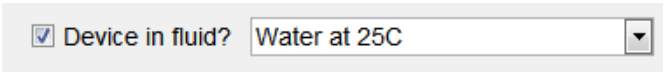


The interface is very simple to use: it contains most of the common settings for a simple ceramic disk model.

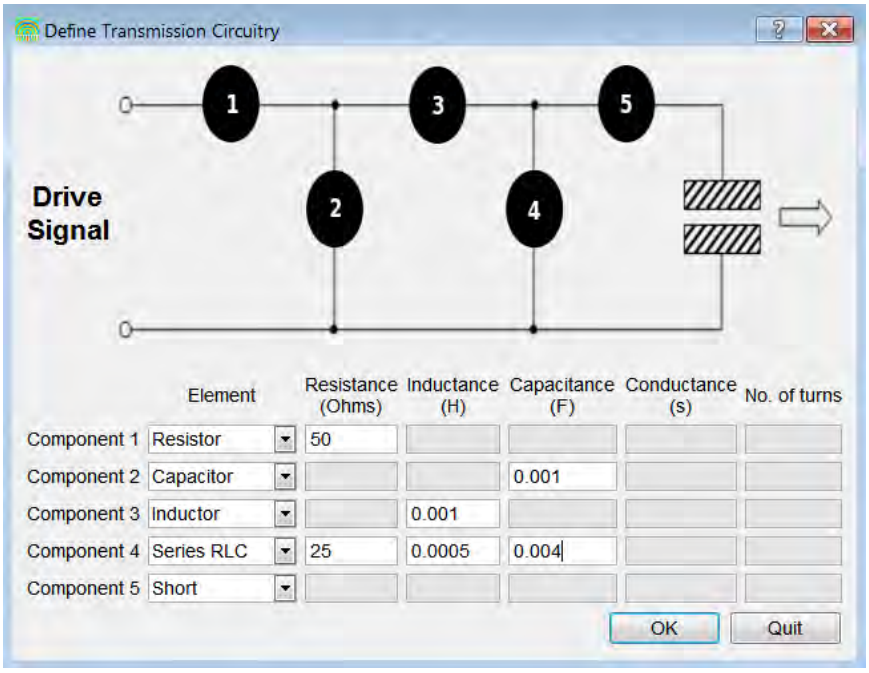
1. From the pull down menu you can select from the existing database of Piezoceramic materials. The thickness of the device can be set underneath—dictates the 'High Freq' in area 2.

2. The device diameter and the frequency range of the model are the following inputs for the model. Be aware that the maximum frequency has been calculated based on the thickness and material of the device so choose below the value that is already shown.

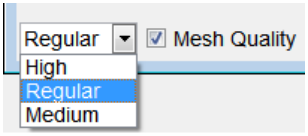
3. Matching and backing layers can be added to the model by selecting the from the drop down menu. For this particular wizard, you can select any combination of 2 matching and 2 backing layers (Max 4 layers). The input parameters will then be available for you to tweak accordingly: selecting the thickness and material type. The diameter of these materials will be the same as the 'device diameter'.



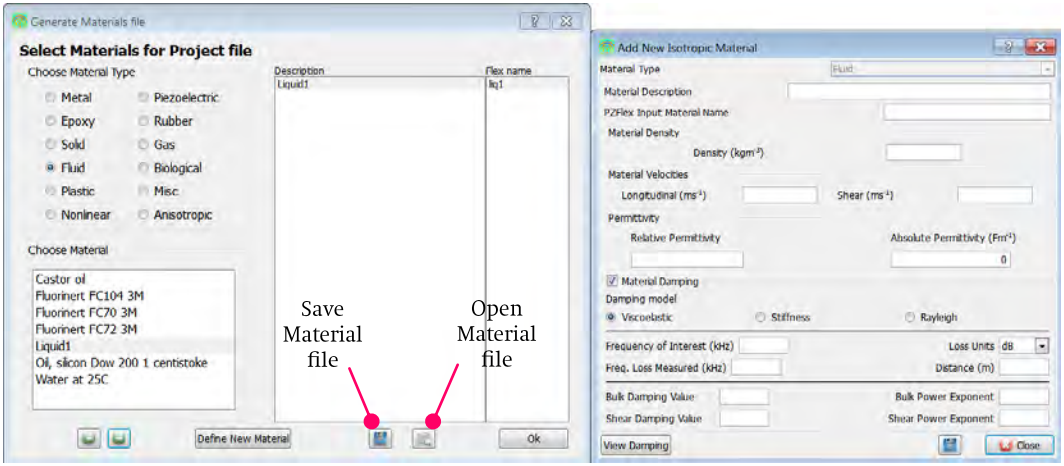
4. Checking the 'Device in fluid' box will show another drop down menu for you to select the type of fluid to model your device in.



5. Checking the 'Add circuit' box will allow you to define a circuit to drive your device. Common circuit components are available: all that is required is for you to set the component values and choose the location based on the circuit diagram shown.

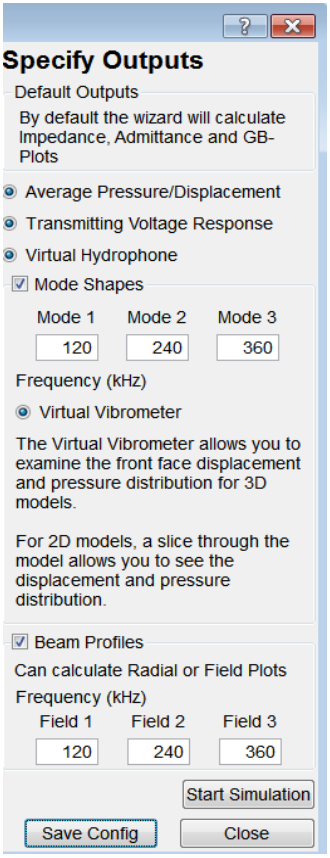


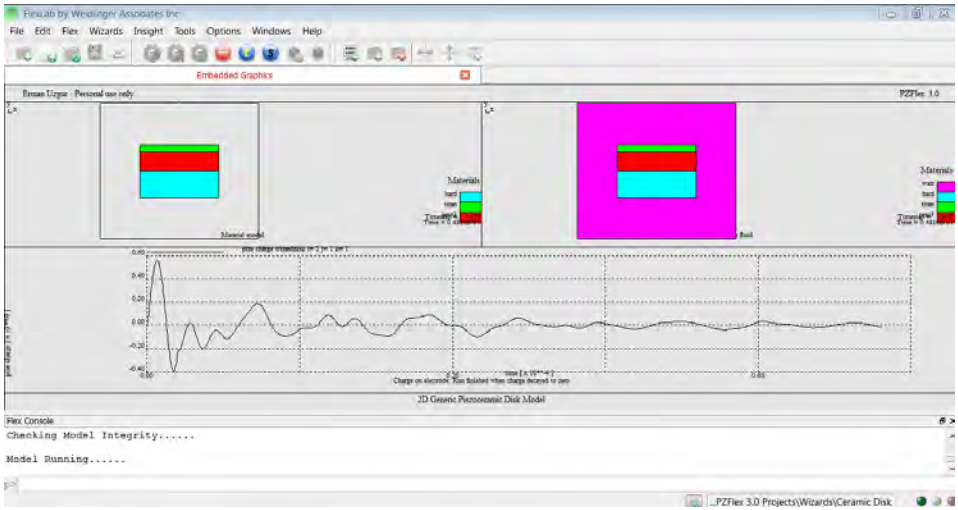
6. 'Mesh Quality' dictates the resolution of the model (elements / wavelength). For a higher quality model select 'High' and for quicker simulation select 'Medium'.



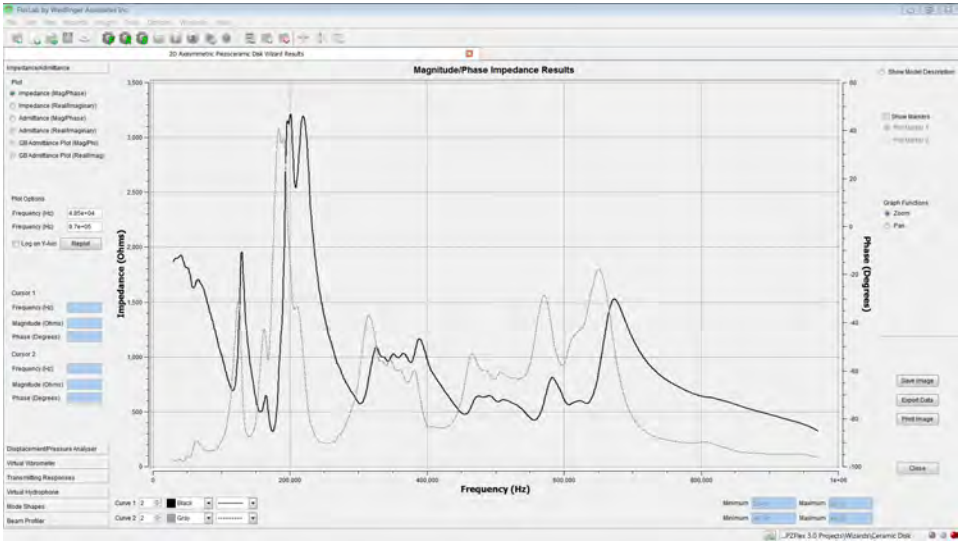
7. The ‘Add Materials’ button will open the interface for creating your own material file and defining new materials. The materials within the PZFlex database can all be altered and added to your custom material file. To define a new material, click on the ‘Define New Material’ button and a new window will open allowing you to select the material type and various input constraints associated with the material. Clicking on the materials will add the materials to the list and with the disk icon below, you can to save the material file. An existing material file can also be opened and edited.

8. With the inputs of the model all set, clicking on the ‘Next’ button will reveal the ‘Specify Outputs’ section of the wizards. The outputs will be the data calculated during the model run. The outputs available are: Average Pressure/Displacement data, Transmitting Voltage Response, Virtual Hydrophone (finding pressure data at specific points), Mode Shapes (at 3 different frequencies), Virtual Vibrometer (explained in wizard directly) and beam profiles. By default, Impedance, Admittance and GB Plots are calculated. With the desired outputs selected, you can save the configuration, allowing you to return to this point.





Clicking the 'Start Simulation' button will begin running the model and will automatically display the post processing interface once it finishes.



The general layout of the post processing window are the various tabs for the specified outputs on the left-hand side and the generic tools on the right such as saving data, saving graph and printing images. The graph shown above is the impedance profile of the device.

Generic Tools

The interface offers general tools you can use to help you analyse or customise the data. Using the 'Curve 1/2' settings you can alter the curve's line width, colour and line type to help distinguish between multiple data sets plotted on the same graph.

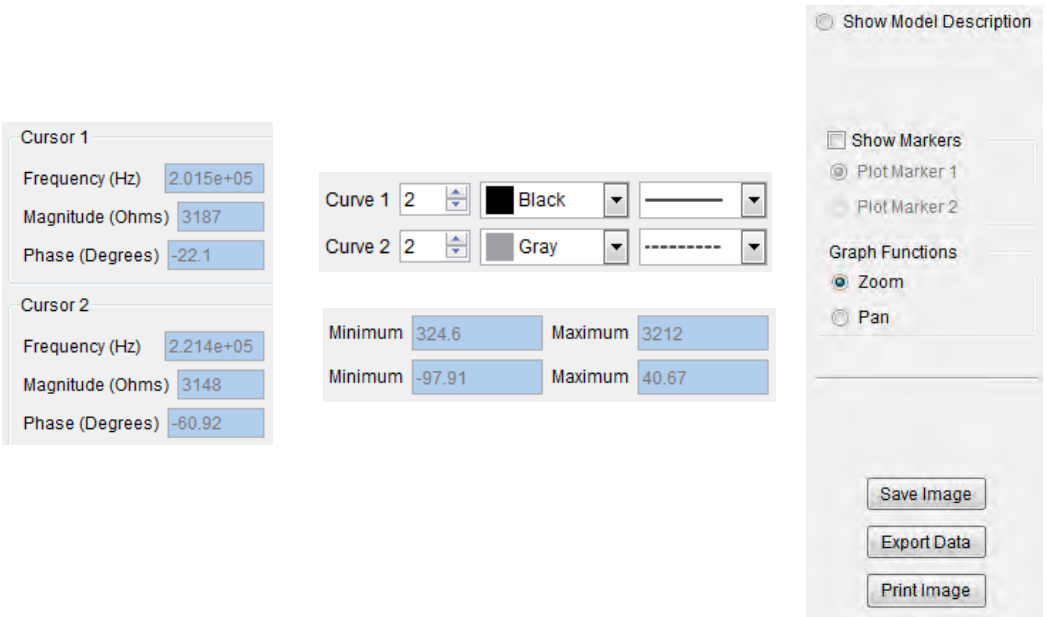
For a refresh on what has been modelled, there is a 'Show Model Description' checkbox in the top right hand corner which offers a description of the model.

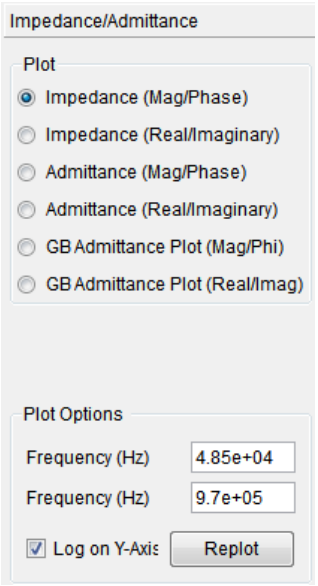
Plot markers are also available to give specific data readings on the graphs. There are two cursors which can be used simultaneously on the one graph. Clicking on specific points on the graph will make them appear and data can be read from the cursor readings on the left side of the interface. There are also Minimum/Maximum values of the plotted graph shown at bottom right corner of the graph.

Using the right mouse button, you can highlight an area to zoom or pan around the point you have clicked. The two options are chosen on the right side of the interface.

To save graphs, export or print the current data, there are separate buttons to do so.

All the mentioned Generic Tools will be found in most of the post-processing tabs with a slight variation depending on the data being extracted from the model.



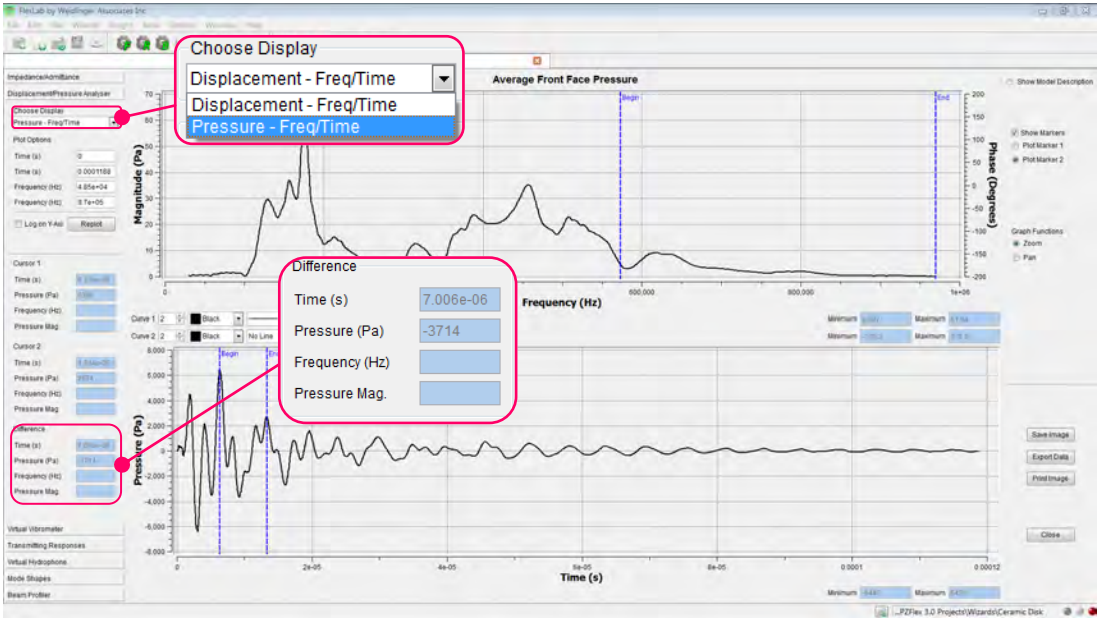


Impedance/Admittance

In this output tab, you can observe the impedance and admittance data of your device. You can select from the range of graphs to be displayed in either magnitude/phase or real/imaginary terms. There is also an option to plot the Y-axis on a logarithmic scale.

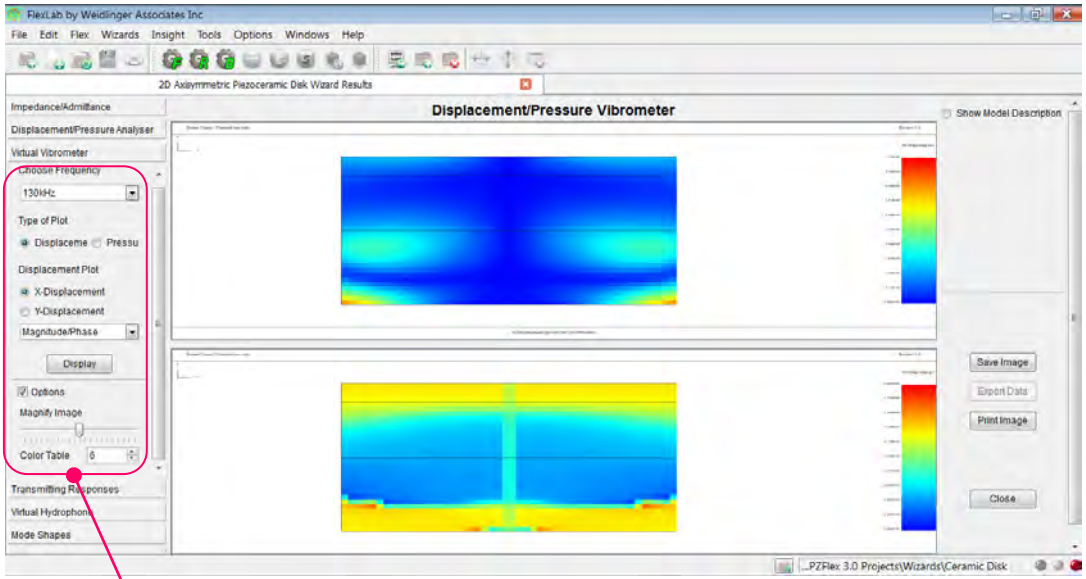
Displacement/Pressure Analyser

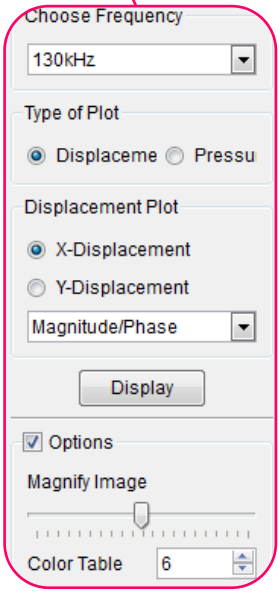
The layout remains similar to the impedance/admittance output tab. All the generic tools are available. You can select from the drop down menu to either display the average front face displacement or the average front face pressure. The data cursor readings on the left has an added section for displaying the difference between the marker points.



Virtual Vibrometer

The virtual Vibrometer allows you to examine visually the front face displacement and pressure distribution for 2D and 3D models. For a 2D model, a slice will be taken through the middle to gain the displacement and pressure distributions.

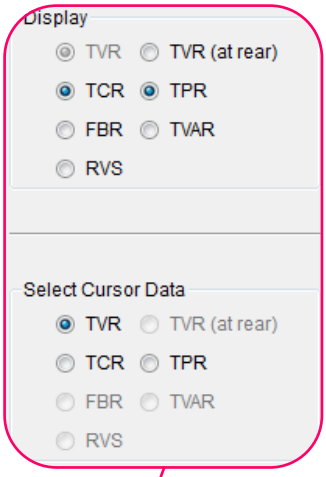




The drop down menu in this post-processing tab lets you select the frequencies (specified at the 'Specify Outputs' section prior to starting the simulation) at which the displacement/pressure data will be generated. Clicking on each different selector will allow you to vary between the displacement or pressure data as well as X and Y direction displacement (not applicable to pressure data). Another drop down menu allows you to choose the data to be plotted in magnitude/phase or in real/imaginary terms.

With the settings confirmed, clicking on 'Display' will plot the displacement or pressure fields. Additional options allows you to magnify the image or select a different colour scheme for plotted data. A set of these images will be available in the directory where you saved the wizard configuration file.

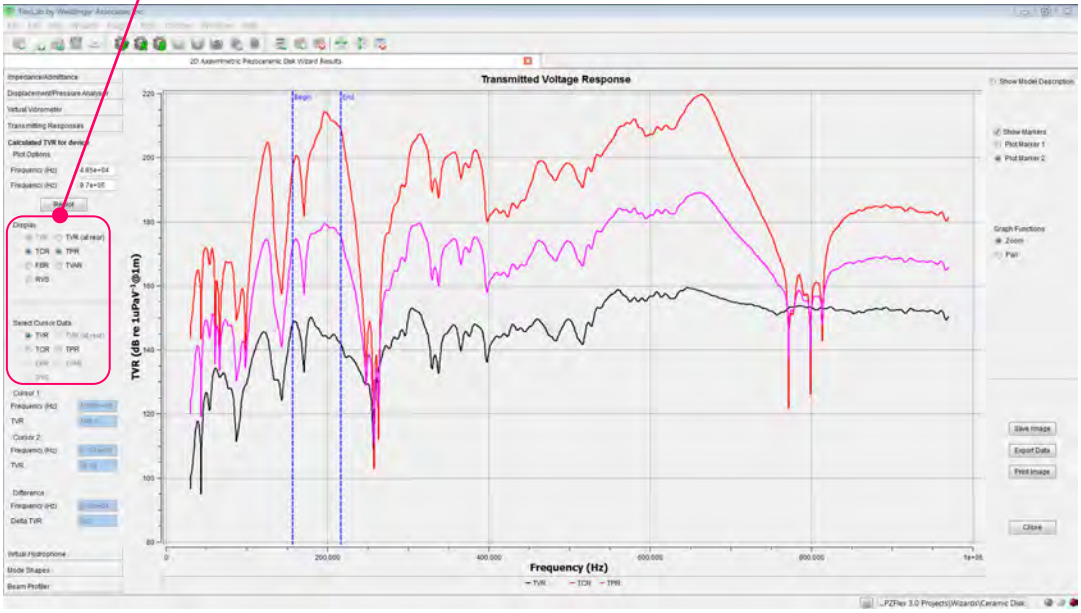
Transmitting Responses



The options for the transmitting responses differs slightly again. You will find that you can select from a range of responses including:

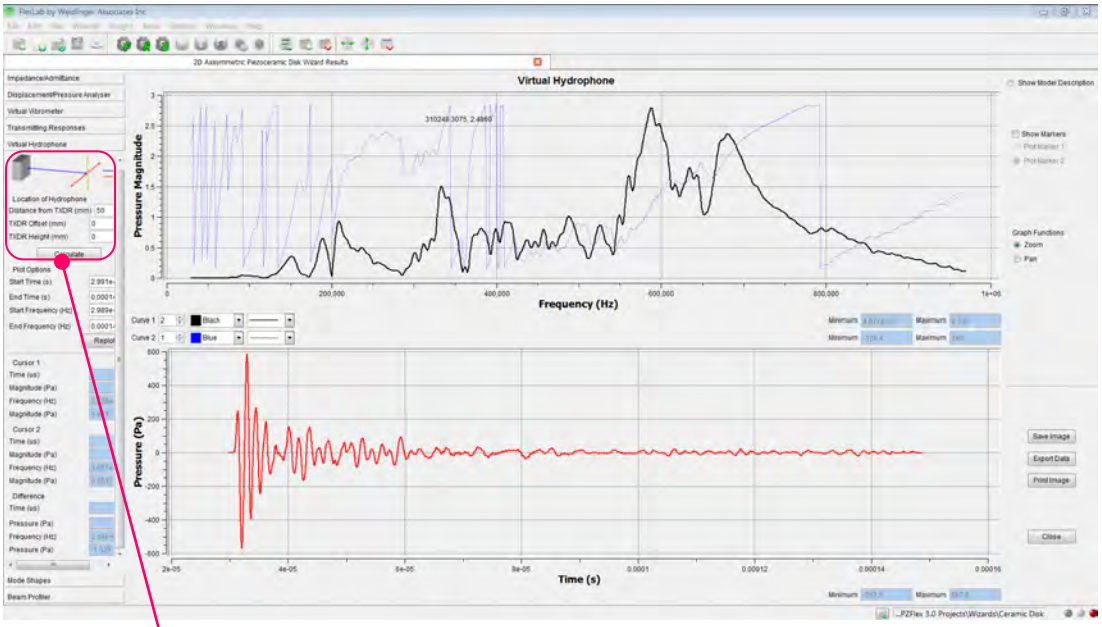
- TVR (default)
- TVR (at rear)
- TCR
- TPR
- FBR
- TVAR
- RVS

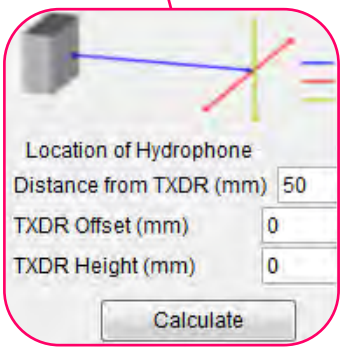
For each response selected, the curve will be plotted on the same graph with a predefined colour. The legend below the x-axis will show the colour corresponding to the data curve. The plot markers can again be used to identify specific data points on the graph however, since there may be more than one curve on the graph, there is an option to select which response you are using the data cursors for.



Virtual Hydrophone

The virtual hydrophone as the name suggests enables you to find the pressure data a specific distance from your device in the fluid you have chosen selected for the model.





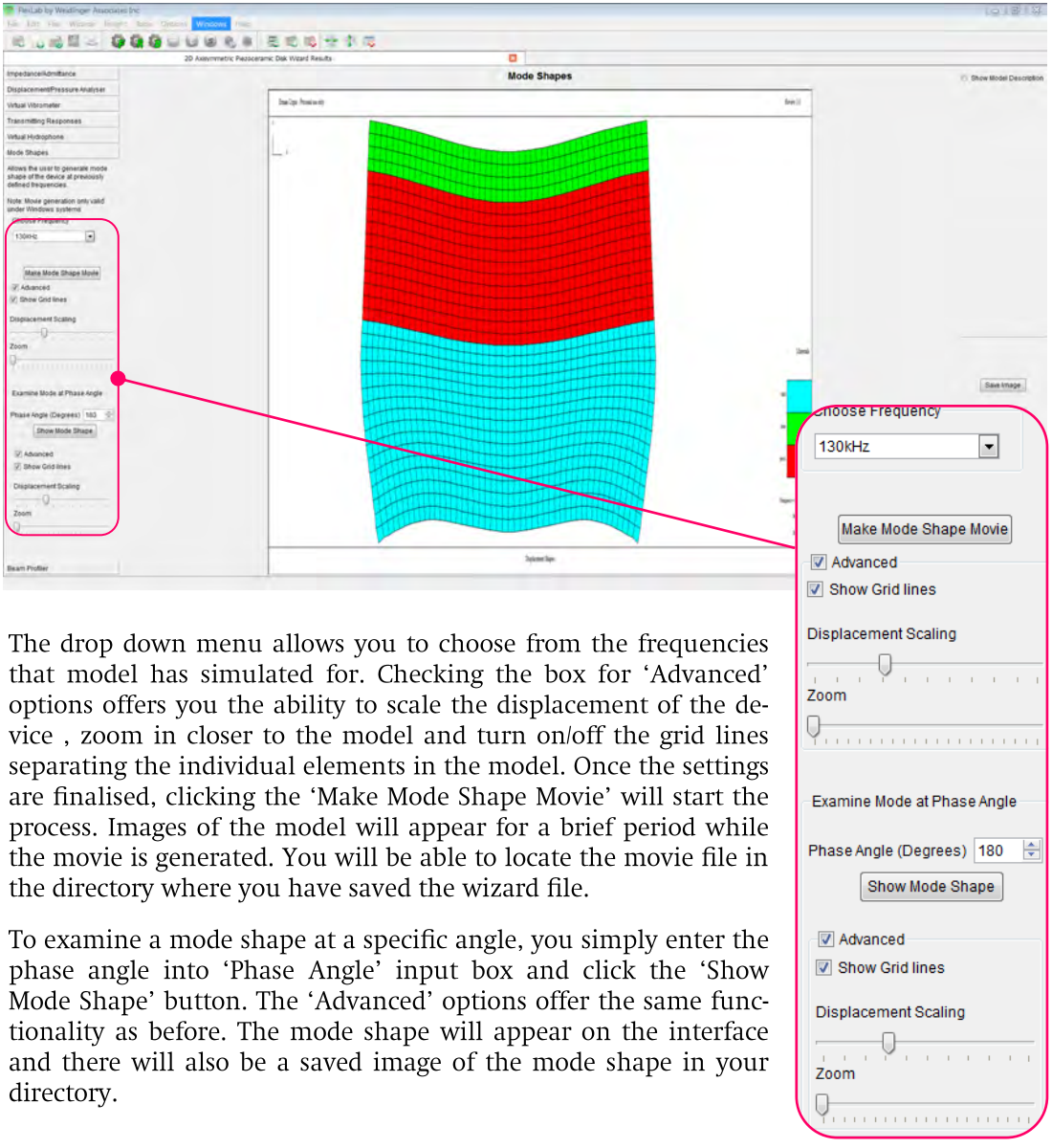
To find the pressure of a particular point in the model, you require 3 physical dimensions referenced from the centre of the device's front face:

- The distance from the centre point in the direction of propagation (blue)
- The horizontal distance from the centre point (red)
- The vertical distance from the centre point (green)

With the distance entered in the input box, clicking the 'Calculate' button will generate the pressure data.

Mode Shapes

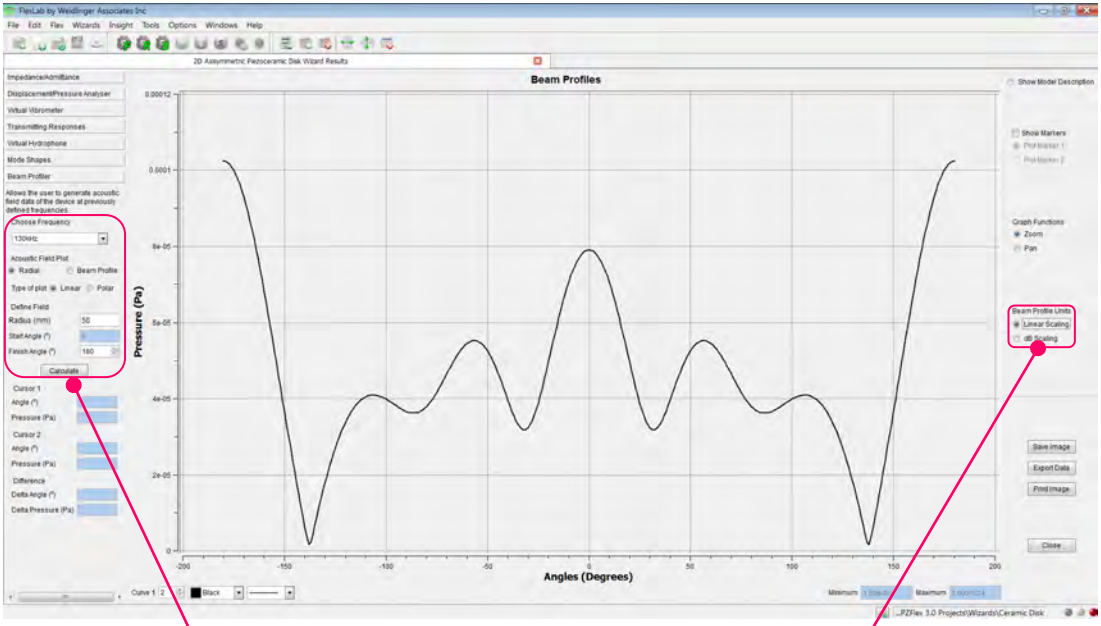
The mode shape post-processing tab is where you are able to generate mode shape movies at your specified frequencies. You can also extract the exact mode shapes at particular phase angles.

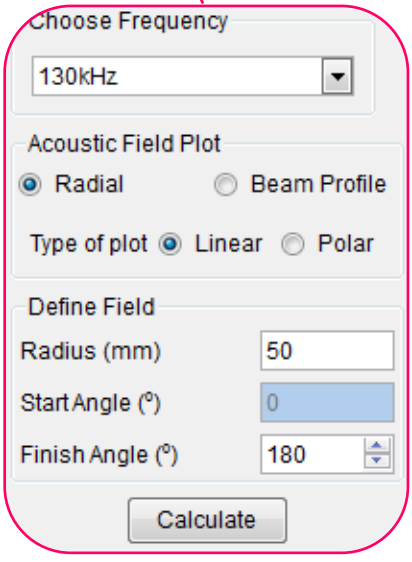


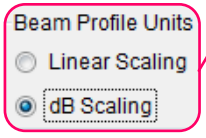
The drop down menu allows you to choose from the frequencies that model has simulated for. Checking the box for 'Advanced' options offers you the ability to scale the displacement of the device , zoom in closer to the model and turn on/off the grid lines separating the individual elements in the model. Once the settings are finalised, clicking the 'Make Mode Shape Movie' will start the process. Images of the model will appear for a brief period while the movie is generated. You will be able to locate the movie file in the directory where you have saved the wizard file.

To examine a mode shape at a specific angle, you simply enter the phase angle into 'Phase Angle' input box and click the 'Show Mode Shape' button. The 'Advanced' options offer the same functionality as before. The mode shape will appear on the interface and there will also be a saved image of the mode shape in your directory.

Beam Profiler



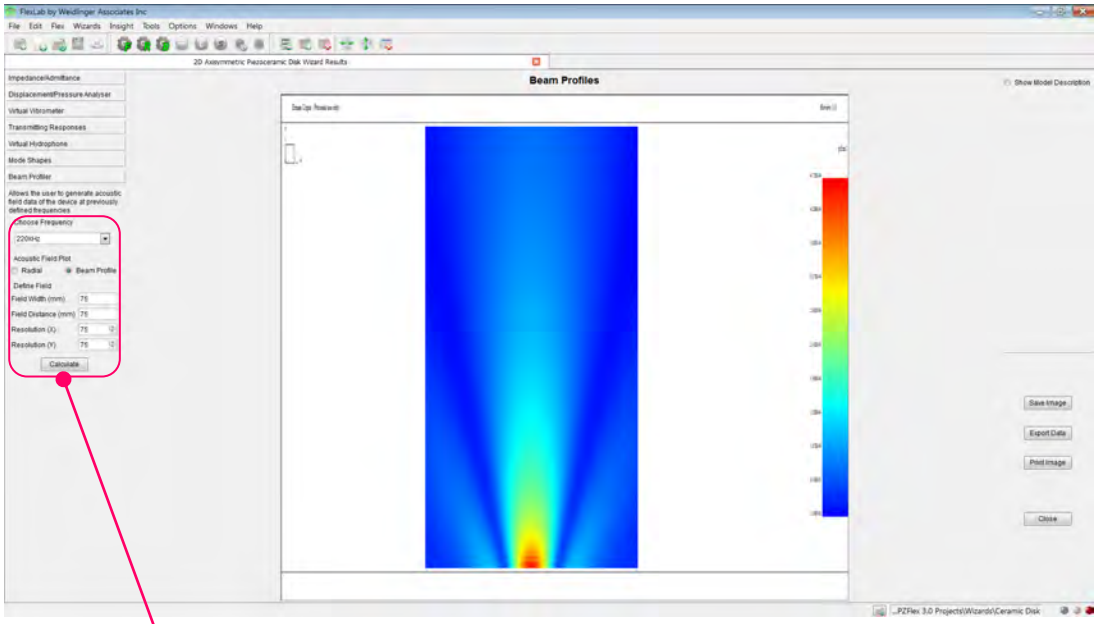


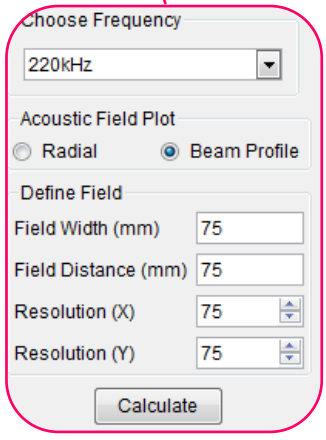


There are 2 types of acoustic field plots that can be obtained: radial acoustic field and beam profile plots.

Again, the frequency can be chosen from the drop down menu which applies to both radial and beam profile plots. The type of graph (Radial plot only) can be selected: linear or polar. Additional settings for the radial plot include the radius of the defined field and angle to sweep across. 'Calculate' will then generate the plot.

There is also an option to plot the y-axis on a dB scale.





Beam profile plots are very easy to set up. To create a 2D slice of the pressure field through the centre of the device, the dimensions of the pressure field are required: the width and distance of the field. There are also resolution parameters which controls the detail of the beam plots.

The beam plot should appear on interface after it has been calculated. The last beam profile generated will be saved as an image in your directory.

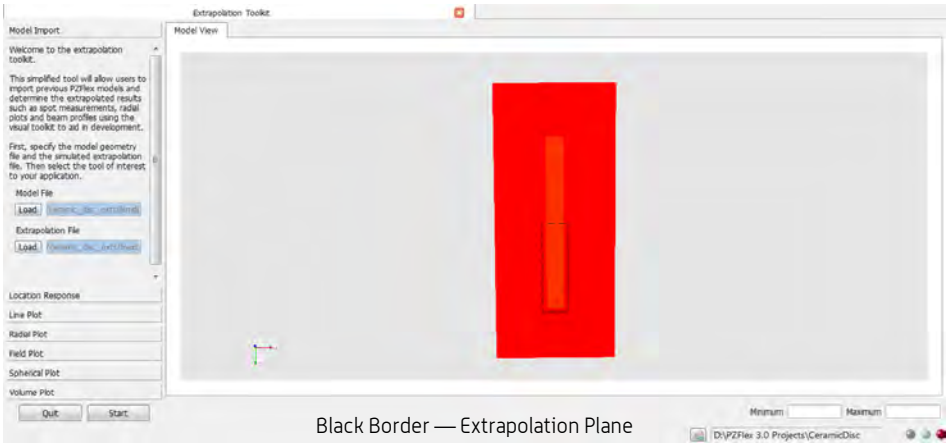
Settings—Flex Wizard Options

Within the main PZFlex settings there are options that governs its basic functionality:

- The number of CPU cores used for Wizard projects
- Default units used when plotting data
- The accuracy of values displayed (values shown beyond decimal point)
- Default background colour for plots
- Option to delete print files after wizard run finishes

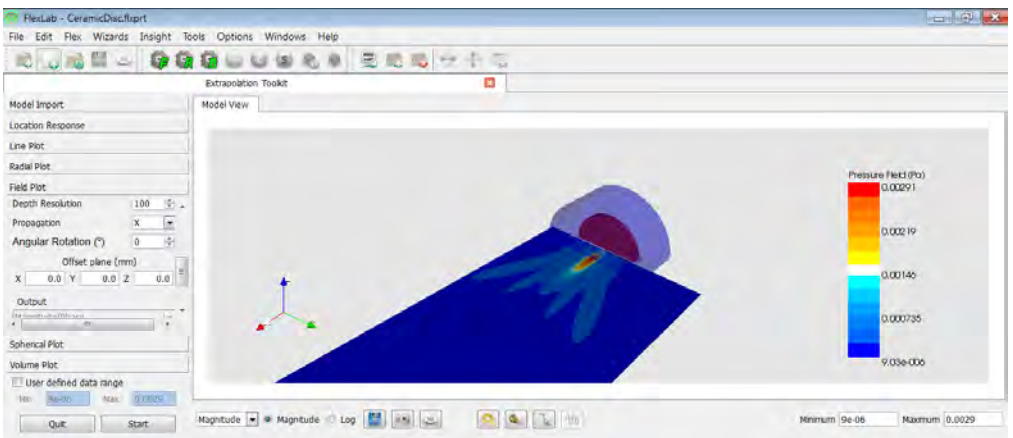
Extrapolation Toolkit

Extrapolation is a highly useful tool which allows various useful outputs to obtained from a simple model without modelling excessively large 3D models. The pressure history of each point on the extrapolation plane is stored and is treated like a source. The contributions of each source is then summed around the extrapolation plane. This process allows data to be obtained from beyond the confines of what has been modelled.



Black Border — Extrapolation Plane

A variety of plots can be obtained as outputs from the Extrapolation toolkit: Line plots, Radial plots, Field plots, Spherical plots and Volume plots. A location response is also available.

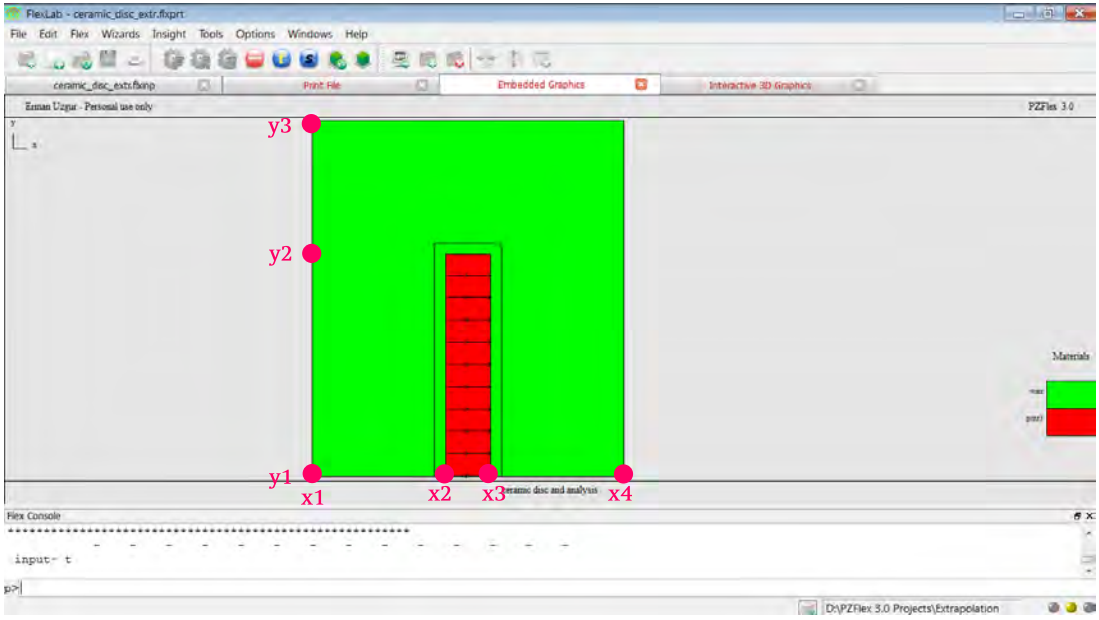


Using Extrapolation

In order to use the extrapolation toolkit, you must define an extrapolation boundary around your piezoelectric device.

```
extr
    ref in $x2 $y1 0
    defn kirc
    node $i2-5 $i2-5 $j1 $j2+5
    node $i2-5 $i3+5 $j2+5 $j2+5
    node $i3+5 $i3+5 $j1 $j2+5
    driv func
    end
```

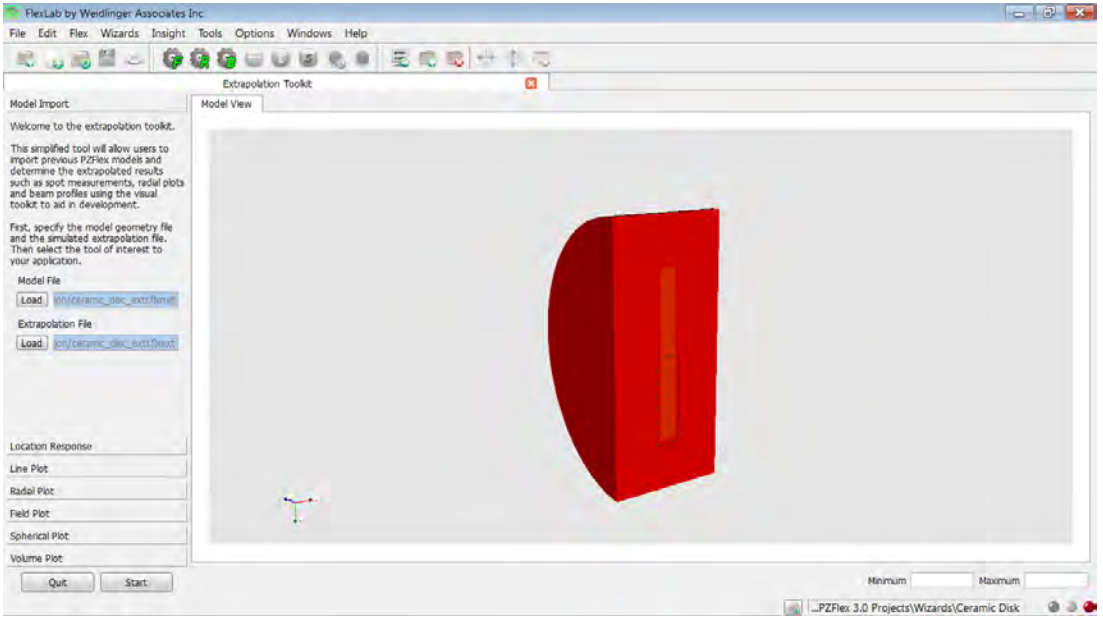
Using the EXTR primary command will allow you to set the extrapolation boundary around device. The REF subcommand is used to specify a reference point which outlines the interior and exterior sides of an extrapolation surface. DEFN is used to define the type of extrapolation surface which in this case is a Kirchoff extrapolation surface. Using the NODE subcommand, we identify the nodes around the device that will become the extrapolation surface. The idea is to keep the boundary at a close distance to the source (**5 elements**) to maintain a level of accuracy and minimise simulation runtime.



Note: To check if your extrapolation boundary is as you expect, use the GRPH— DRAW command to draw the lines to represent the declared nodes used for extrapolation.

The DRIV subcommand adds the drive function information to the extrapolation file which can be used to compute the Transmit Voltage Response (TVR) in the Extrapolation toolkit.

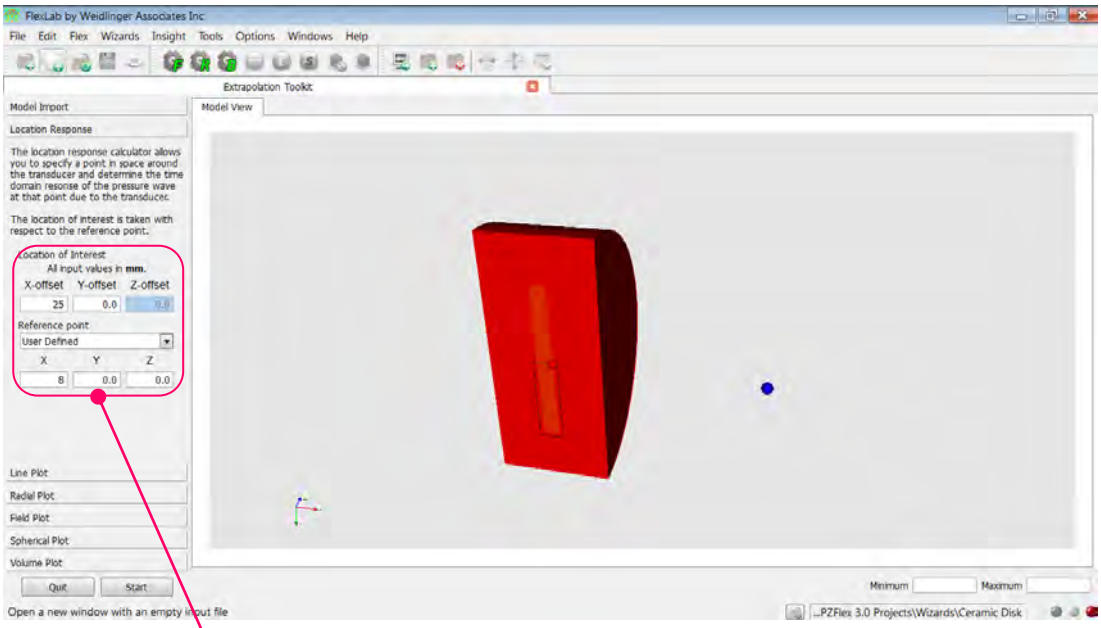
After successfully running your model, open the extrapolation toolkit which will require you to load 2 files into the toolkit: the model file (**.flxmdl**) and the extrapolation file (**.flxext**). Click on the load buttons and browse your save directory for the files.

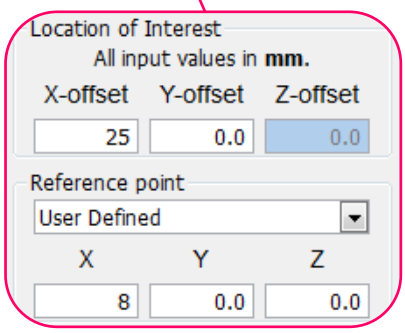


Your model will appear on the interface. Using your mouse and keyboard, you are able navigate around and examine the model:

- Left Mouse Click (Hold) + Move — Rotates model in all directions
- Right Mouse Click (Hold) + Move Up/Down or Mouse Scroller — Zoom in/out of model
- CTRL + Left Mouse Click (Hold) + Move — Rotates model about a point
- SHIFT + Left Mouse Click (Hold) + Move — Pan around model

Location Response





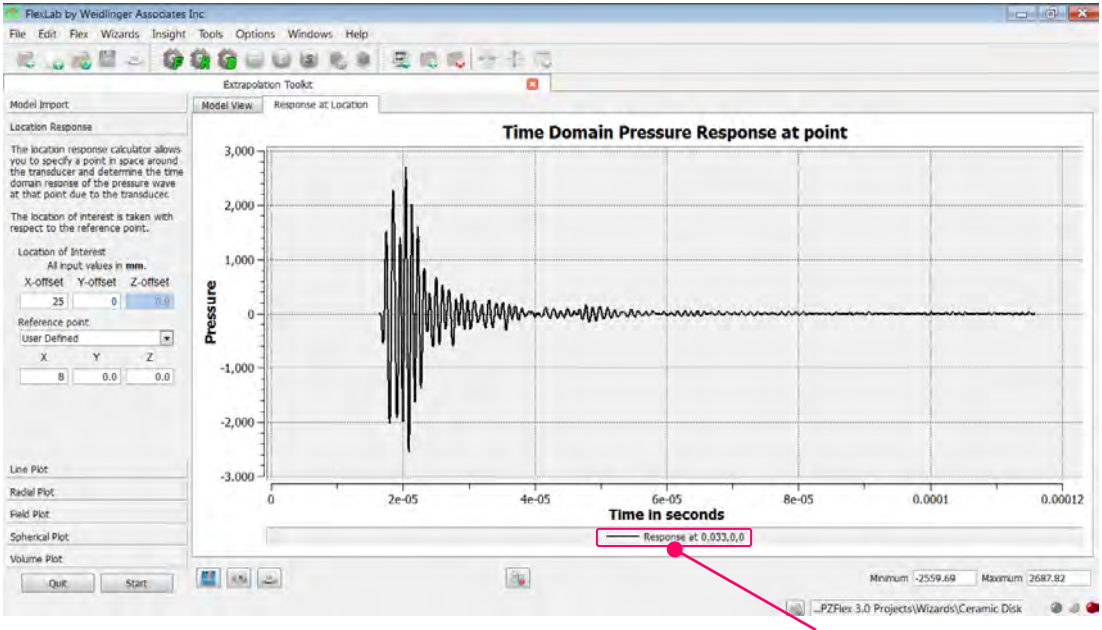
The location response tool gives you the ability to specify a point in 3D space around the transducer to determine the time response of the arriving pressure wave when the device is active.

There are 2 main settings for the location response to function: the reference point and location of interest from the reference point.

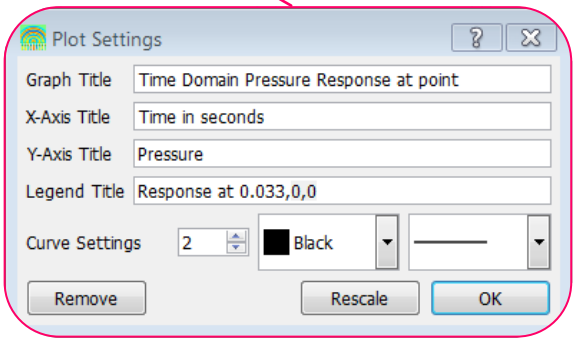
The drop down menu allows you select from a range of predefined reference points and if you would like to choose your own reference point, you can select 'User Defined' and continue to enter the position of your point. The XYZ values for moving the origin are referenced from the Global Origin (0,0,0) of the model: select the 'Global Origin' point from the drop down menu to find out its location.

With the reference point selected, enter your desired location in terms of XYZ distances and you shall be able to see your point of interest appear as a blue dot on the interface. Negative values are permitted for the XYZ distances.

Note: The Z-offset will not be accessible for 2D models



Click the 'Start' button and the response will be calculated. You can zoom in on certain parts of the graph using the left mouse button and reset the view of the graph with the right.

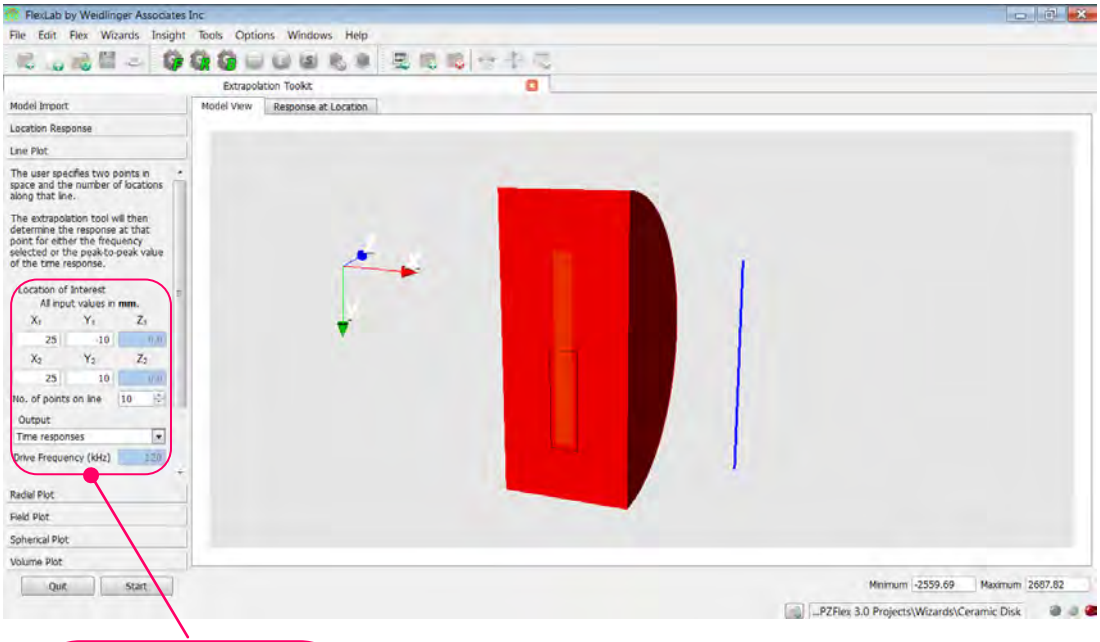


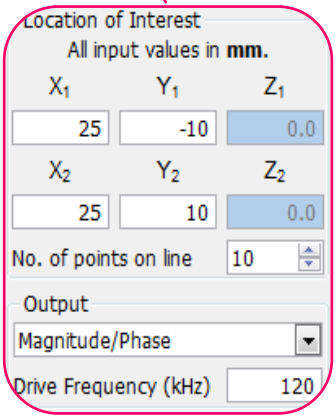
Clicking on the curve response in the graph legend will bring up a window allowing you to alter the typical settings for the curve (line width, colour and line type) and the title for the graph, axes and data set.

The data can be exported as a data file (**.csv**), saved as a jpeg or printed out using the icons below the bottom left-hand corner of the graph.

Line Plot

The line plot works in a similar fashion to the location response except, now your point of interest becomes a 'line' of points.

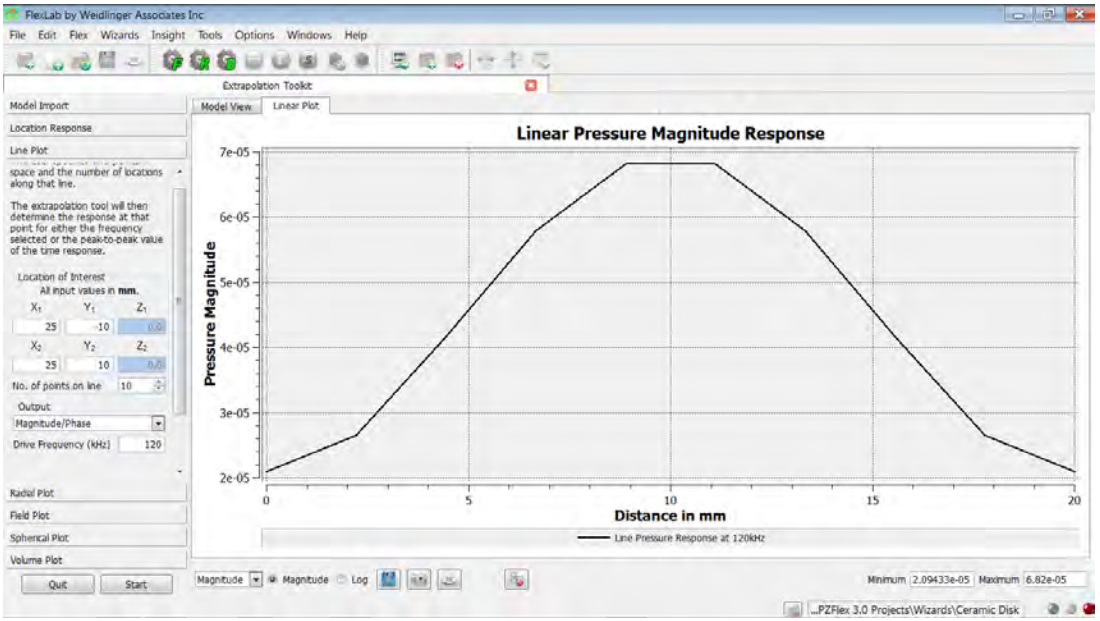


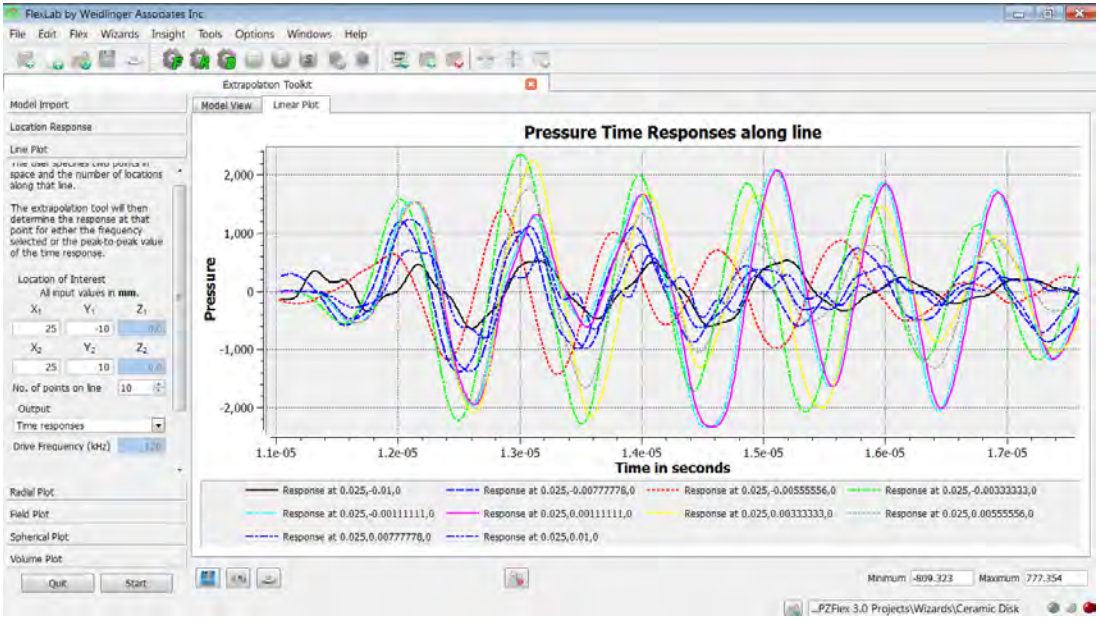


The line plot requires 2 points to start and end the line: enter the XYZ distances for both points and a blue line will appear on the interface. The number of points which the pressure data points to be calculated along the line can also be altered (1000 points max).

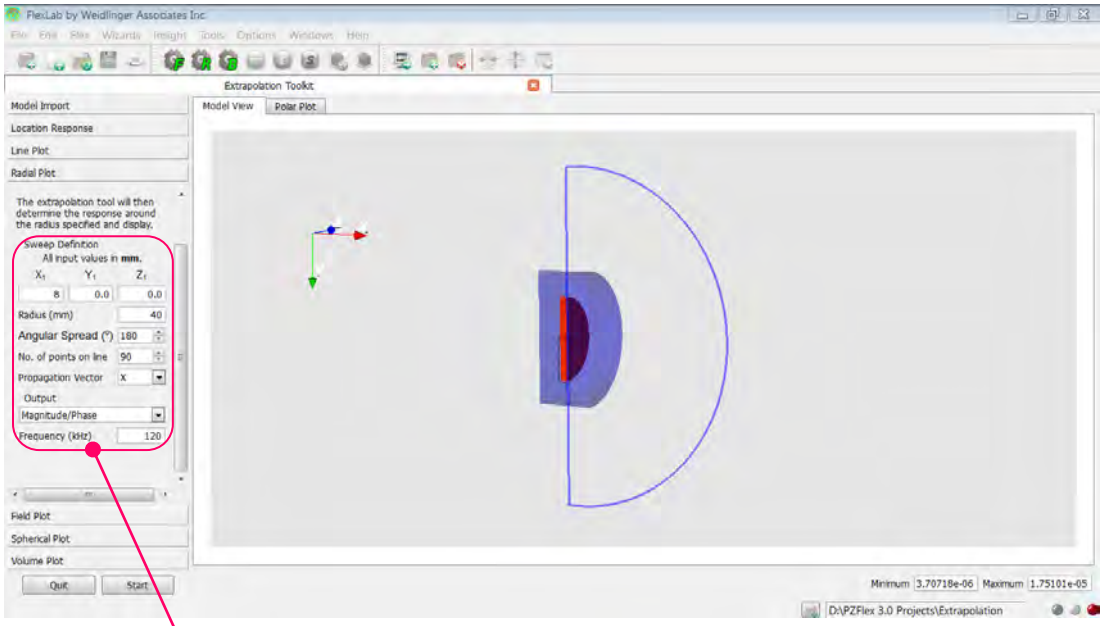
Select the frequency to drive your device and the output type: magnitude of the pressure or the time responses as the wave passes through the points along the line.

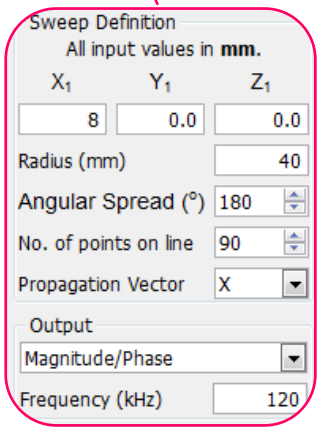
For the time response there will be many curves on the graph so use the curve customisation to distinguish between the data points .


FlexLab by Weidlinger Associates Inc
File Edit Flex Wizards Insight Tools Options Windows Help
Extrapolation Toolkit
Model Import
Location Response
Line Plot
Model View
Linear Plot
Linear Pressure Magnitude Response
Pressure Magnitude
Distance in mm
Line Pressure Response at 120kHz
Location of Interest
All input values in mm.
Output
Magnitude/Phase
Drive Frequency (kHz)
Radial Plot
Field Plot
Spherical Plot
Volume Plot
Quit
Start
Minimum 2.09433e-05
Maximum 6.82e-05
...PZFlex 3.0 Projects\Wizards\Ceramic Disk


FlexLab by Weidlinger Associates Inc
File Edit Flex Wizards Insight Tools Options Windows Help
Extrapolation Toolkit
Model Import
Location Response
Line Plot
Model View
Linear Plot
Pressure Time Responses along line
Pressure
Time in seconds
Location of Interest
All input values in mm.
Output
Time responses
Drive Frequency (kHz)
Radial Plot
Field Plot
Spherical Plot
Volume Plot
Quit
Start
Minimum -809.323
Maximum 777.354
...PZFlex 3.0 Projects\Wizards\Ceramic Disk

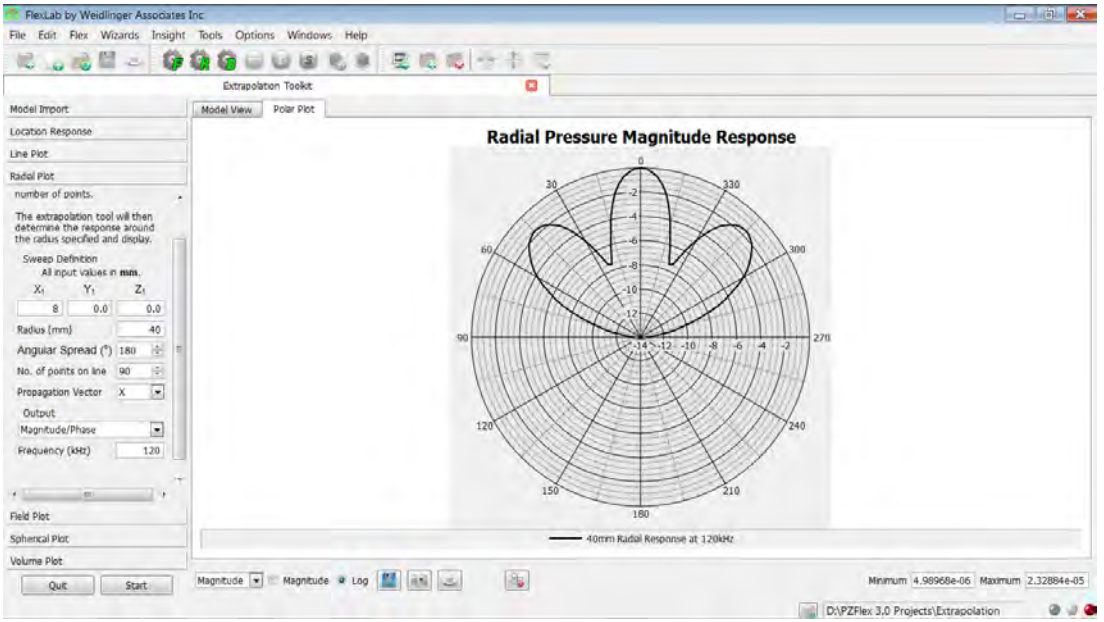
Radial Plot



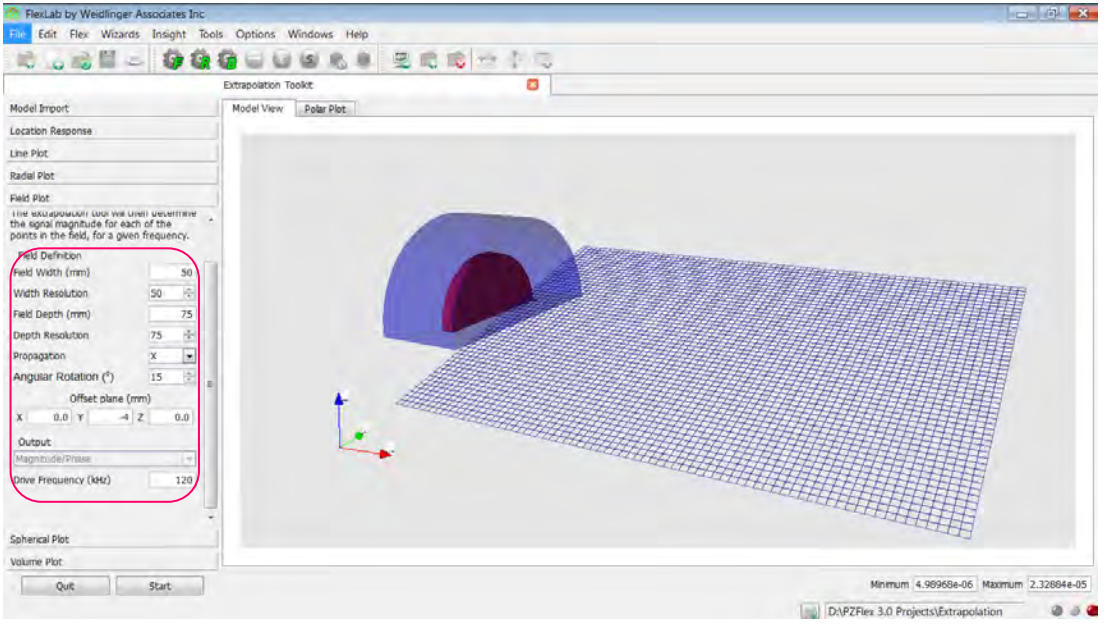


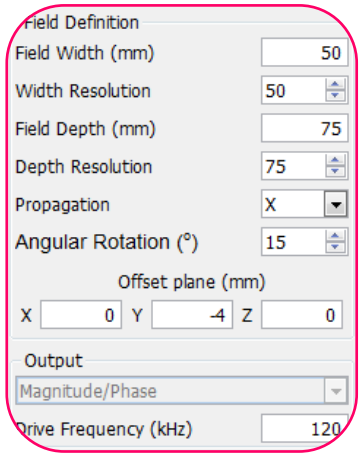
Radial plots can be obtained from your device very simply. Begin by entering the point where the radial plot will begin which will be the front face of the device (8 mm along the x-axis from the Global Origin in the example). The 'Radius' of the radial plot which dictates how far the radial plot will extend in the propagation direction. The 'Angular Spread' is as the name suggests, is the angle which the 'field' is spread for the calculation. The 'Propagation Vector' is to match the plot to the direction at which the pressure wave will be travelling in. Select your drive frequency and radial plot is ready to be calculated.

The radial plot will be shown on screen and give you the opportunity to alter the y-axis to a logarithmic scale before printing, saving the plot as an image or exporting as a data file (**.csv**). Clicking on the data legend will allow changes to be made to graph similar to precious plots in the toolkit.


FlexLab by Weidlinger Associates Inc
File Edit Flex Wizards Insight Tools Options Windows Help
Extrapolation Toolkit
Model Import
Location Response
Line Plot
Radial Plot
number of points.
The extrapolation tool will then determine the response around the radius specified and display.
Sweep Definition
All input values in mm.
X1 Y1 Z1
8 0.0 0.0
Radius (mm) 40
Angular Spread (°) 180
No. of points on line 90
Propagation Vector X
Output
Magnitude/Phase
Frequency (kHz) 120
Field Plot
Spherical Plot
Volume Plot
Quit
Start
Model View
Polar Plot
Radial Pressure Magnitude Response
40mm Radial Response at 120kHz
Magnitude
Log
Minimum 4.98968e-06
Maximum 2.32884e-05
D:\PZFlex 3.0 Projects\Extrapolation

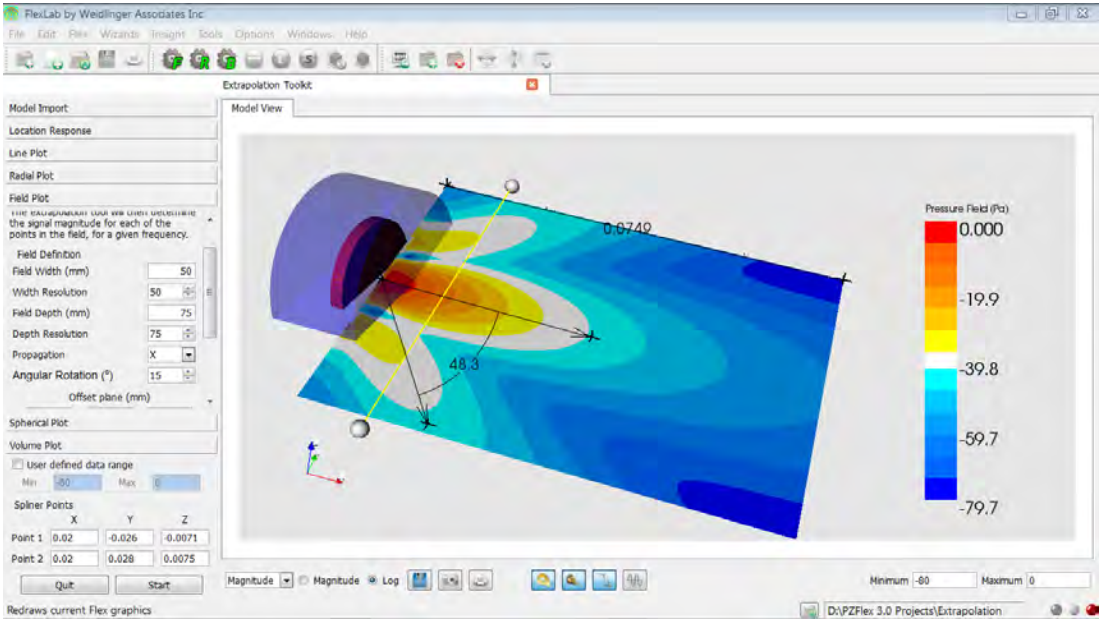
Field Plot

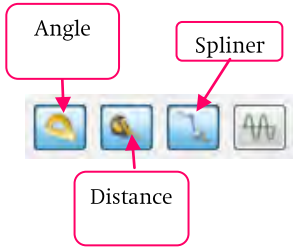

FlexLab by Weidlinger Associates Inc
File Edit Flex Wizards Insight Tools Options Windows Help
Extrapolation Toolkit
Model Import
Location Response
Line Plot
Radial Plot
Field Plot
the signal magnitude for each of the points in the field, for a given frequency.
Field Definition
Field Width (mm) 50
Width Resolution 50
Field Depth (mm) 75
Depth Resolution 75
Propagation X
Angular Rotation (°) 15
Offset plane (mm)
X 0.0 Y -4 Z 0.0
Output
Magnitude/Phase
Drive Frequency (kHz) 120
Spherical Plot
Volume Plot
Quit
Start
Model View
Polar Plot
Minimum 4.98968e-06
Maximum 2.32884e-05
D:\PZFlex 3.0 Projects\Extrapolation



The field plot tool allows you to generate a pressure field anywhere within the 3D space of your model.

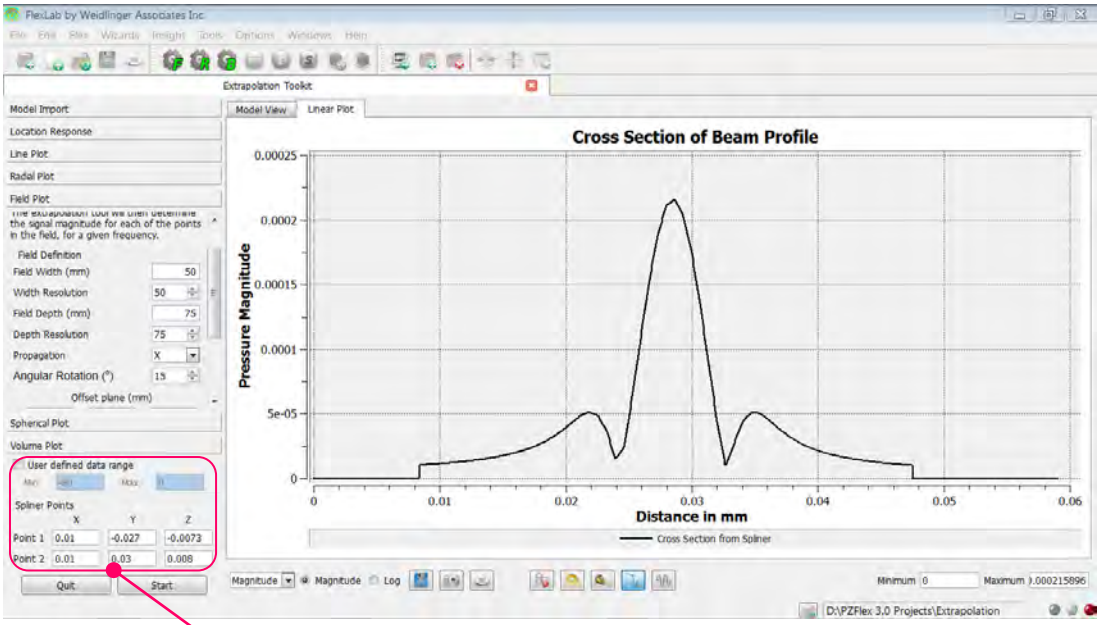
The size of the pressure plot is defined using the 'Field Width' and 'Field Depth' parameters. The 'Resolution' inputs determines the detail of the field plot i.e. how large an 'element' of the field is. The 'Propagation' setting, again, refers to the direction of the traversing wave. The 'Angular Rotation' allows the you rotate the field up to 180 degrees. The 'Offset' parameters shifts the plane in any of the XYZ directions giving you complete control over where the field plot will be extracted. Finally, select the frequency to drive your device and activate the tool.

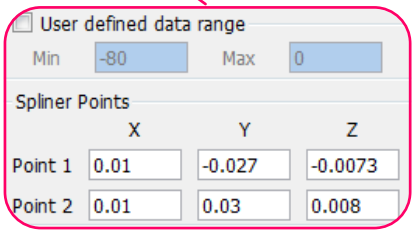




There are post-processing widgets that allow more to be shown on top of the basic pressure field plot. There are 3 widgets available: the angle measurer, the distance measurer and the spliner. The angle and distance measurer are activated by clicking on the field plot.

The spliner tool allows you to extract the pressure data straight from the pressure plot. By placing the spliner anywhere on the pressure field, data across the spline can be exported and displayed as a graph.

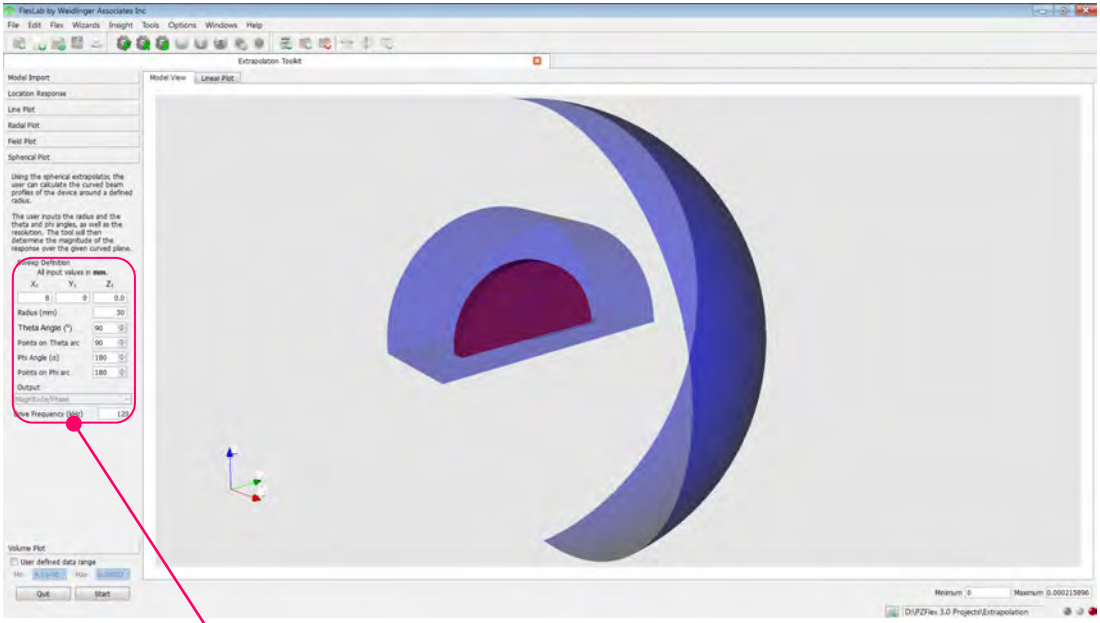


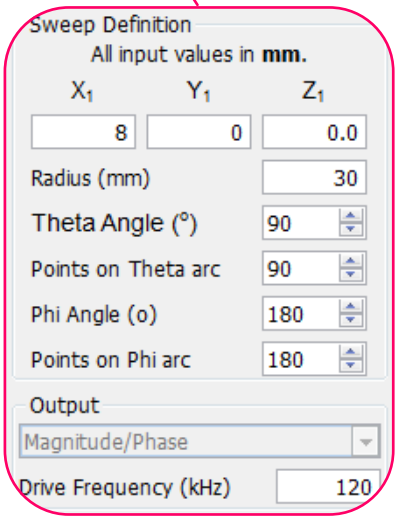


The spliner tool can also be placed on the field plot by using the XYZ input values (m) giving you better precision. Checking the 'User defined data range' box allows you to set the minimum/maximum amplitude for the calculated field plot.

Spherical Plot

The spherical plot generates a 'shell' of pressure data around your device.



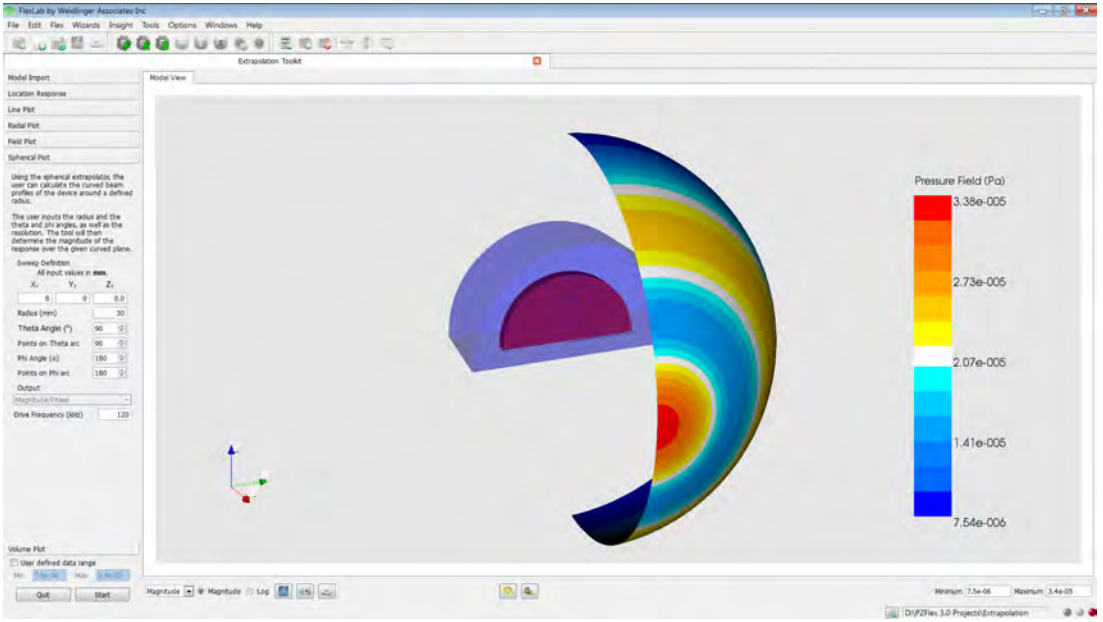


The XYZ parameters shifts the centre point of the sphere within the 3D space of the model (referenced from the Global Origin). The 'Radius' controls the spherical radius. There are 2 angles which sets the segment of the sphere's shell which pressure data will be calculated for:

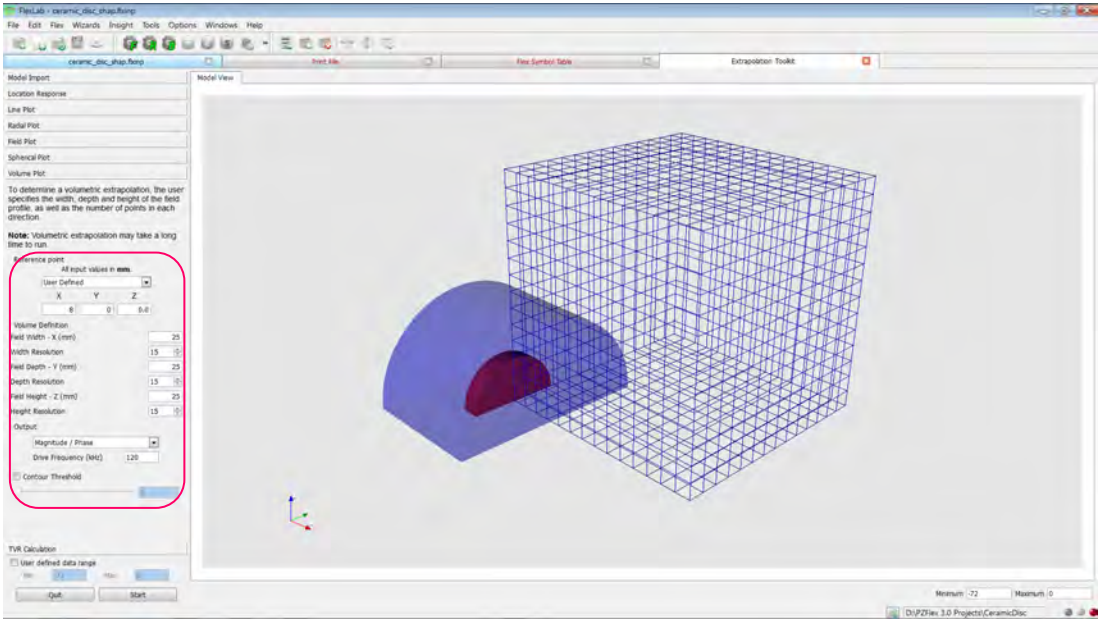
- 'Theta' dictates the angle progressed around the Z-axis (range — 0 to 360)
- 'Phi' controls the angle progressed around the X-axis (range — 0 to 180)

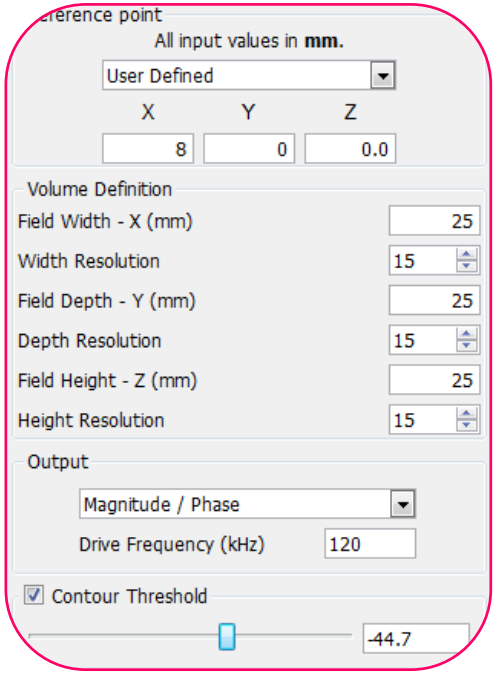
'Points on Arc' are the resolution settings for the corresponding angle directions to generate the grid system around the shell of the sphere. The drive frequency can be set at the bottom and the spherical plot can be generated.

All the general tools are available along with the angle and distance measuring widgets.


Pressure Field (Pa)
3.38e-005
2.73e-005
2.07e-005
1.41e-005
7.54e-006

Volume Plot

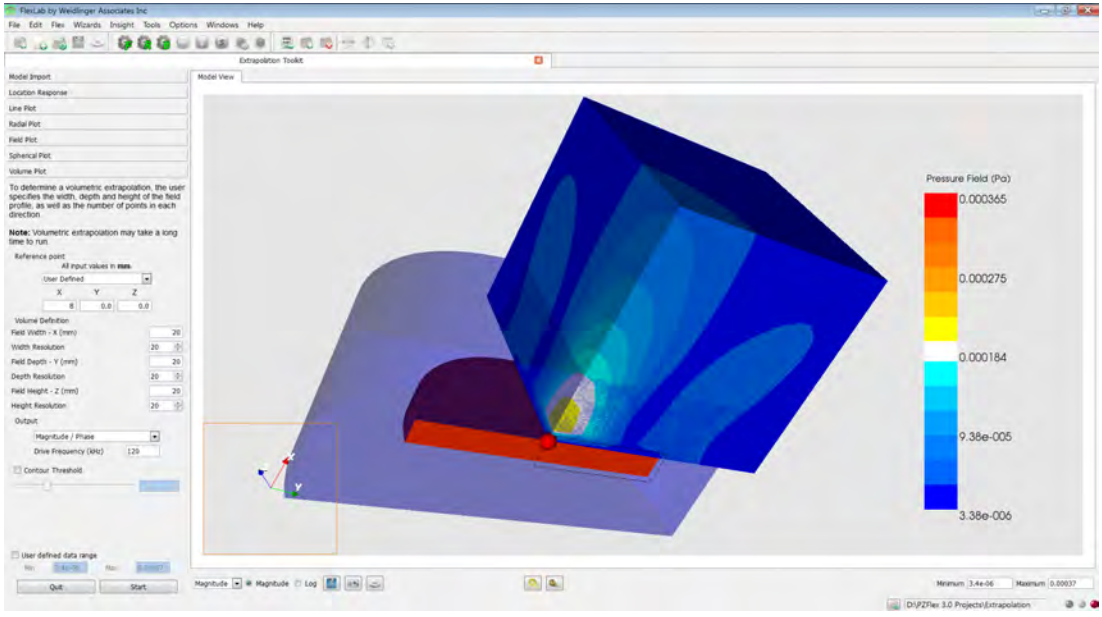


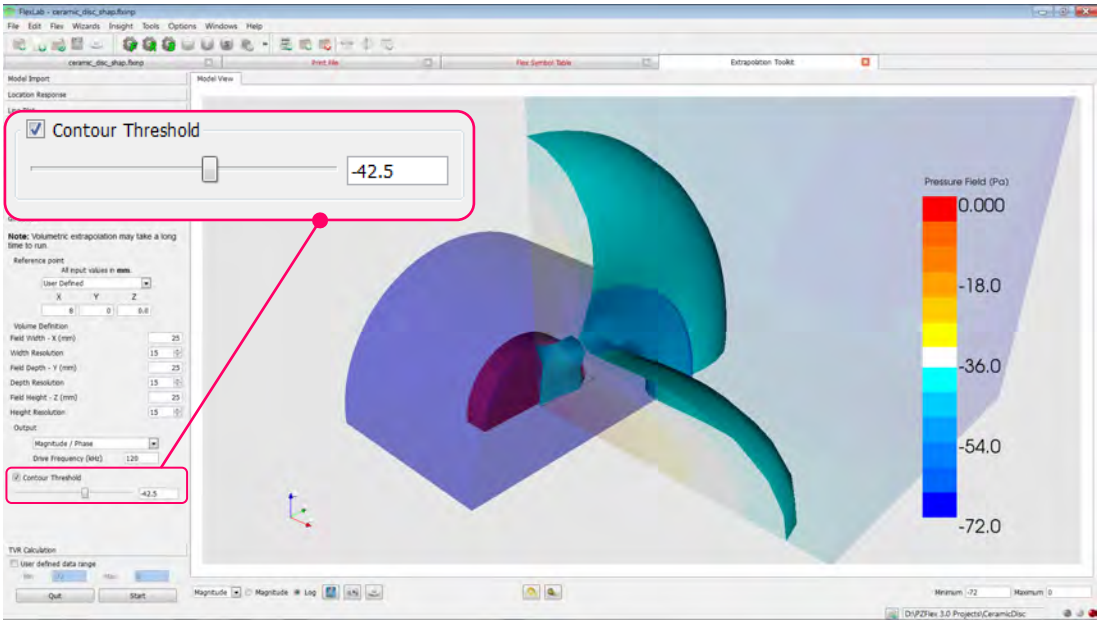


The settings for creating a volume plot is much like the others. A range of reference points can be selected from the drop down menu or you can define your own using XYZ input parameters. Using the 'Width', 'Depth' and 'Height' settings, the volume of the 3D grid can generated. The 'Resolution' inputs, again, governs the size of the 'elements' for the 3D grid. With the driving frequency for the device set the volume plot can calculated.

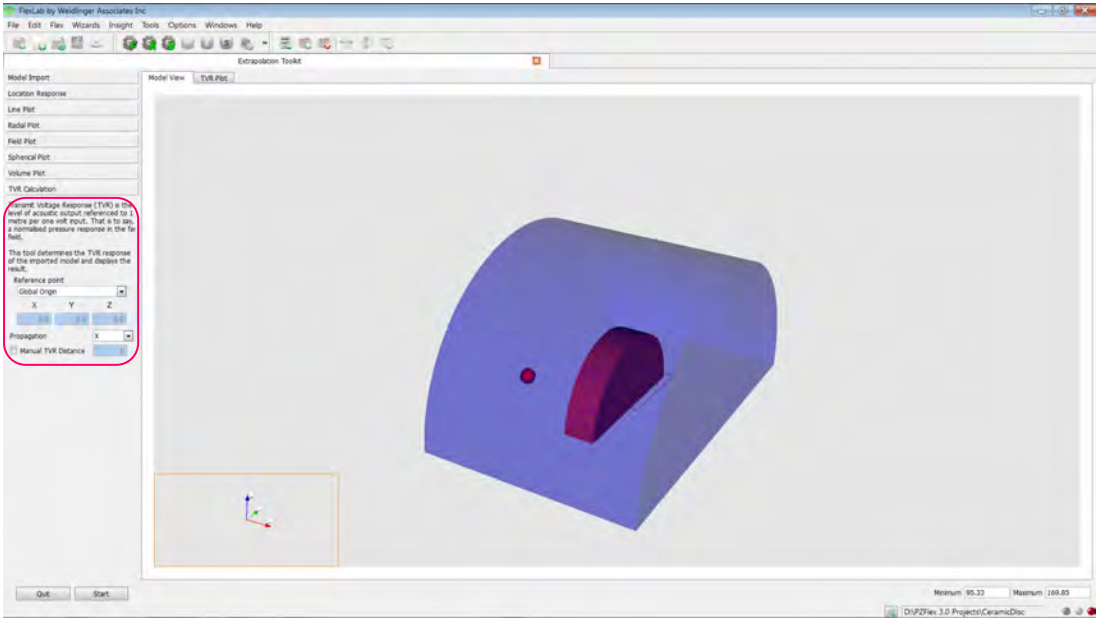
The 'Contour Threshold' option allows regions of the same pressure value to be highlighted, to show the shape of the pressure distribution.

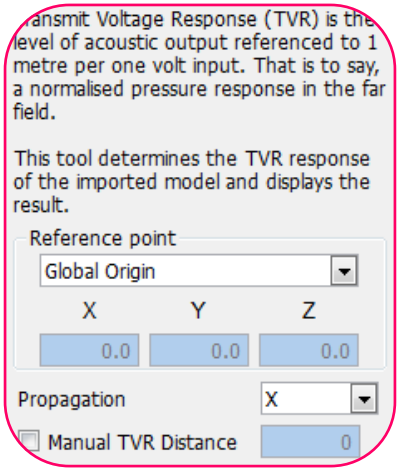
The general tools found here are the same as the Spherical Plot section.





TVR Calculation

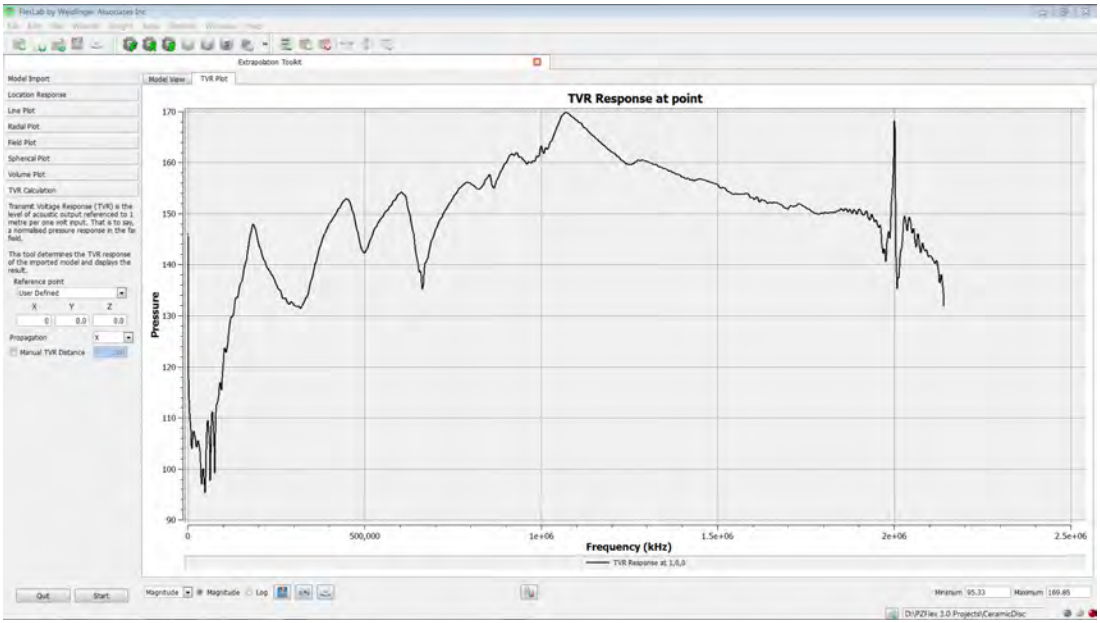




The TVR calculation can be performed in this tab. Note that the DRIV subcommand must be used within the flex input file in order to be able to access the 'TVR Calculation' tab.

The propagation vector can be altered to suit the direction of propagation of your model.

Due to the variable nature of the far-field region, the 'Manual TVR Distance' checkbox allows you to set a custom distance suited for your model.



Settings — Extrapolation Settings

Within the main PZFlex settings there are options that governs its basic functionality:

- Showing model geometry with symmetry flipped out
- Changing colour tables for graphs and model geometry units
- Option to clear linear plots and polar plots after each simulation run

Insight

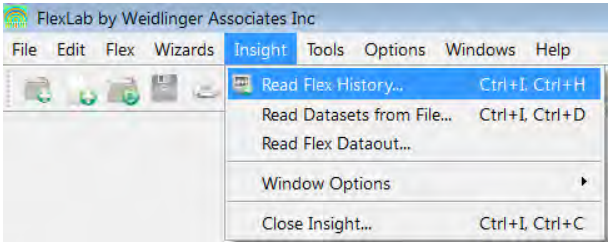
Insight allows you to view your time histories and data out arrays after the execution of your model.

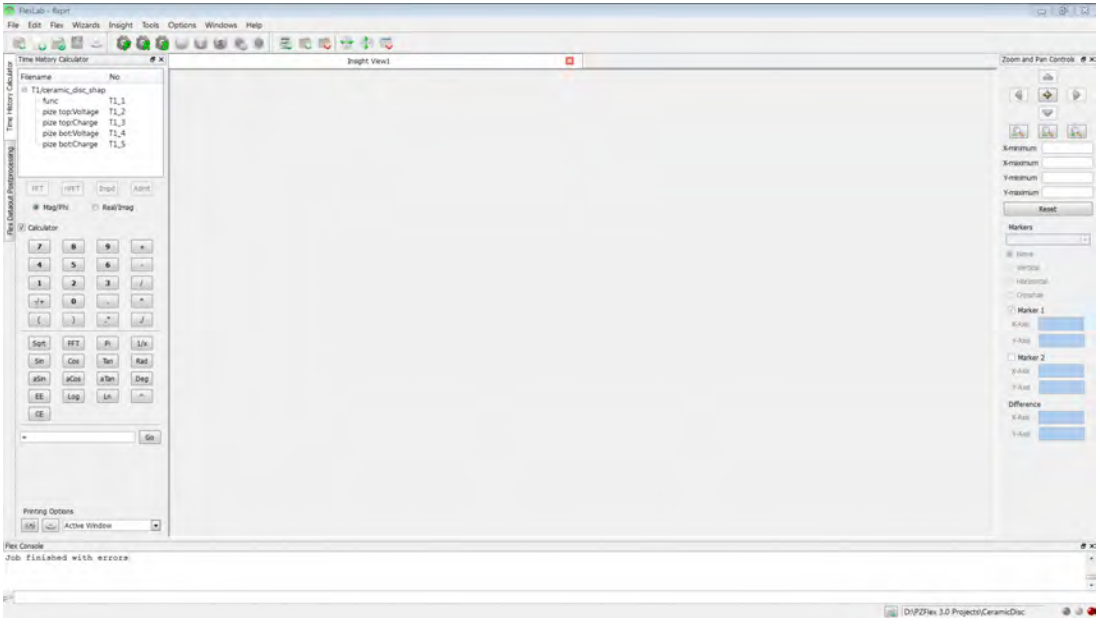
Flex History

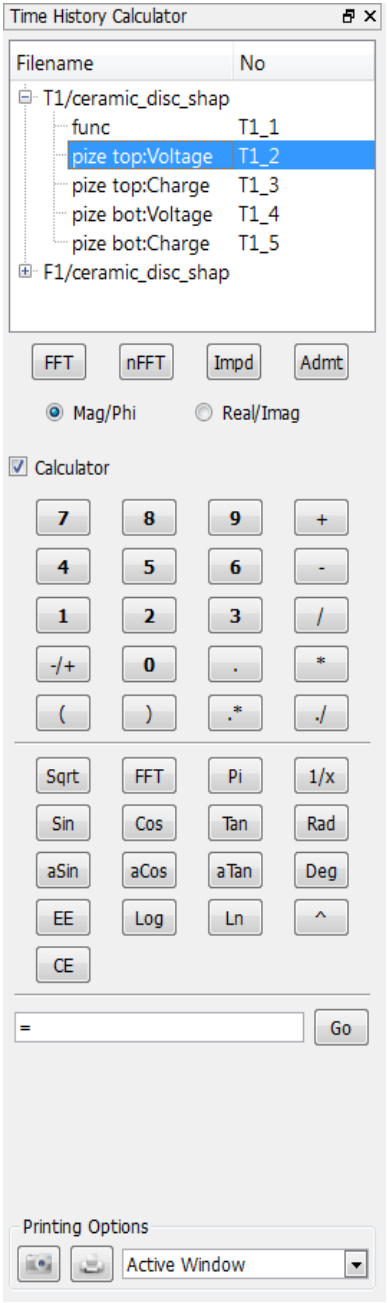
Insight allows the data from the save time histories to viewed and analysed. Data such as impedance and admittance profiles can be generated as well performing Fast-Fourier transforms on the saved signals. Insight also provides you with the basic tools to alter the axis scales, data lines and the ability to output the data as Jpegs.

Using Flex History

To use to the Insight Flex History Reader, it can be selected from the Insight tab at the top of PZFlex interface:







On the top of the Insight interface, we have dedicated icons to:

1. Open flex history files
2. Open new Insight windows for plotting
3. Close Insight windows
4. Split an Insight window vertically
5. Split an Insight window horizontally
6. Remove the current active window

An active window is found within an Insight window: an Insight window is split into active windows to allow a number graphs to be plotted and compared.

On the left side of the Insight interface we have the browser for the saved time histories from the model. Double-clicking on them will plot them in the selected active window. Below the browser, there are 4 buttons:

1. FFT: perform Fast Fourier Transforms
2. nFFT: normalised Fast Fourier Transforms—Normalises your performance data to the drive function
3. Impd: generates Impedance profile
4. Admt: generates Admittance profile

For the Impedance/Admittance profiles you can select to display in Magnitude/Phase or Real/Imaginary terms.

Checking the Calculator box will open up basic calculator functions to apply to your data sets.

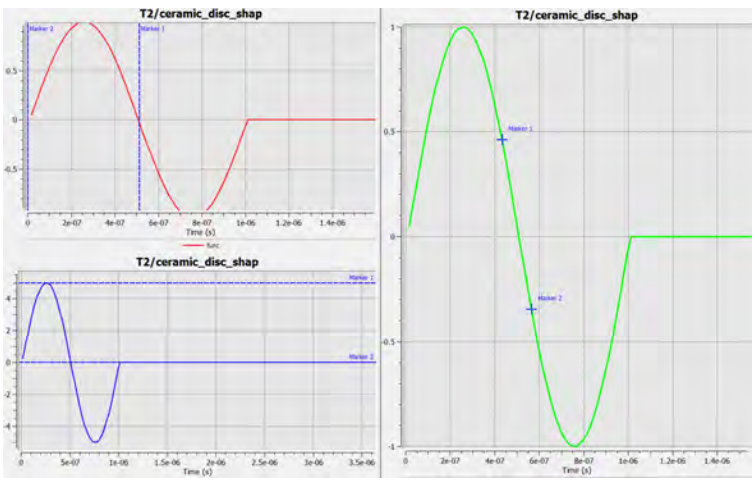
At the bottom corner, the drop down menu allows you to select the windows to save as images or print them out.

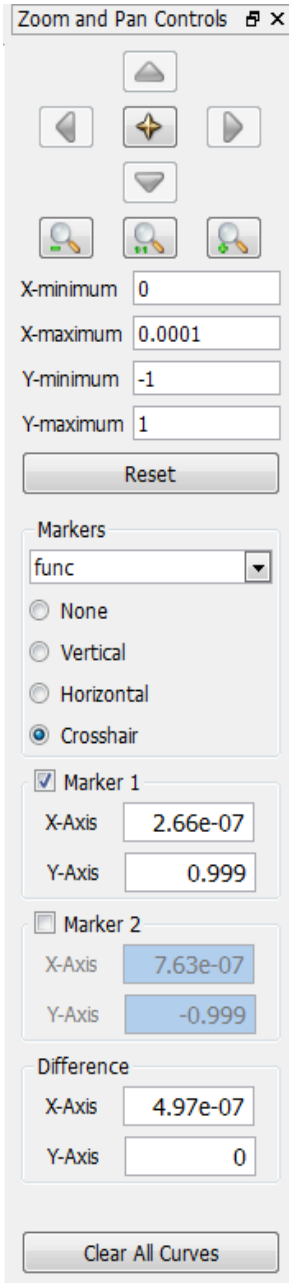
The functions on the right side of the Insight interface will be familiar from the Wizards and Extrapolation toolkits.

The centre ‘star’ icon toggles between the zooming and panning functionality. When zoom is selected, the 3 magnifying icons can be used to zoom out, reset the view and zoom in respectively. When panning is selected, the directional icons become available to navigate around the graphs in the active window (s). The mouse can also be used to select an area to zoom (left mouse button) and reset the view (right mouse button).

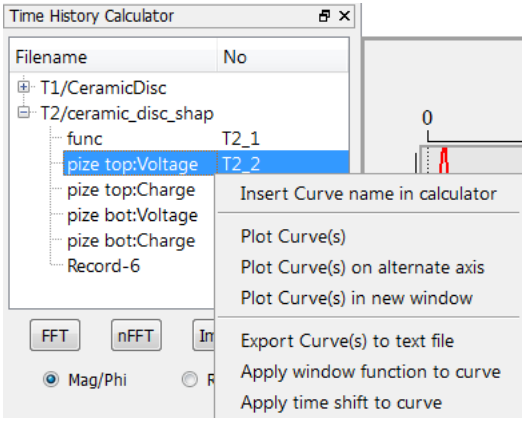
The XY-minimum/maximum displays the min/max values of each axes of the graph. They can be changed to allow you to zoom in on an area of the graph or to simply resize the graph. The values can be ‘Reset’ using the button.

A variety of markers for identifying points on the graph can be chosen. The down menu lets you choose the data set to use the markers for; the more graphs you plot, they will appear in the menu for selection. The different marker types all work in the same fashion and will depend on what data you would like it to identify more clearly from each axis. The marker’s data readings will be shown in their corresponding sections. Note that the markers snaps to the data curve so you do not need to worry about accurately selecting a point on the curve. Finally, the ‘Clear All Curves’ button at the bottom can be used to remove all graphs from an active window, making room for a new plot.

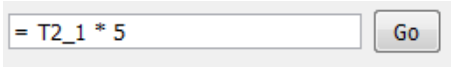




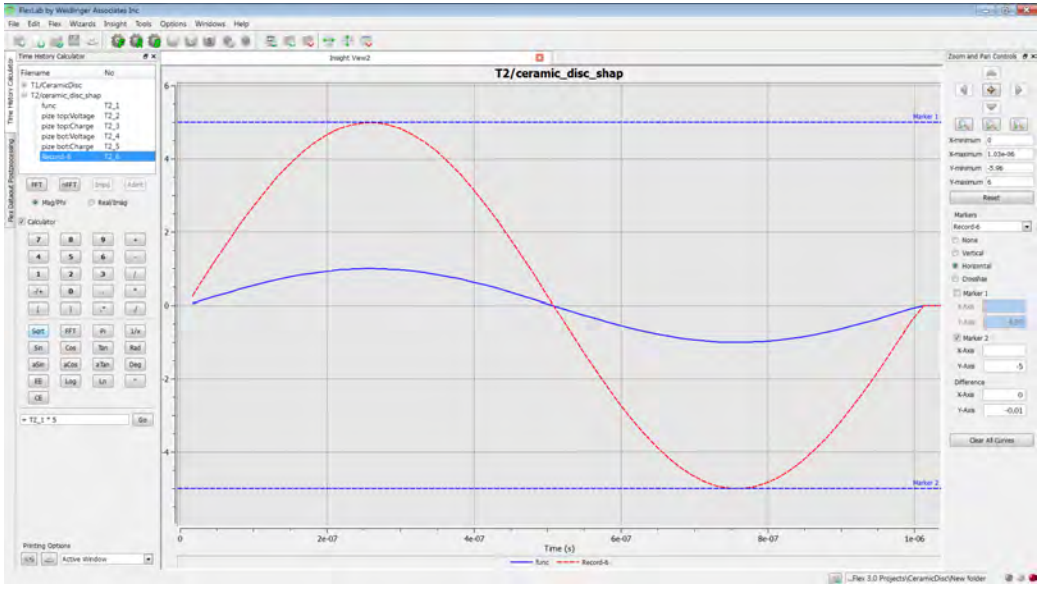
In the time history browser, if you right click on any time history, the pop up menu offers you a variety of options to manipulate your data:

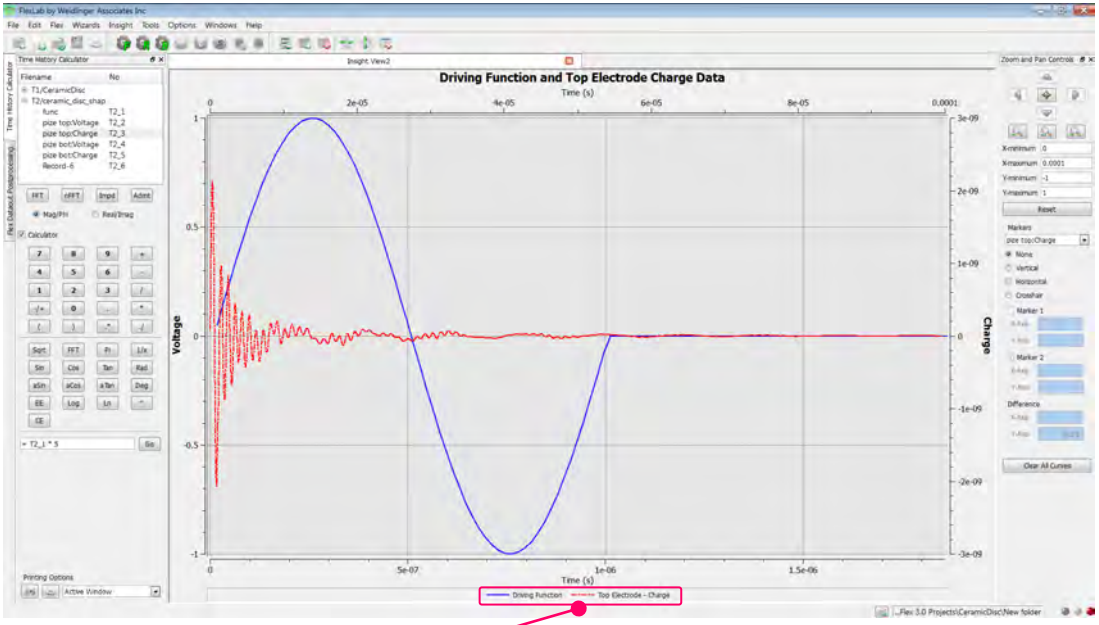


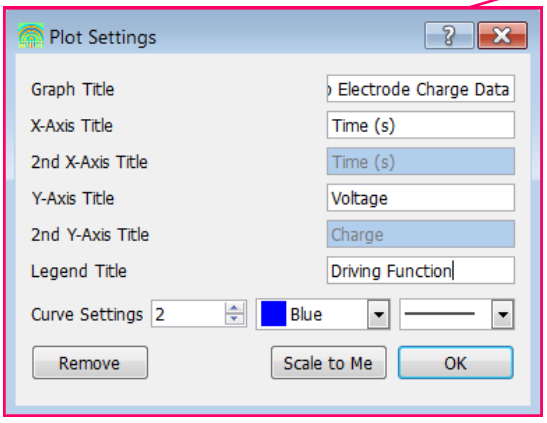
The first options is used in conjunction with the calculator: the time history name is entered into the calculator input box and will allow you to perform mathematical operations on the data for instance multiplying by an arbitrary number:



Clicking 'Go' will generate a new time history (Record-X) in the browser, allowing you to plot the data. It can be seen below that the function has increased in amplitude by 5 fold.







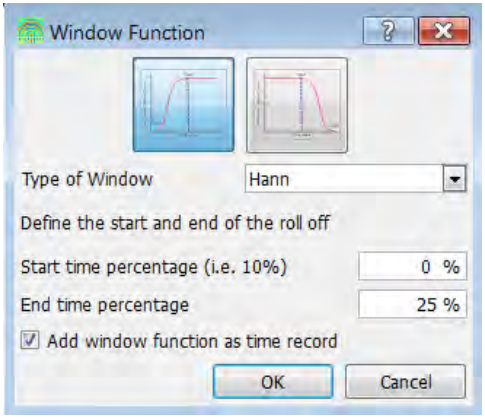
The 'Plot Curve (s)' command is self explanatory. 'Plot Curve (s) on alternate axis' allows a different data type to be plotted in the same active window with another set of axes.

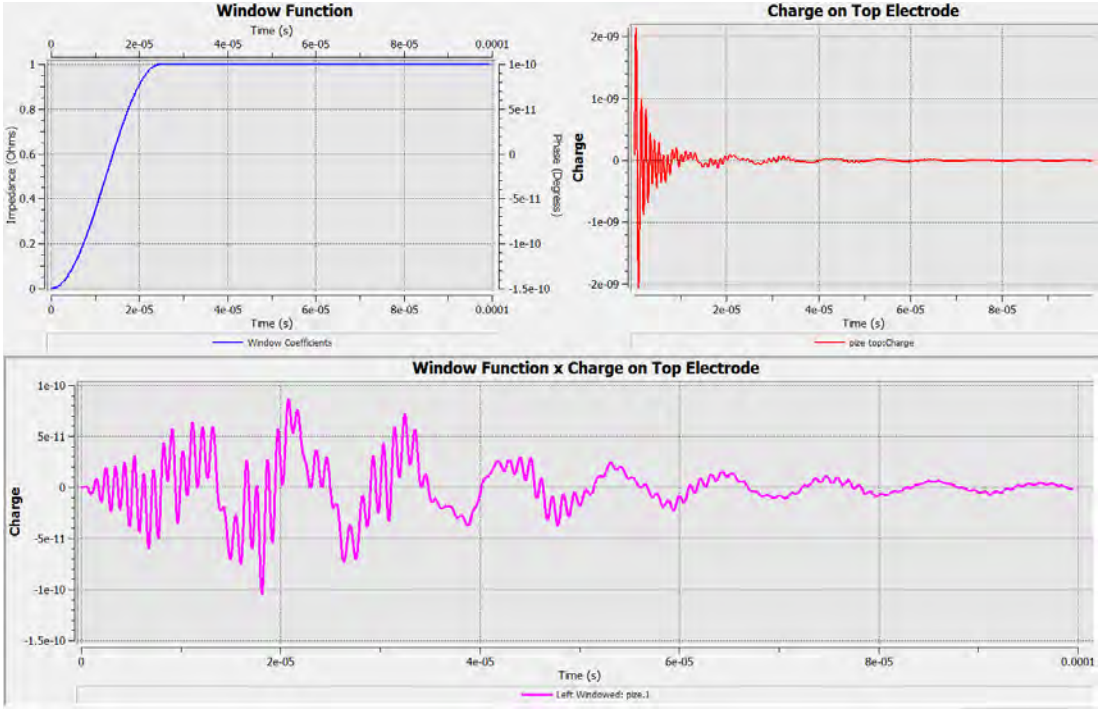
Right clicking the data legends will open a window allowing you to alter curve attributes (line weight, colour and line type), rename the axes for the curve selected and the main title for the graph. Select the other curve in the legend to change the alternate axes names.

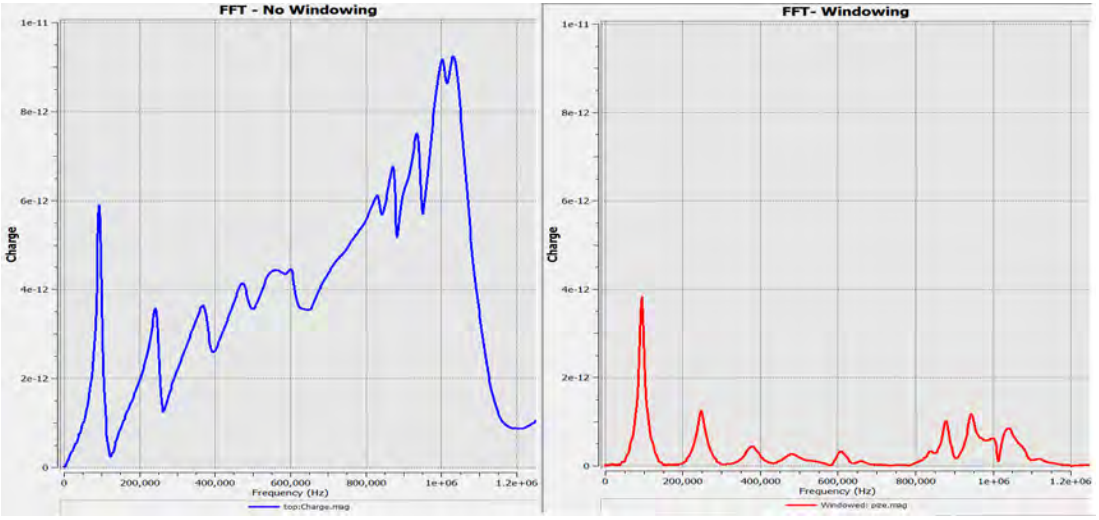
The 'Export Curve(s) to text file' option does exactly that, all that is require it to name the text file and choose the save location. The text file will contain the time data in one column and the amplitude values in the next.

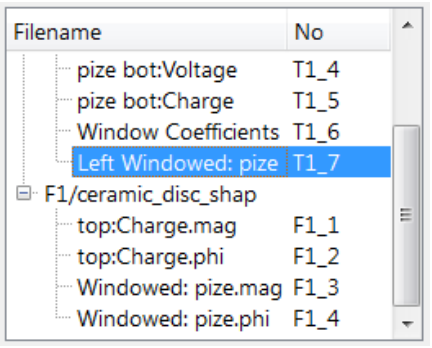
Window functions can also be applied to the time histories. Windowing helps to gain cleaner Fast Fourier Transforms as it can reduce the high frequency noise components in the signal and minimise spectral leakage.

Selecting 'Apply window function to curve' will open up a new window. The Hann window is the first window function displayed and can be applied to your curve very easily. You can select between left-side or right-side Hann window using the 2 clickable icons. The 'Start time percentage' and 'End time percentage' refers to the percentage of the overall runtime of the signal i.e. 50% will be halfway through the signal., therefore, you can start at any percentage as long as it is lower than the end time percentage. Checking the 'Add window function as time record' box saves your window function as a time history which can be plotted in an active window to check if it has been correctly configured.





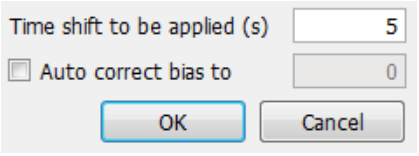




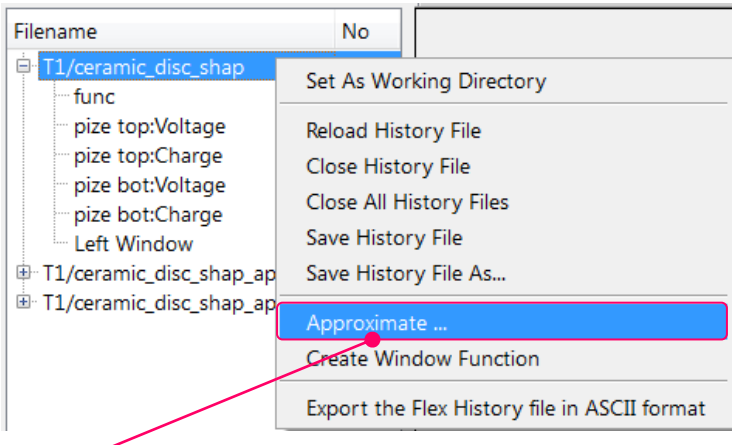
The window function and the new curve will appear in the time history browser, allowing you perform an FFT to the new signal and compare the difference with the original FFT.

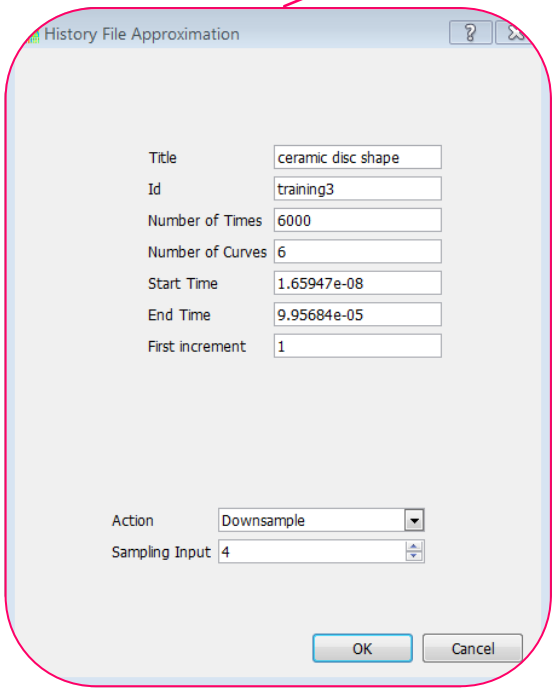
The Hann window is one of many window functions that can be applied to your signal. The window functions include:

- Hamming
- Blackman
- Blackman-Harris
- Bartlett
- Cosine



Time shifts can be applied simply by entering the desire valued. You are also given an option to auto-correct bias to a specific time value. Clicking ‘OK’ will generate a new time-history in the browser for you view.





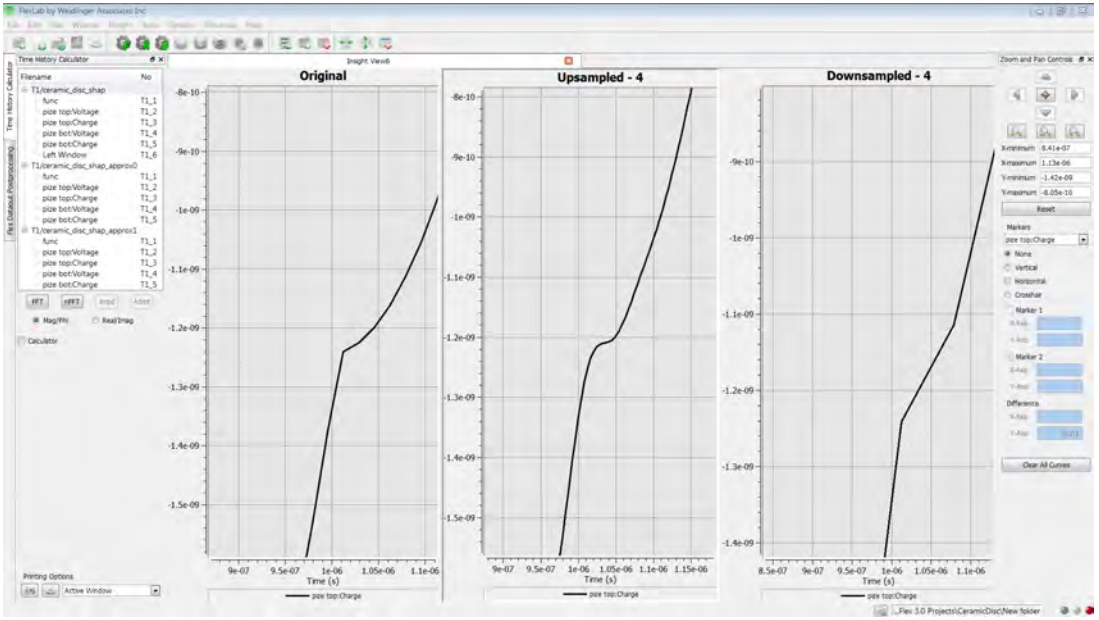
The right clicking on the main time history header offers slightly different options to the individual time histories. First of all you can 'Reload', 'Close' and 'Save' the time history files you have altered. There is also an 'Approximate' function which allows full time histories to be padded or subsampled.

From the drop down menu you can select to:

- Downsample
- Smooth and upsample
- Spline
- Smooth

The number of samples can then be controlled by the 'Sampling Input' (100000 maximum).

Your 'Approximated' data will then appear in the time history browser.



The approximated data plotted side by side with the original shows the effects of the padding and subsampling. Remember to save your new approximated time histories before closing the flex history files.

Settings — Insight Settings

Within the main PZFlex settings there are options that governs its basic functionality:

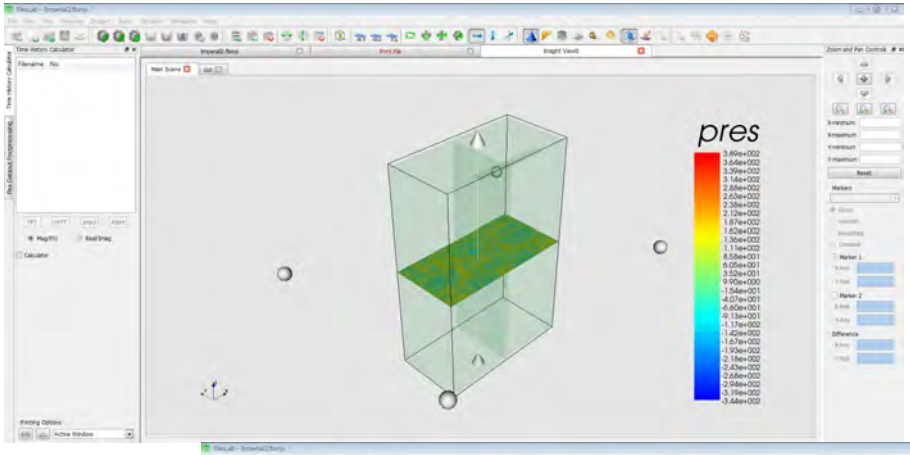
- Setting default units used in plots (frequency and time)
- Significant figures for numbers and (amount of time padding used for FFTs
- Default background colours for plots (grey (default), white or black)
- Font options for plotting (main title, legend title, axis titles & axis scales)
- Impedance plot options—plotting magnitude and phase together and the ability to apply a Hann window to data before calculating impedance/admittance profile. The Hann window runs from 90% to 100% of the time window to remove any offset at the end of the time trace

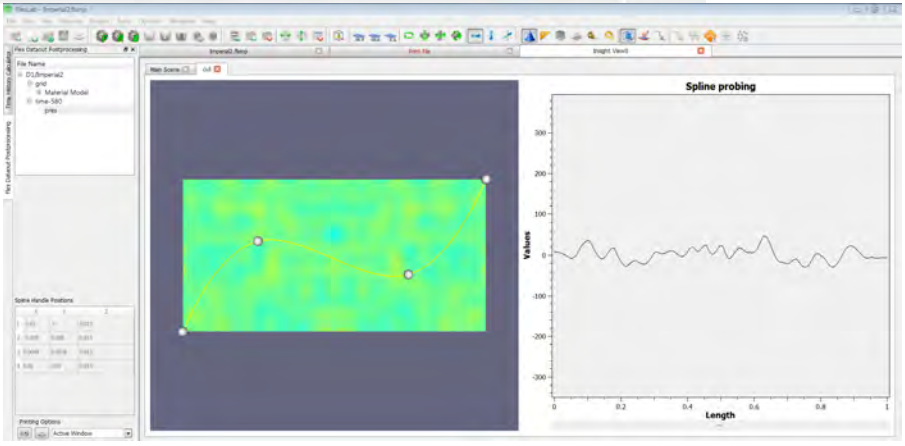
Flex Dataout Reader

The dataout reader allows you to view the **.flxdato** file containing the different data arrays calculated during the execution of the model.

For 3D models, you can intuitively examine your model and extract specific pressure data out. The capabilities in the dataout reader include:

- Full visualisation of 3D model with symmetry replication
- Measuring widgets
- Spline and cut tool — allowing 2D pressure plot to obtained at any plane within the 3D geometry of the model
- Spline probes — obtain graphs of pressure values along the flexible probe
- Plot mode shapes and generate mode shape movies



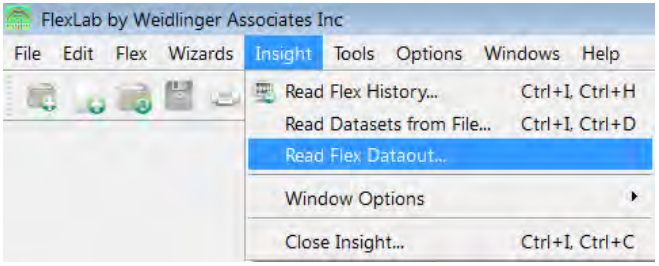


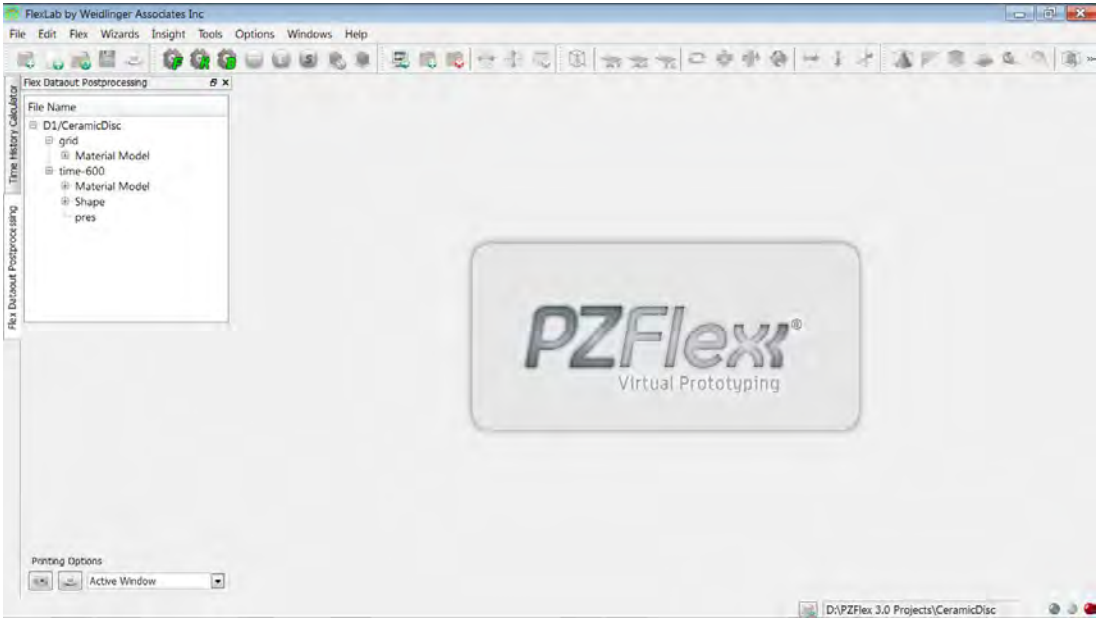
Using Flex Dataout

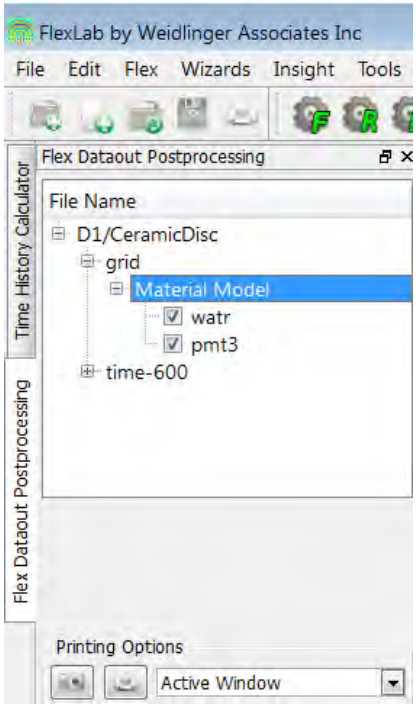
To use Flex Dataout, you need to output the necessary data arrays from your flex model file. After the model executes, the calculated parameters will all be stored in arrays ready to be outputted using the following command:

```
data
    out pres
    out shap/all
    end
```

The code will generate a flex datout file (**.flxdato**) that can be loaded into the flex dataout post-processing interface.





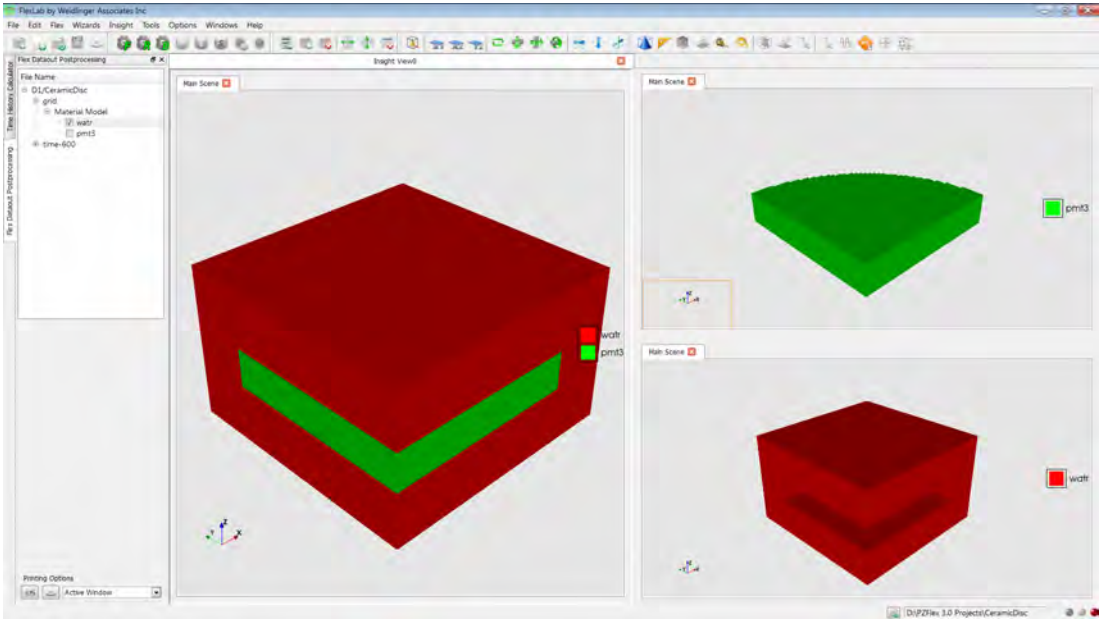


The layout of the flex dataout interface is very similar to the flex history interface.

On the left we have the browser to access all the saved data arrays. The printing options at the bottom offers the usual functionality of capturing images and printing off the active windows.

At the top of the interface, we find the same icons from the flex history interface such as splitting, closing active windows and opening new insight windows. However, there is a much more extensive icon bar than found in flex history interface. Flex dataout deals with data that can be represented in a visual form and so there are many more interactive tools available.

Starting off, you can begin to select some of the data arrays using the data browser. The material model is standard and should always be obtainable from the extracted data arrays: double-clicking on the 'Material Model' should generate the model in a new active window.

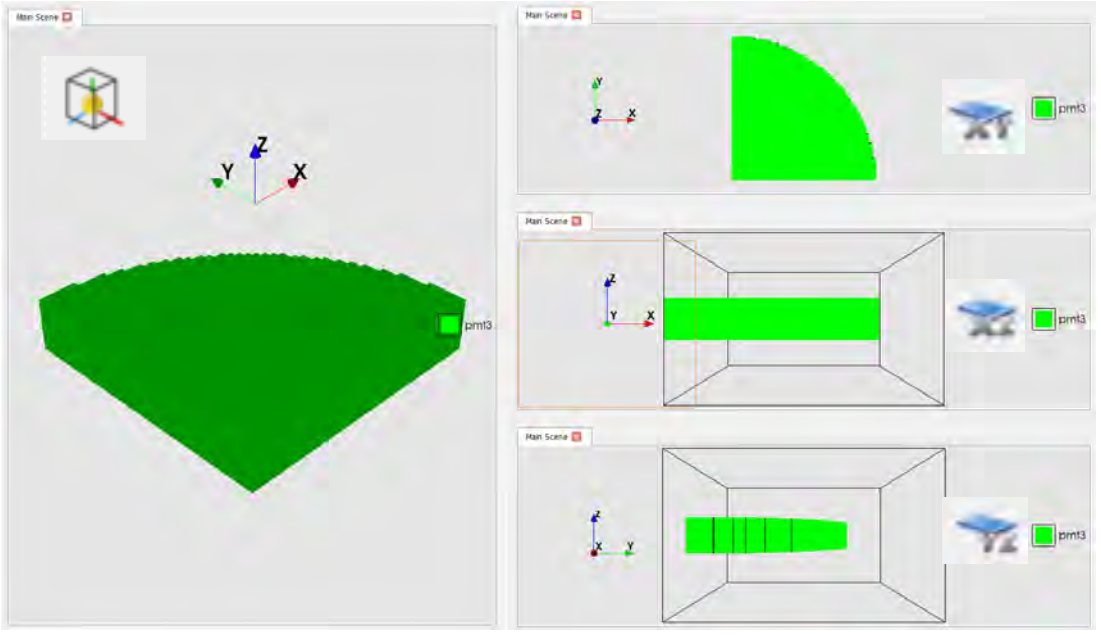


For each active window, you can customise each plotted material model to outline the general construction of the 3D model. The checkboxes next to the materials in the browser can be used to specify which materials to plot and omit.

Icon Bar

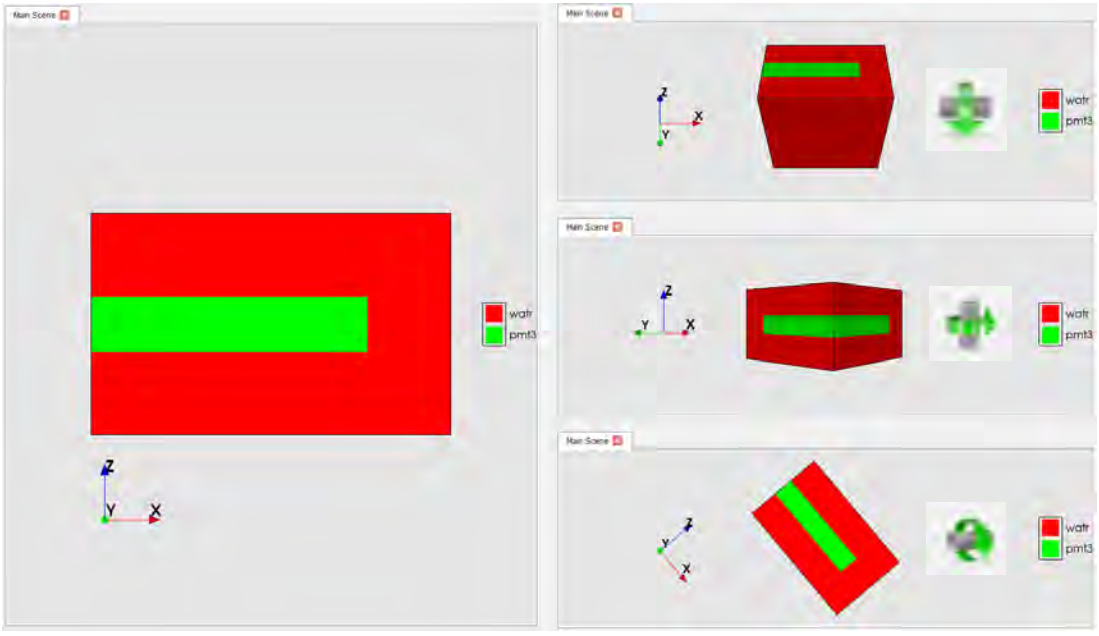
Model Views

As mentioned before there are new icons to be aware of that will help with the 3D visualisation of the extracted data.

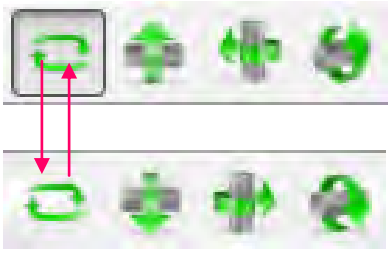


The 4 icons above control the viewpoints of the 3D model. Starting from the left, the first icon sets the viewpoint so you gain an isometric view of the model. The 3 respective icons sets the viewpoint to view a specific plane of the model: XY, XZ or the YZ plane.

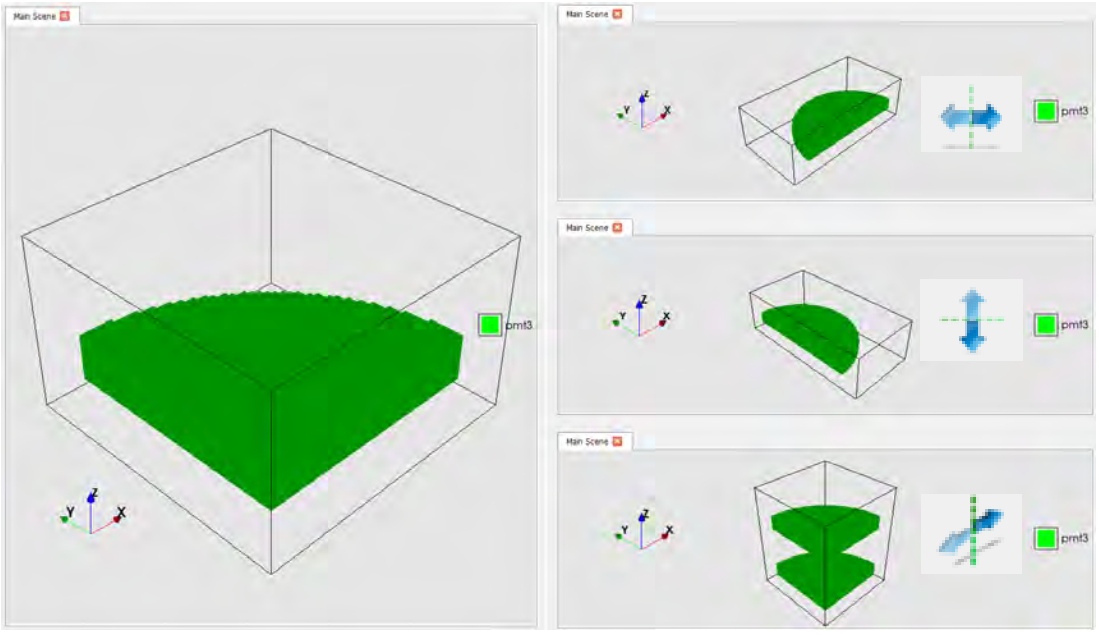
Model Rotation



The 4 icons control the rotation of 3D model in order for you precisely navigate around your model. The 3 icons, starting from the right, control rotation around each axis leaving the last icon which reverses the direction of rotation.

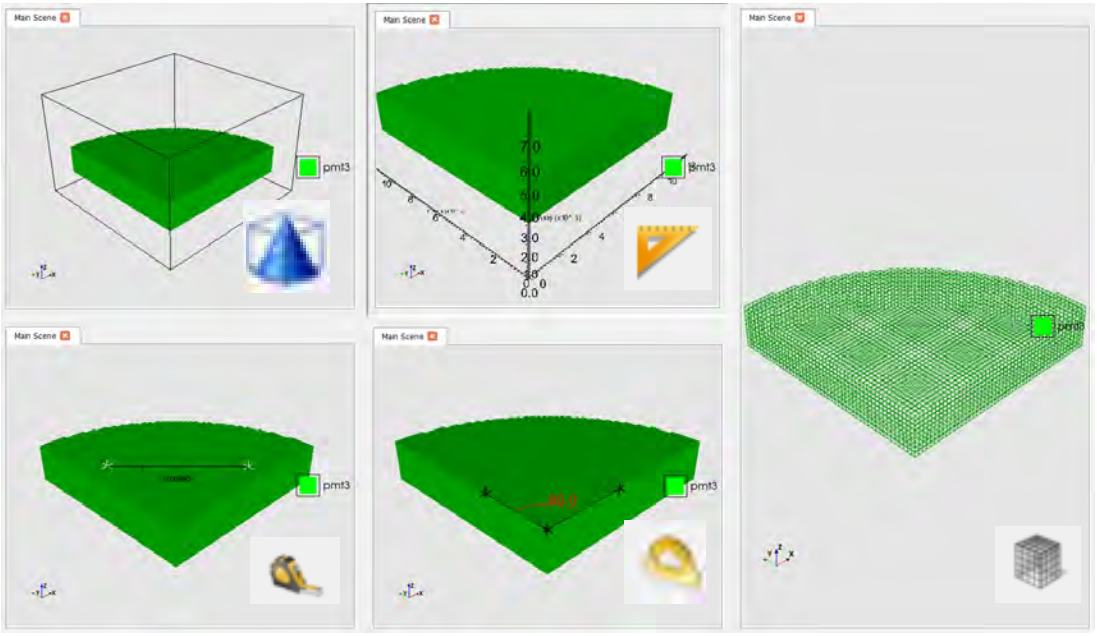


Model Symmetry



The 3 icons control symmetry conditions about each axis. As 3D models are highly excessive to model, they are usually carried out with symmetry conditions. Therefore you gain a representation of the full model through applying the symmetry conditions feature.

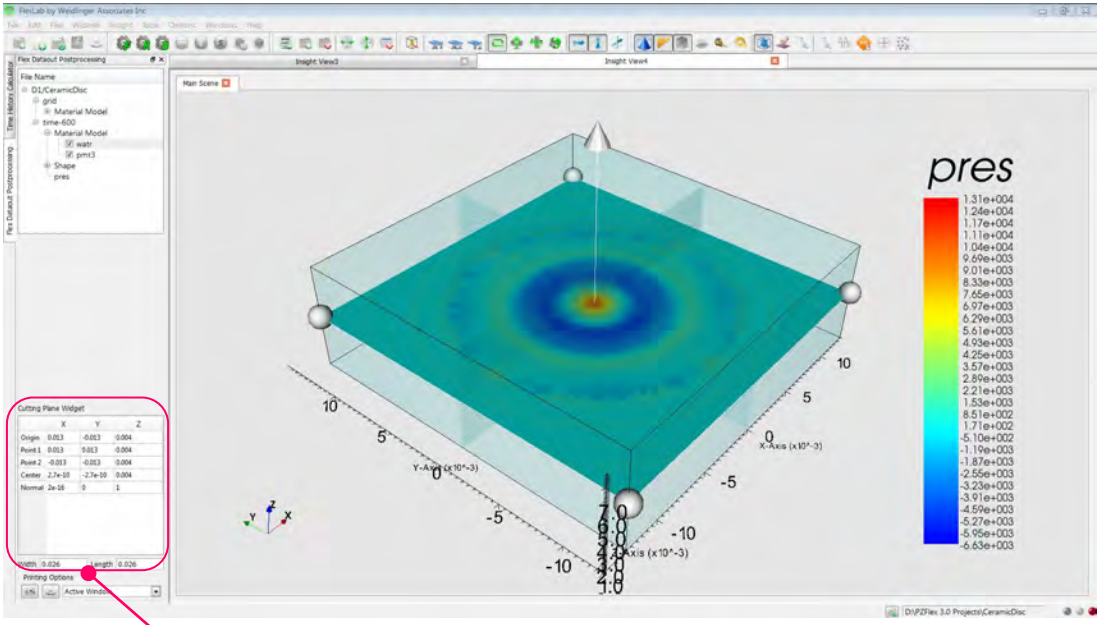
Data Analysis Tools & Widgets



Starting from the left:

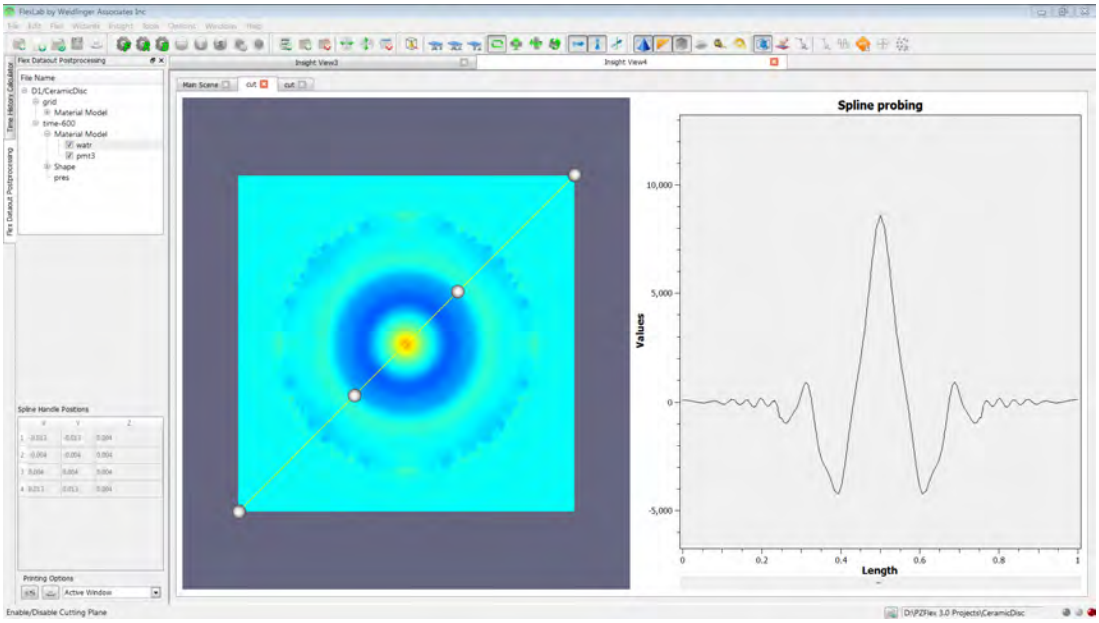
- Data Bounding Box—shows the boundaries of the model
- Show Axis Geometry—shows physical dimensions along each axis
- Show Elements—displays model as a mesh of elements
- Show Distance Widget—measures distance between 2 points on model
- Show Angle Widget—measures angle between two conjoined lines

The icon not demonstrated above is '2D Contour Plotting' which applies to 2D pressure plots. The function converts plain 2D pressure field plots into contour plots, taking advantage of the 3D graphics i.e. areas of higher pressure will protrude out of the 3rd axis (see *Using Isosurface and Contours—P107*).

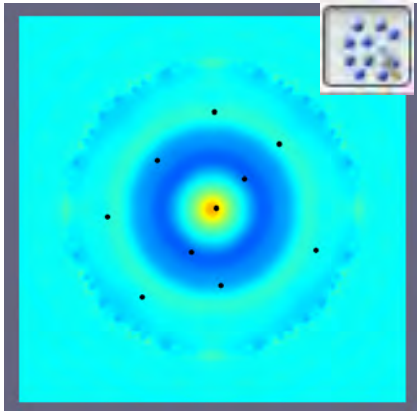


‘Enable/Disable Cutting Plane’ is very useful for extracting pressure field data anywhere from your 3D model. Enabling this feature will generate a small plane within your 3D model and using your mouse you can expand the plane by clicking and dragging out the 4 spherical points at each corner, move the plane by holding and dragging with the left mouse button and then using the cone shaped handles, you can orientate it at any angle you desire. There is also a coordinate system at the bottom left corner for you precisely situate the plane at specific points. The ‘Width’ and ‘Length’ parameters control the size of the plane. Best way to use the coordinate system is to pick one point, ‘Origin’, and make modifications to that point only as the rest will automatically change with it.

Individual
Spline Points



When you are satisfied with the placement of the cutting plane within your model, you can extract it using the 'Grab Cutting Plane Data' icon which will open up a new window on the interface. The splining tool extracts all the pressure data along the spline and displays the data on the graph on the right. The points on the spline can be moved allowing data to be extracted anywhere on the pressure plane. The general tools allow you to export the spline graph data to ASCII file (right-click graph), save the data as images and/or print out the active windows.



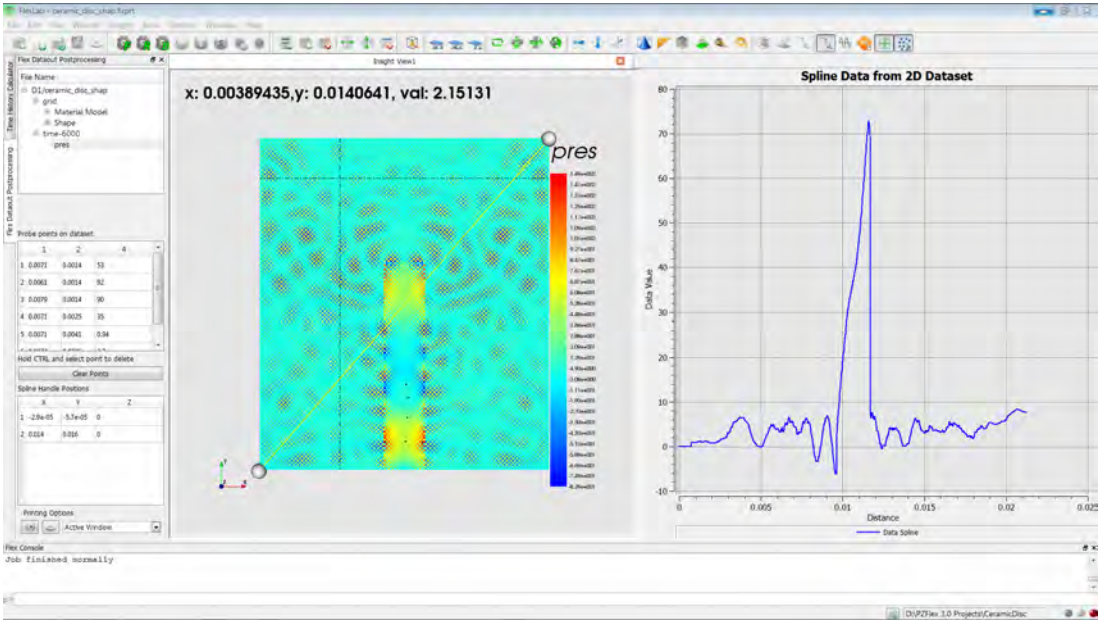
You can switch to individual spline points rather than using connected spline. Each point is noted in the dataset viewer at the bottom left corner of the interface, giving you information on the point's location and magnitude value. You can remove the points by clicking on the 'Clear Points' button (removes all points) or individually remove each point by holding CTRL and left-clicking on them directly.

The ‘Export Model File to STL’ icon (centre) allows model files to be converted into a format to be used by the Solidworks program. (*See Solidworks Interface Guide—P108*)

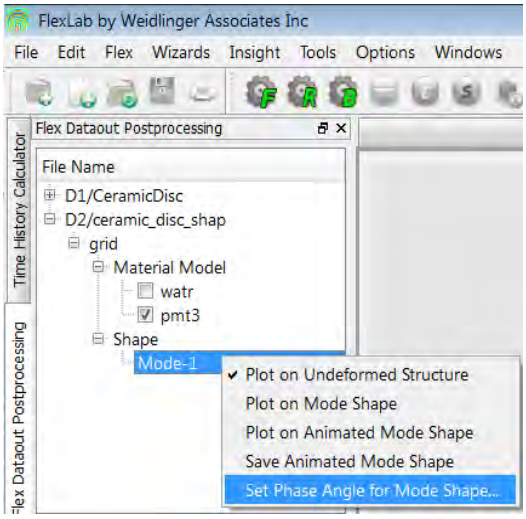
The other remaining icons should be familiar:

- Spline tool—generates spline to take pressure data from
- Spline plotter—plots pressure data from spline onto a graph
- Point probe—returns location and magnitude values of a point on the pressure plot
- Multi point probe—returns location and magnitude values of multiple point on the pressure plot

Note that these icons are only activated if the model under analysis is a 2D plain strain model with the exception of the STL exporter. The spline widget works in the same way as it does with 3D models. With the point probes, clicking on the pressure plot will show the point’s data within the active window itself or in the multipoint data viewer on the left depending on the widget selected.



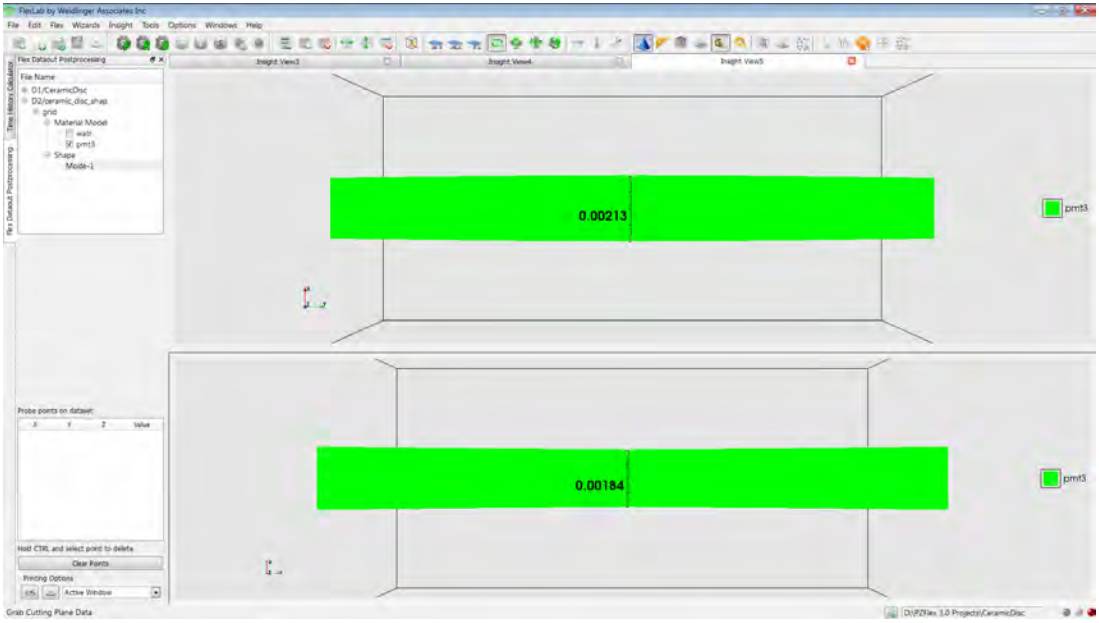
Mode Shapes



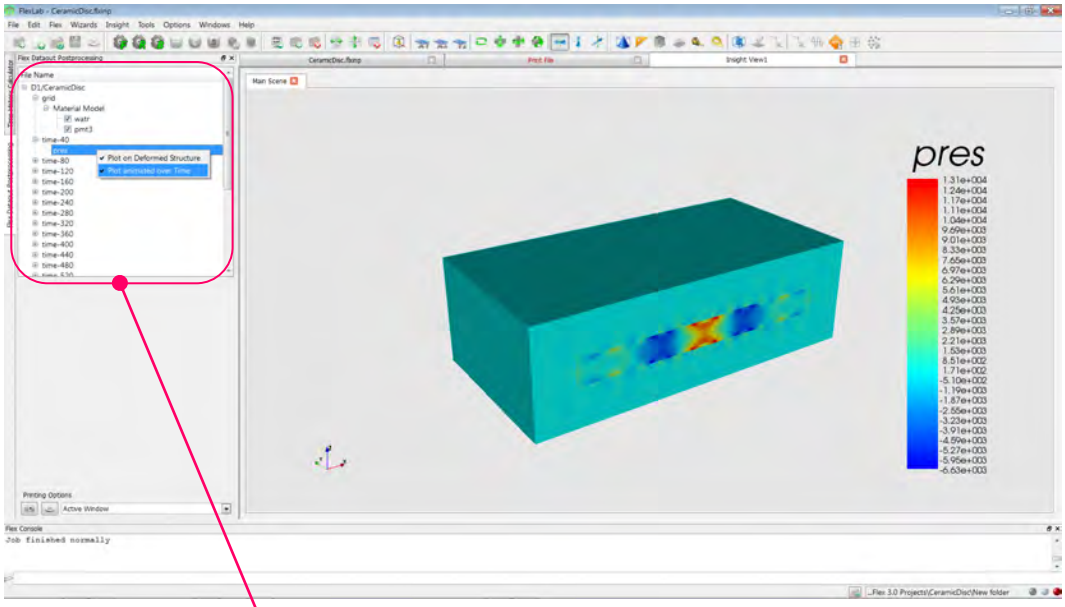
Mode shapes can be easily obtained from the model in the flex dataout interface.

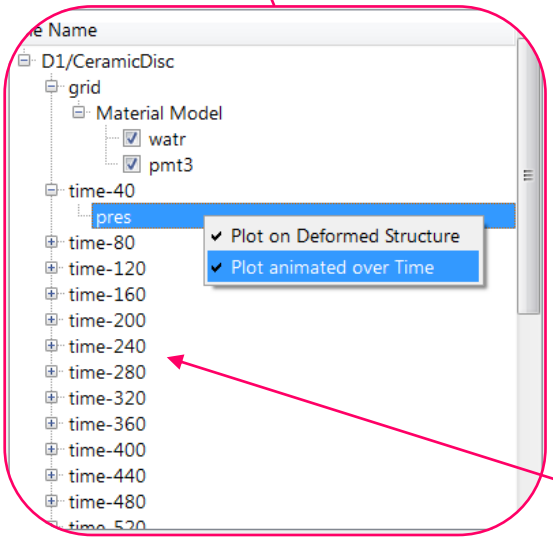
Right clicking on the 'Shape' data in the dataout browser will bring up a menu giving you options to:

- Plot on Undeformed Structure (relevant to pressure plots)
- Plot on Mode Shape (relevant to pressure plots)
- Plot on Animated Mode Shape—shows the animation of the device's mode shape on the interface
- Save Animated Mode Shape—saves animation as an .AVI file at your specified location
- Set Phase Angle for Mode Shape—choose to plot device at a specific phase angle



Pressure Wave Animations





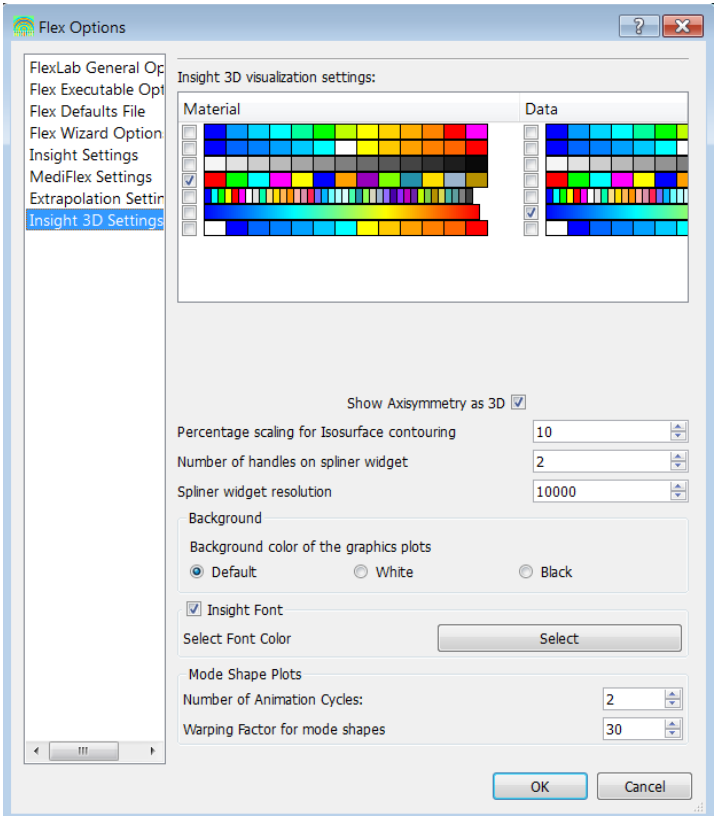
In Flex dataout you are given the option to plot an animation of the pressure waves traversing around your model. In order to animate it using the dataout interface, you are required to output the pressure data array within the execution loop in your flex model file. After the model executes fully you can access the pressure arrays within the data browser allowing the animation to be shown in the active window.

```
proc plot save

exec $nexec_loop
/*.....
data
    out pres
    end

end$ proc

proc plot $nloops
```

Settings — Insight 3D Settings

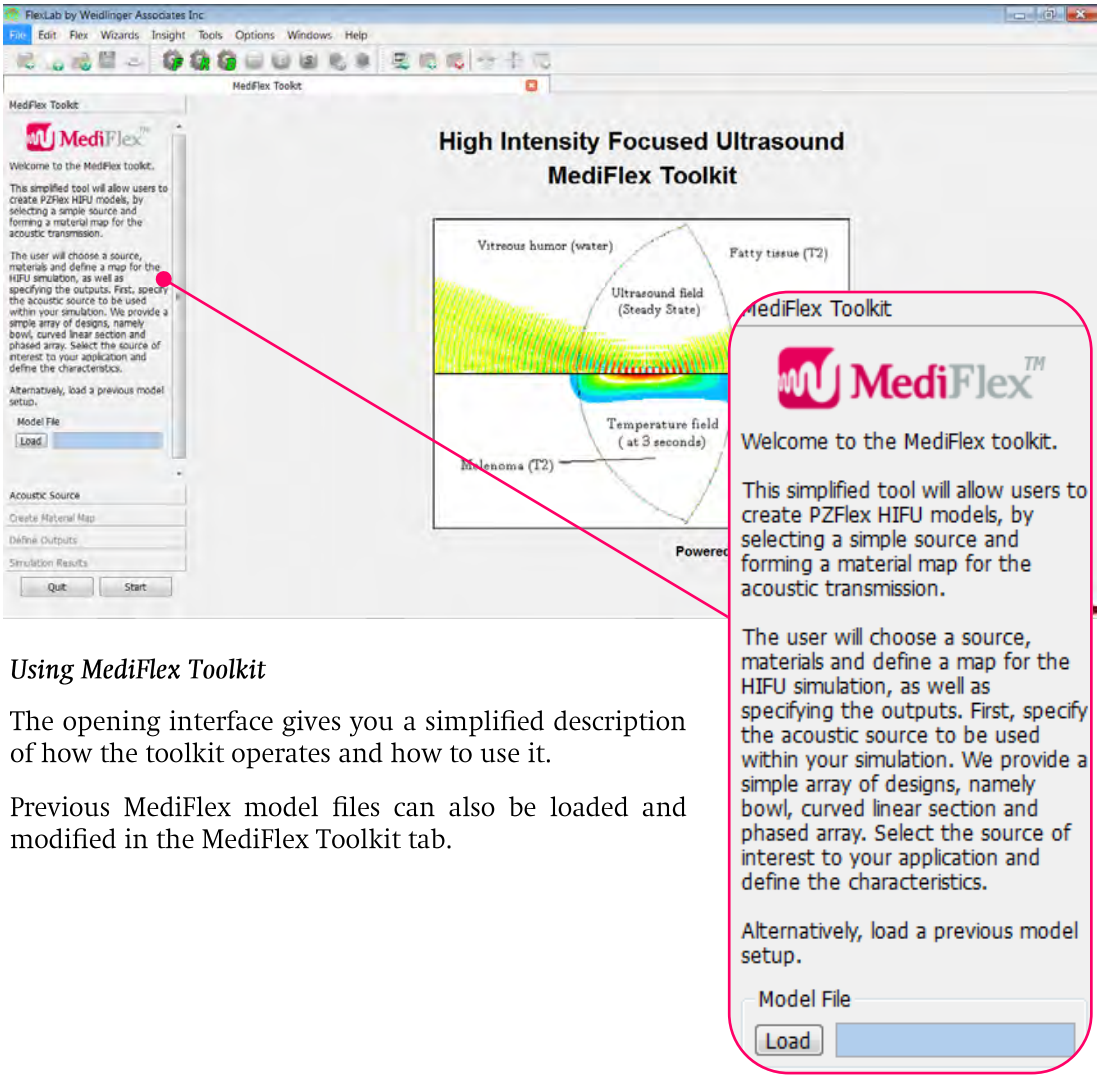
Within the main PZFlex settings there are options that governs its basic functionality:

- Default colour tables used in plots (materials and magnitude data)
- 2D axisymmetry models are displayed as 3D models (enable/disable)
- Percentage scaling for 2D isosurface contouring
- Number of handles on spliner widget—(affects 2D plain strain spline only)
- Spliner widget resolution—the number of sampled points along the length of spline (10000 max)
- Default colour of backgrounds for graphic plots (grey (default), white or black)
- Colour of font used in interface
- Number of Animation Cycles—the number of repetitions of the 0 to 360 degree phase of a mode shape when animated
- Warping Factor for mode shapes—the scaling of the mode shape displacement



MediFlex Tookit

The High Intensity Focused Ultrasound (HIFU) toolkit allows the user to select a theoretical source and model a medically-related wave propagation scenario in an intuitive manner. The graphical user interface lets you select the materials within the existing database or your own custom material file and add them to the model with the *Add (Shape)* functionality. The outputs from the model are then visualised through the insight graphics class to provide quick and simple feedback.

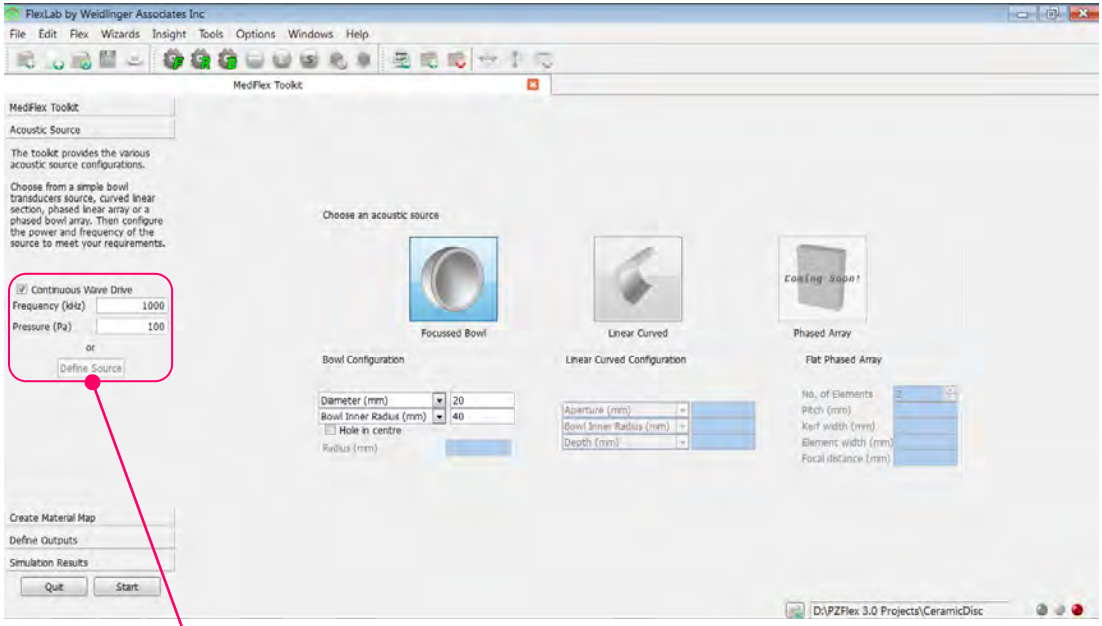


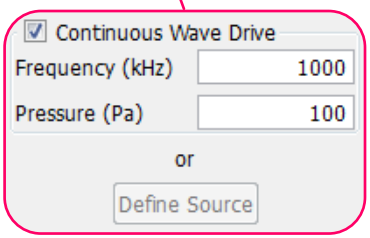
Using MediFlex Toolkit

The opening interface gives you a simplified description of how the toolkit operates and how to use it.

Previous MediFlex model files can also be loaded and modified in the MediFlex Toolkit tab.

Acoustic Source





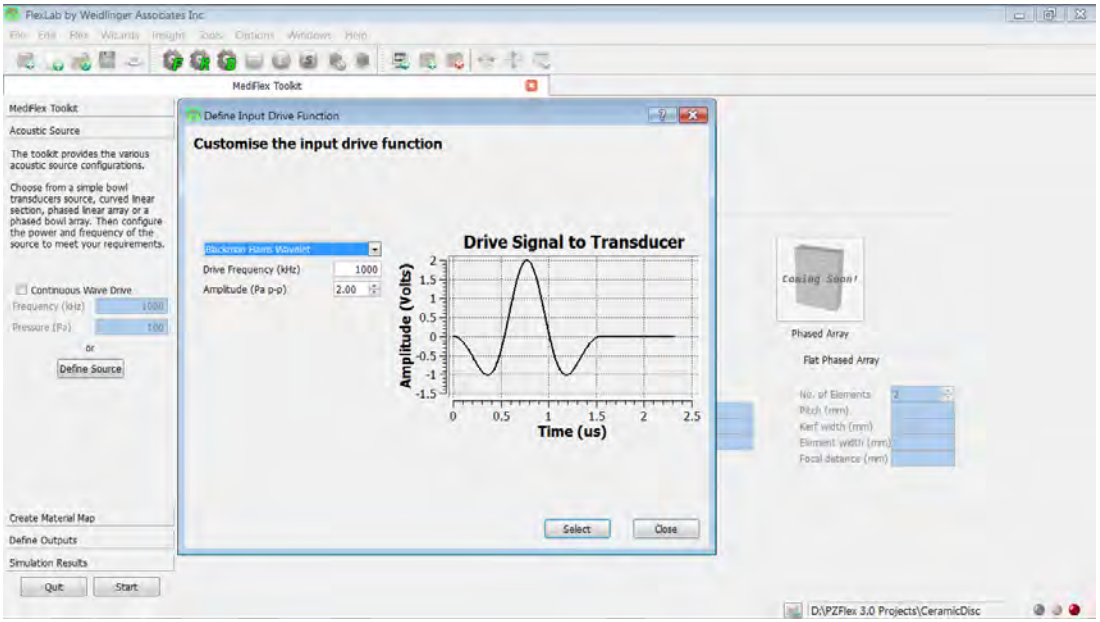
The Acoustic Source tab provides you with the common transducer configurations used in medical practice:

- Simple Bowl Transducers
- Curved Linear Sections
- Phased Linear Arrays (in development)
- Phased Bowl Arrays (in development)

With each type of transducer, there are related parameters underneath allowing their physical characteristics to be set accordingly.

After the transducer selection, the drive function can be set at the side. The default function is a ‘Continuous Wave Drive’ signal with the ‘Frequency’ and ‘Pressure’ control parameters to be set. There are many more drive functions available if you unselect the checkbox and click on ‘Define Source’.

Note if Thermal Drive simulations are required, remember to set the pressure of the drive function high enough for energy to reach the focal point.



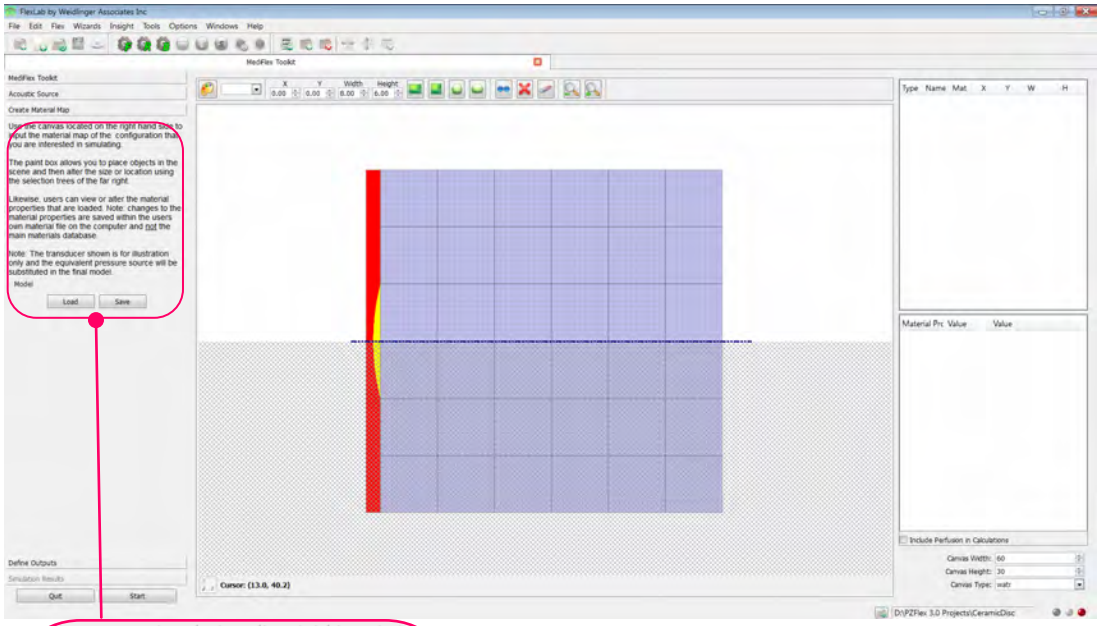
'Define Source' opens a new interface for you to select a different drive function. From the drop the menu, the following drive functions are offered:

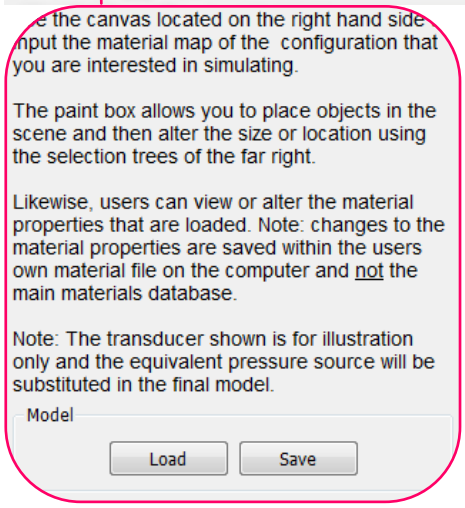
- Sinusoid Pulse
- Blackman Harris Wavelet
- Square Wave Pulse
- Chirp
- User Defined
- Rikert Wavelet
- Gaussian
- Apodised Step

Each function has their associated control parameters underneath and the graph on the right displays a preview of the drive signal.

With the drive functions chosen, you can begin to build up your model in the 'Create Material Map' tab.

Create Material Map



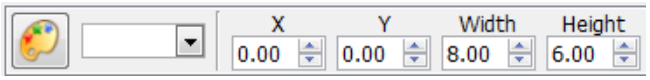


The interface offers a general description of the tab and allows you to load a previous model or save the current model generated.

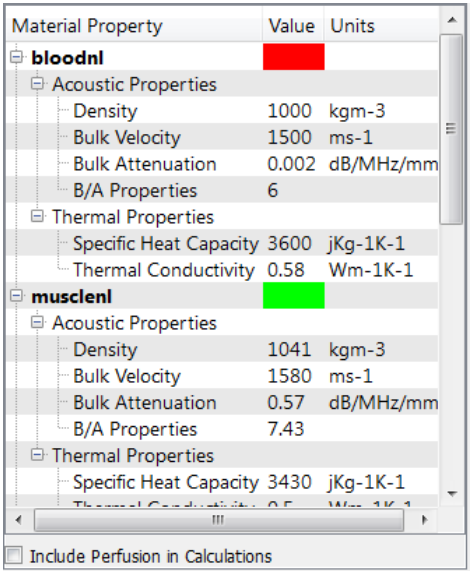
At the centre of the interface, there is the grid where you can generate the model. As you sweep the pointer across the top half of the grid you will notice the cursor reading will indicate the exact location of the pointer. This will be useful for placing materials later. Due to the symmetrical nature of the transducer, symmetry conditions will be applied to the grid itself.

At top of active window, there are unique icons used for building up your model.

On the right side, we have two sections that will show the different elements that are currently making up the model (top) and a section outlining the properties of the each material that can be used in the model (bottom).



The first icon you will be interacting with will be the 'Load materials for project' illustrated by the paint palette icon. This icon will open up the general PZFlex material database allowing you to select from the existing database or define your own materials to use. Select your materials and you can begin generating the model. You will notice you can choose the materials from the drop down menu (next to palette icon) and also the material properties are all displayed in the material browser section on the bottom right side of the interface. The XY, width and height are the default parameters controlling the material blocks generated using the 'Add' icons.

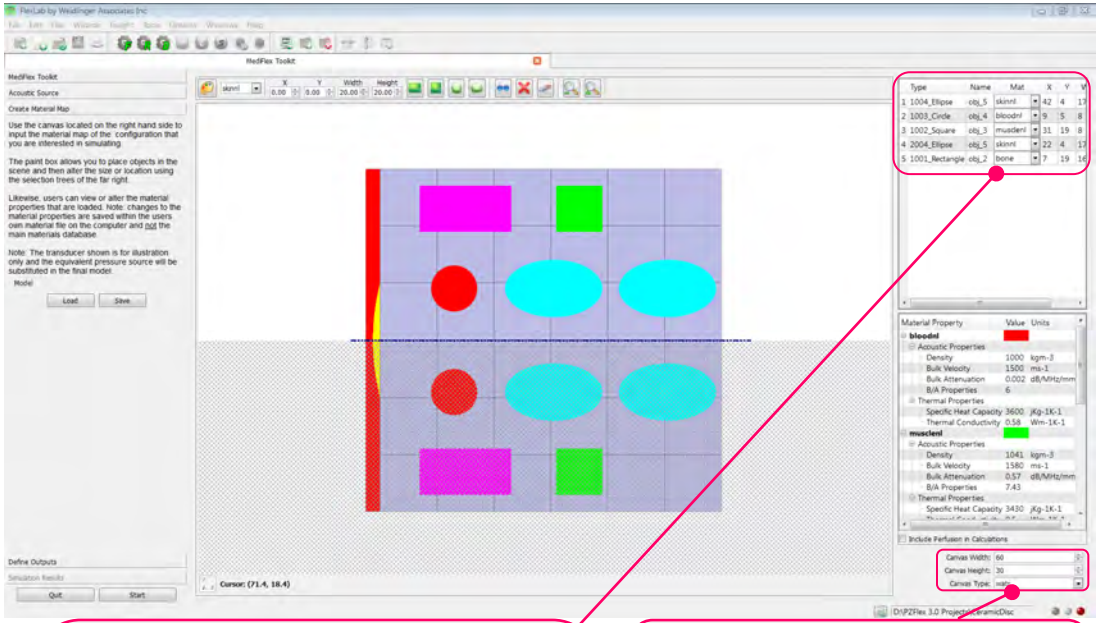


Note there is a checkbox to allow perfusion to be taken into account when modelling the materials.

The next icons all are related to the central material grid: they will allow you to navigate and 'paint' on your shapes that will represent parts of your model. Starting from the left:

- Add Rectangle—generates material block in the shape of rectangle
- Add Square—generates material block in the shape of a square
- Add Circle—generates material block in the shape of a circle
- Add Ellipse—generates material block in the shape of an ellipse
- Clone—creates a duplicate of current selected material
- Delete Object—deletes current material block selected
- Clear—clears all materials off of material grid
- Zoom out—zooms in on material grid
- Zoom in—zooms out from material grid

Note that the type of material added is chosen from the material drop down menu or altered in the element browser.



	Type	Name	Mat	X	Y	W
1	1004_Ellipse	obj_5	skinnl	42	4	17
2	1003_Circle	obj_4	bloodnl	9	5	8
3	1002_Square	obj_3	musclenl	31	19	8
4	2004_Ellipse	obj_5	skinnl	22	4	17
5	1001_Rectangle	obj_2	bone	7	19	16

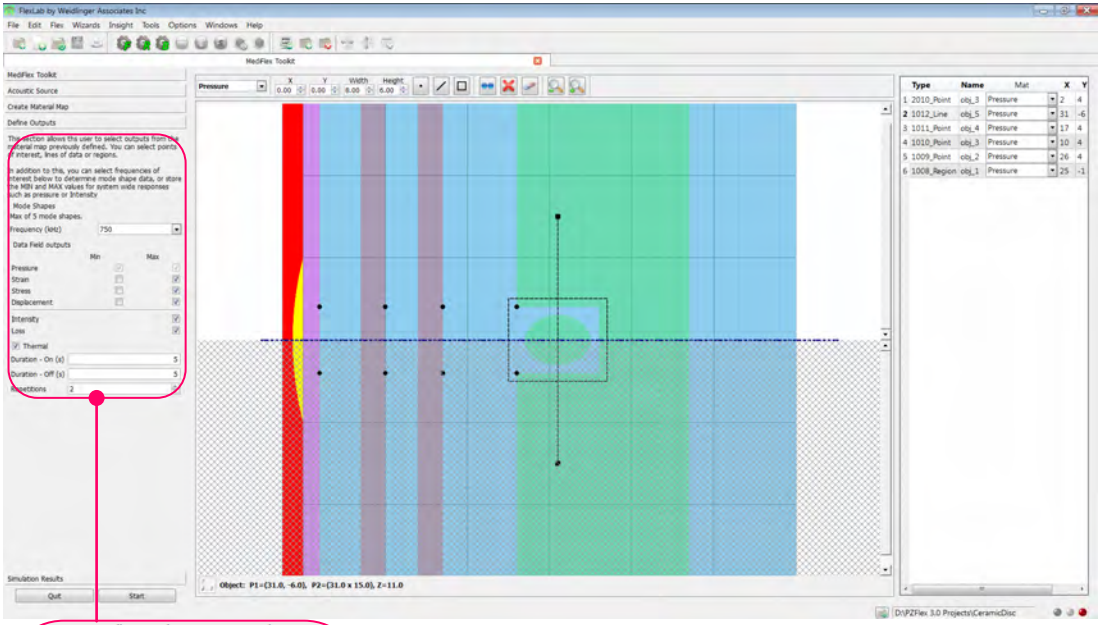
Canvas Width: 60
Canvas Height: 30
Canvas Type: watr

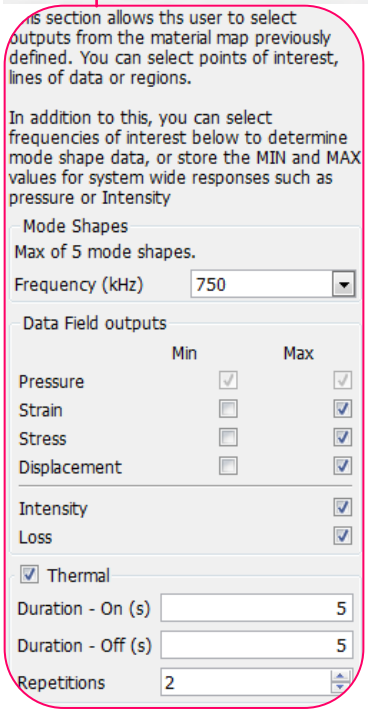
On the top right corner of the interface, you will notice that each material block added to the material grid appears in this browser. This browser allows you to:

- Alter the material of existing material blocks by using the drop down menu
- Relocate defined material blocks using the XY parameters (double-click to enter new XY values)
- Resize defined material blocks using the WH parameters (double-click to enter new WH values)
- Right-clicking on a material will show a menu enabling you to re-order the materials to control overlapping properties and control options to show and snap to grid

There are also settings to govern the size of the material canvas (how big the model will be) and the default material type if no materials are found on the canvas. The materials that can be selected are: water, water (non-linear) and void. When you are satisfied with the model, save the model and proceed to the 'Define Outputs' tab.

Define Outputs



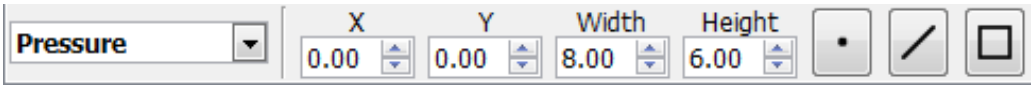


The outputs to calculate during the model are controlled from this part of the interface.

The toolkit offers a wide range of system responses that can be selected:

- Mode Shapes
- Pressure
- Strain
- Stress
- Displacement
- Intensity
- Loss
- Thermal (total duration should not exceed 90s)

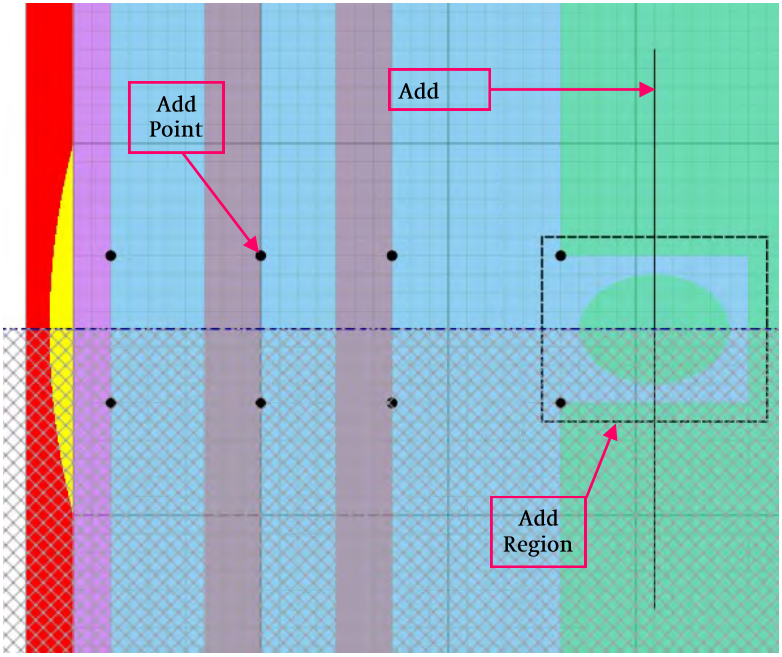
The MediFlex toolkit enables you to intuitively select precise points/regions of interest within the model to gain any of the system responses listed above.



The icon bar differs slightly from the 'Create Material Map' tab. The drop down menu allows you to select a particular system response associated with the extraction point/region added to the model.

	Type	Name	Mat	X	Y	W	H
1	2010_Point	obj_3	Pressure	2	4	0.1	0.1
2	1012_Line	obj_5	Loss	31	-6	31	15
3	1011_Point	obj_4	Pressure	17	4	0.1	0.1
4	1010_Point	obj_3	Pressure	10	4	0.1	0.1
5	1009_Point	obj_2	Intensity	26	4	0.1	0.1
6	1008_Region	obj_1	Pressure	25	-1	12	6

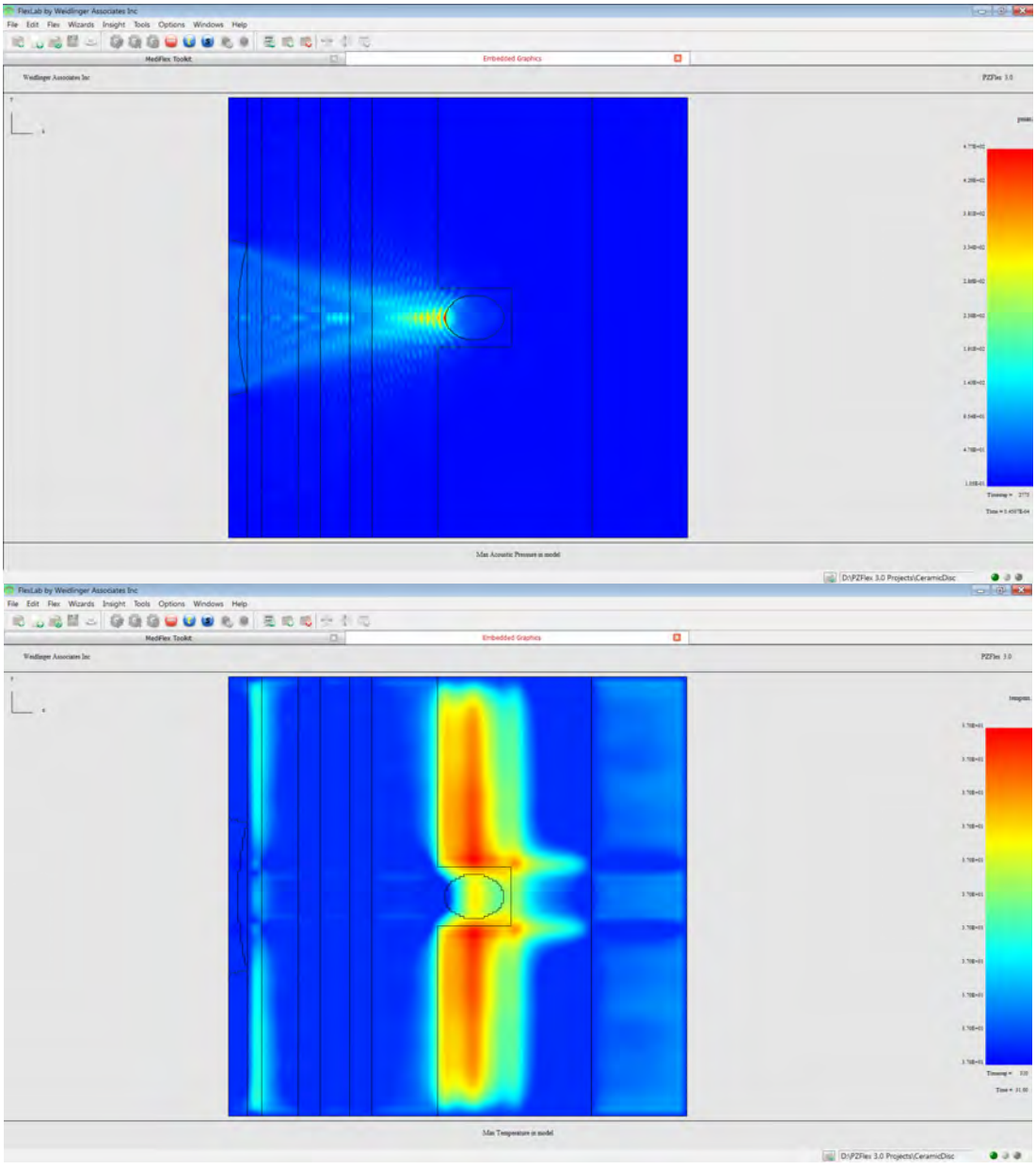
Using the 'Add Point', 'Add Line' and 'Add Region' tools, you can highlight specific areas to extract data from. Each point/region added to the model will appear on the browser on the right side of the interface. The system response to be obtained can be altered here using the drop down menu as well as the location (XY parameters) and size (WH parameters) depending on the extraction type.



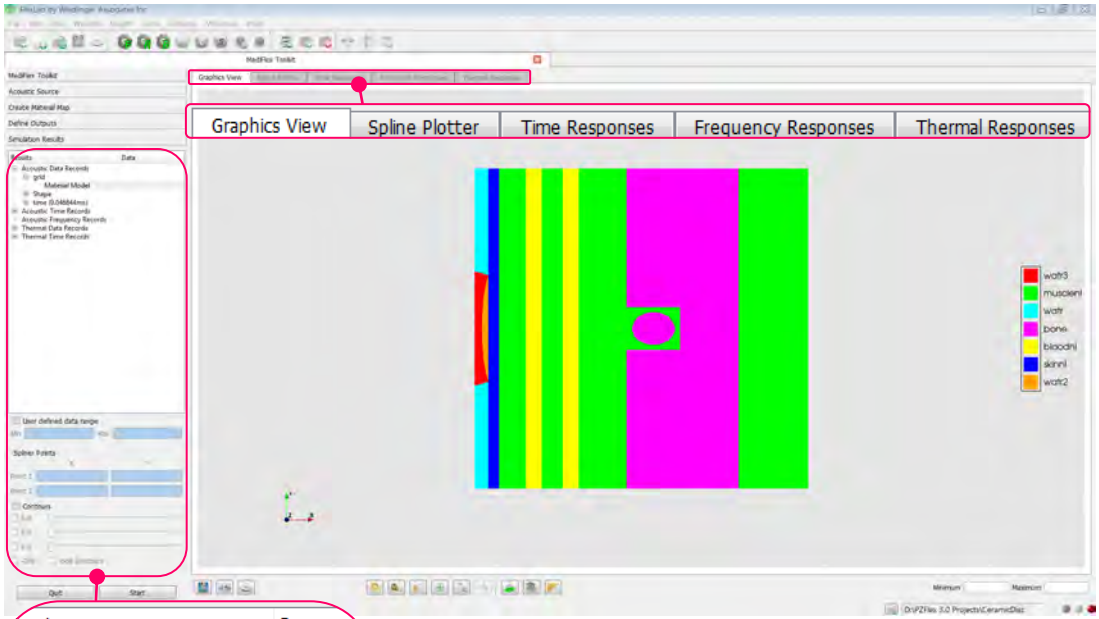
With your outputs all defined, save the model and run the simulation using the 'Start' button at the bottom left corner of the toolkit.

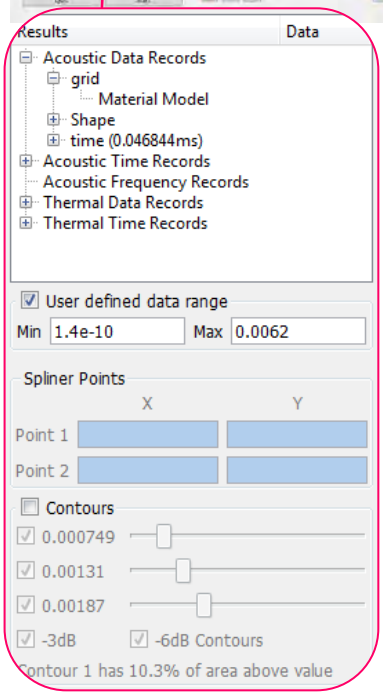
Simulation Results

Simulations screens encountered when running toolkit—wave propagation & pressure regions (top) and thermal drive response (bottom)



When the simulation is complete, you will be taken back to the Mediflex interface at the 'Simulation Results' tab.





The interface will have familiar tools to analyse and manipulate the output data.

The browser on the left gives access to all data outputs calculated from the model simulation. Right clicking an off the output parameters will give unique options to the data set such as 'Plot Data/Curve', 'Plot Frequency Response' and 'Export data to ASCII'.

Underneath there are controls for:

- Setting the amplitude scales
- Mapping out the spliner points on the graphical plot
- Contour controls for highlighting the different areas of pressure

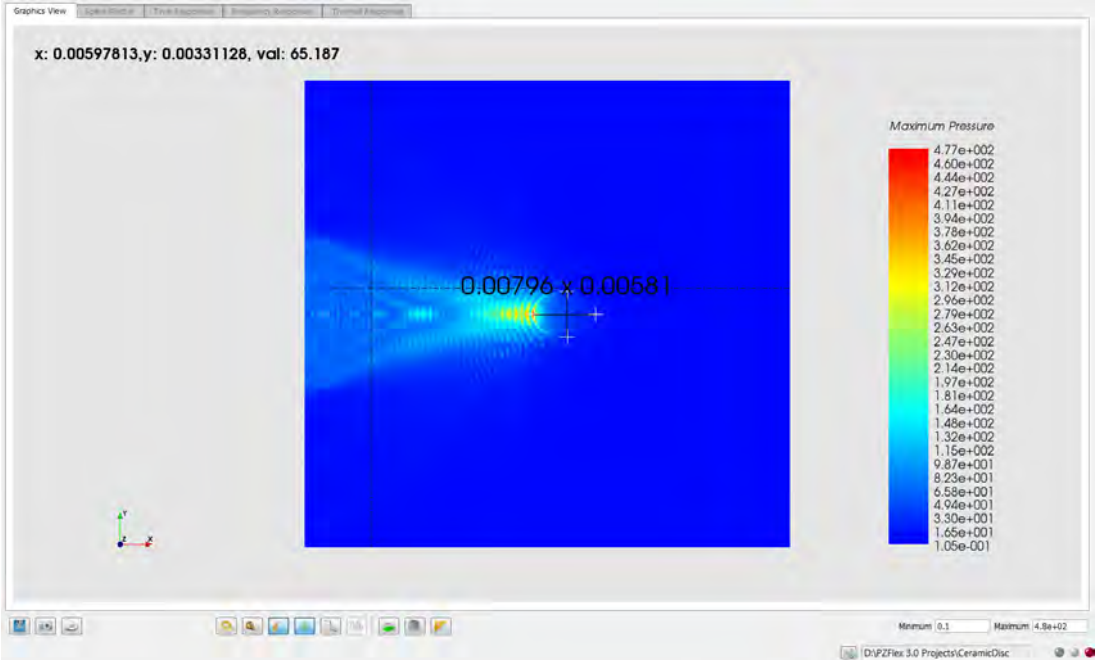
All graphical plots are viewed in the 'Graphics View' tab along the top of the interface which will allow the use of the interactive widgets and features. All other responses will be plotted in their corresponding viewer.

Within the 'Graphics View' tab, the widgets along the bottom offer the same functionality as found in the other toolkits. We have the usual generic tools forexporting to data to ASCII file, saving and printing off the active window plot. The widgets available in this toolkit consist of:

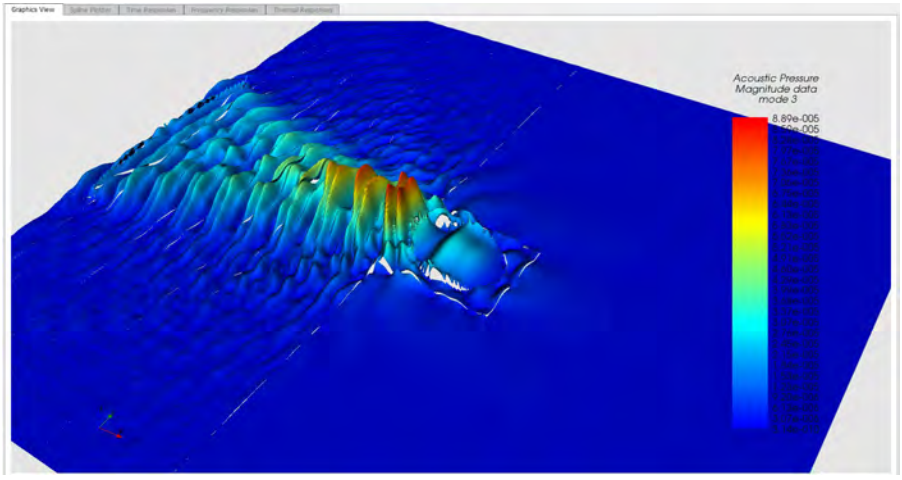
- Angle Measurer
- Distance Measurer
- Bidistance tool—used to determine area
- Pointer—identifies one specific point within the model returning the location and magnitude value
- Isosurface—highlights different areas of pressure using the 3rd dimension (Z dimension)
- Spliner and Exporter—obtains graph of the all the sampled points along spline
- Wireframe—maps out the individual elements of model
- Bounding Geometry—shows the dimensions of the scales along the X/Y axis

Using Pointer and Bidistance

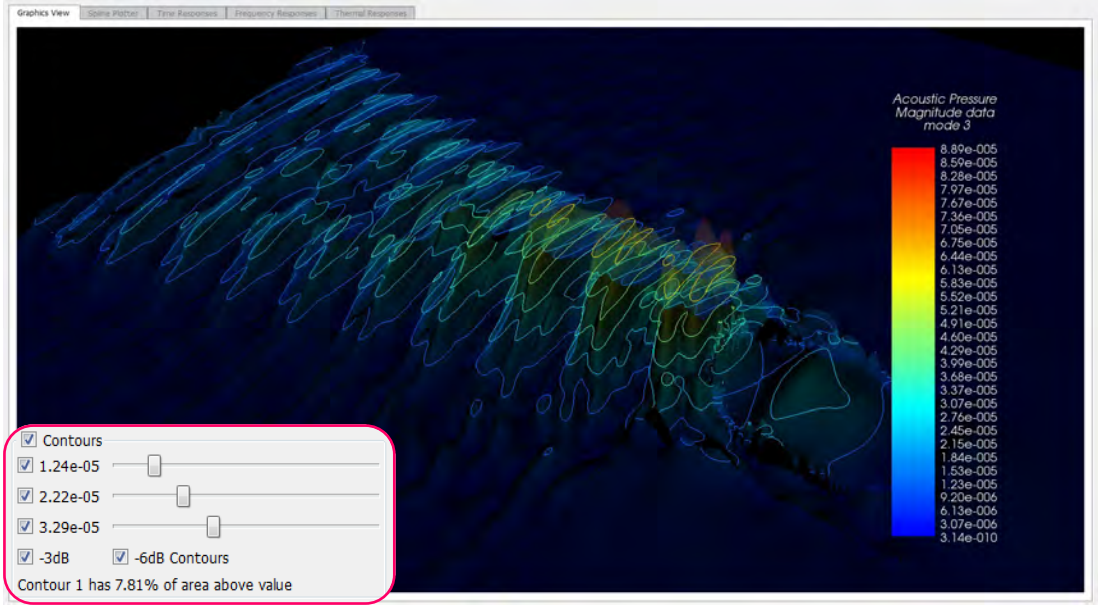
The tools activate by clicking on the plot: the points can be dragged out using the left mouse button.



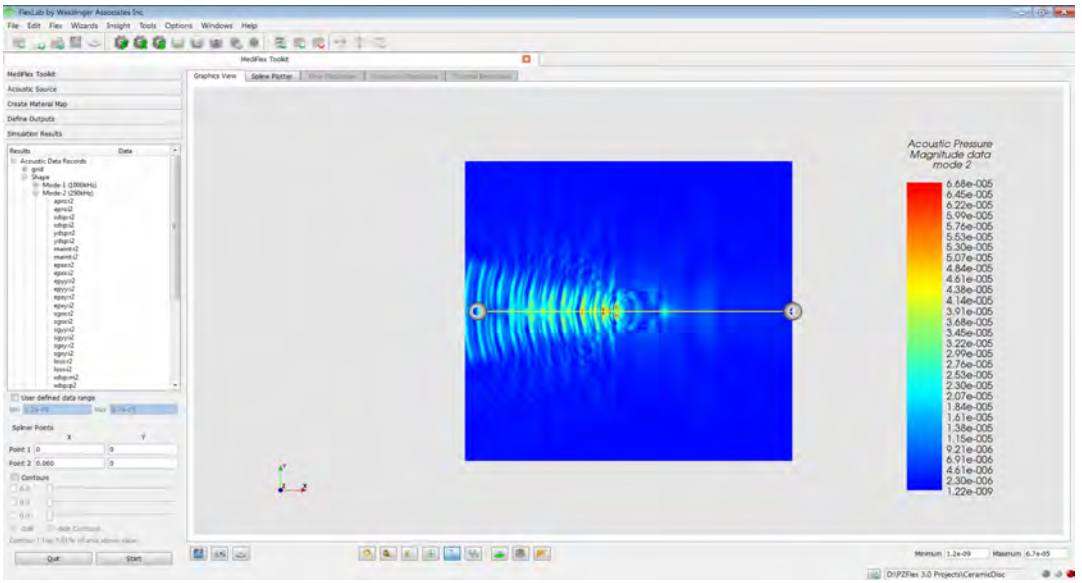
Using Isosurface and Contours



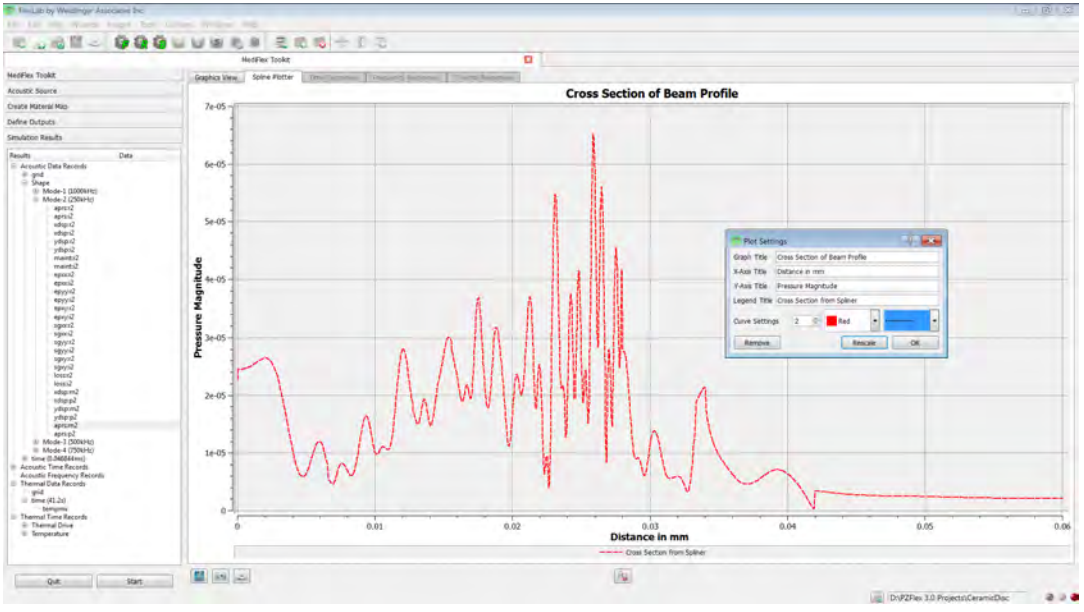
The Isosurface tool will automatically identify the different pressure areas and represent the intensity using the height in the Z dimension. The contour controls can also be used in conjuction with the Isosurface tool to assist in differentiating areas of pressure. The contour settings are easily set with the dragging controls and the predefined −3dB and −6dB amplitude checkboxes. The area above a certain magnitude is also indicated underneath the contour controls.



Using Spliner and Exporter

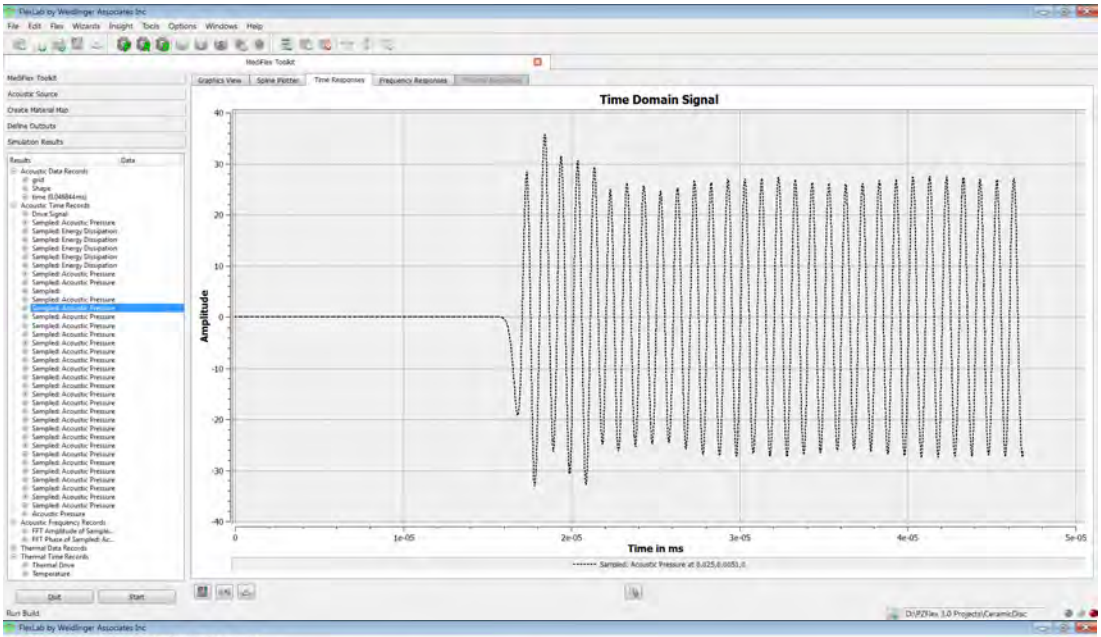


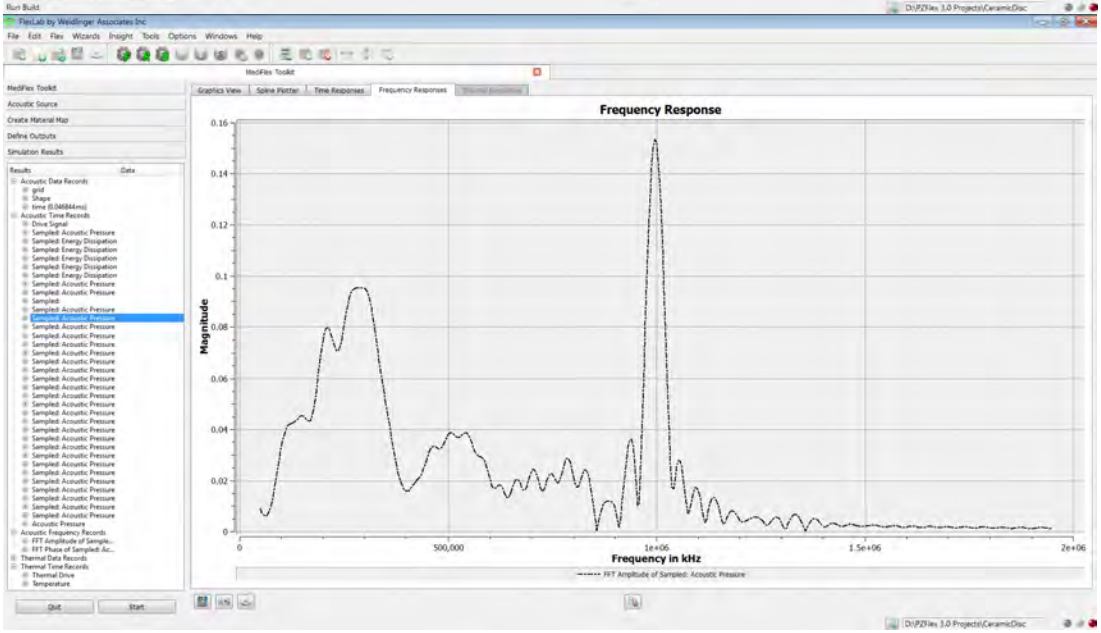
Select the 2 points for the spline and use the export icon to generate the 2D plot of the data. The 'Spline Plotter' viewing tab will contain the extracted data and will give you the option to alter the curve attributes.



Plotting Time Responses

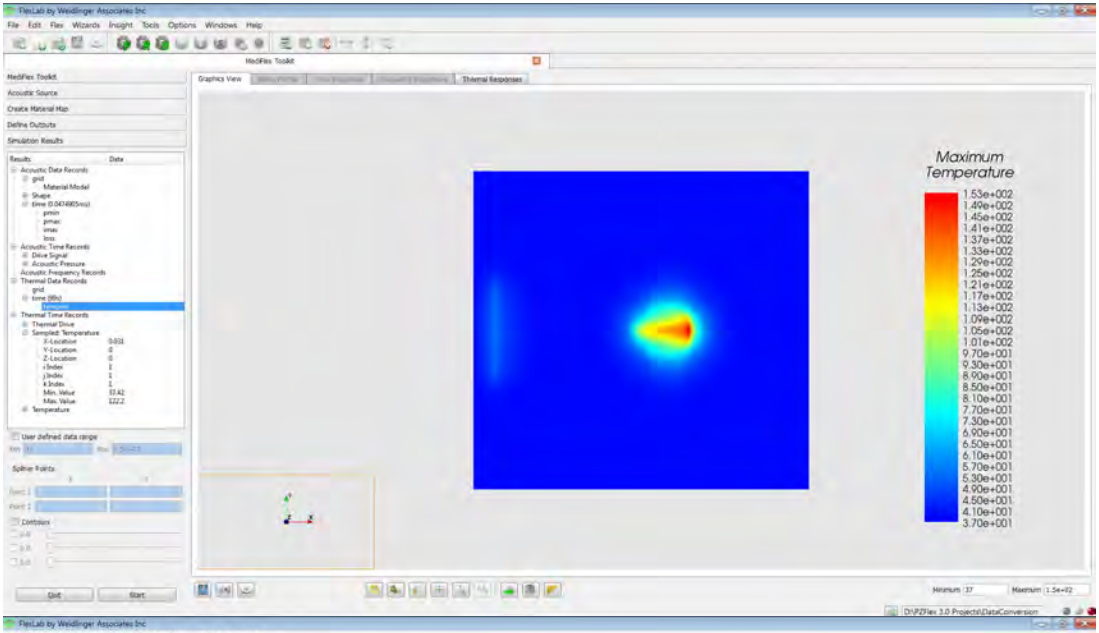
Using the results browser, locate the ‘Acoustic Time records’ and right click to plot either the time response or the frequency response.

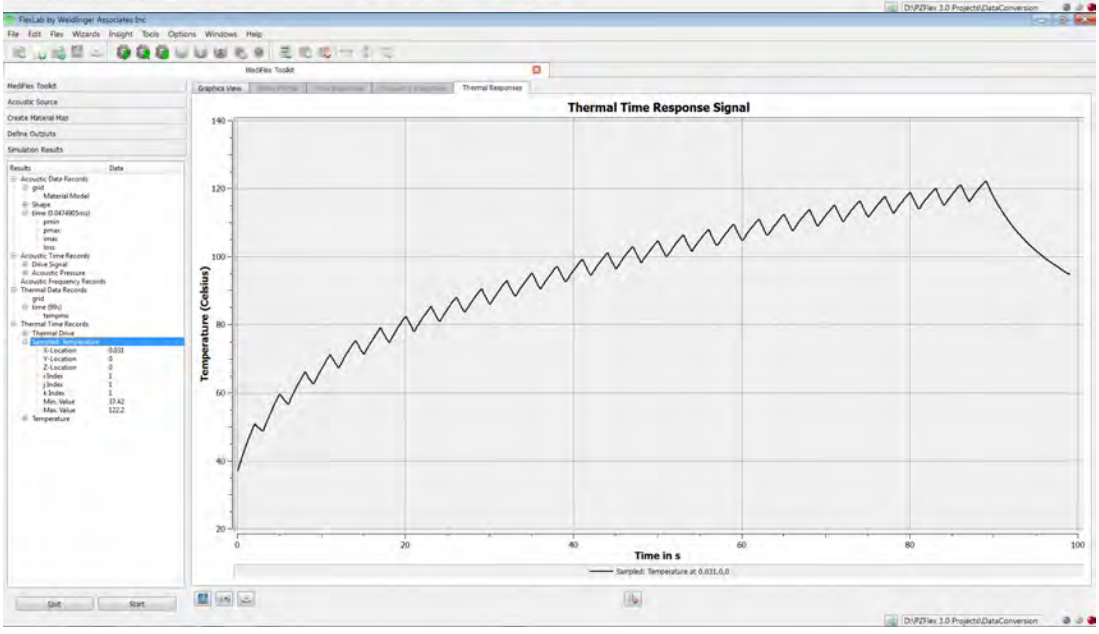




Plotting Thermal Responses

Locate ‘Thermal Data Records’ and ‘Thermal Time Records’ in the data browser to plot the max temperature plot and thermal responses respectively.

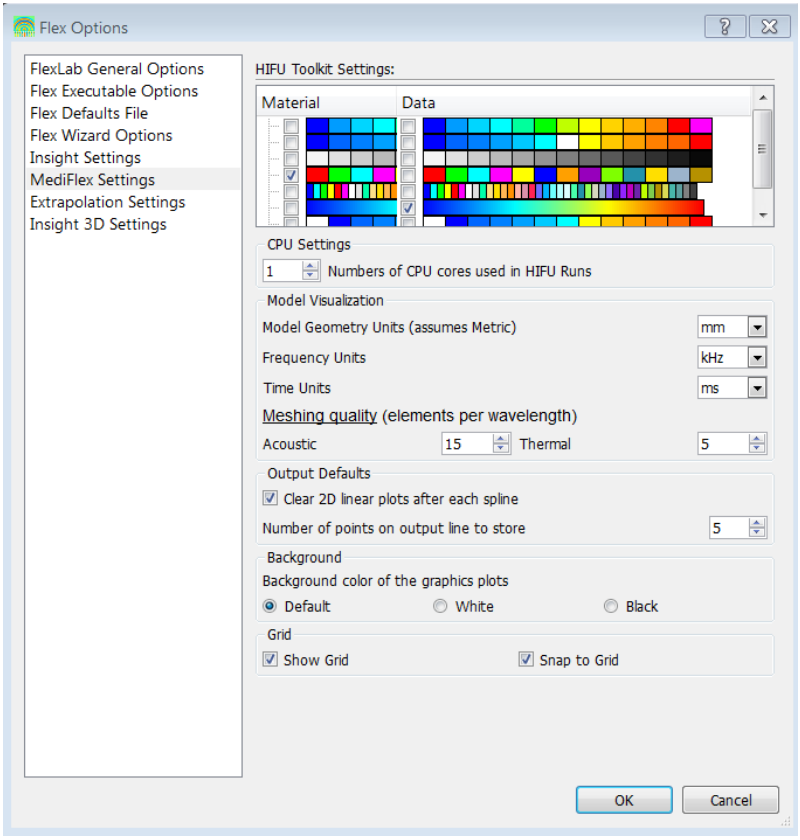




Settings—MediFlex Settings

Within the main PZFlex settings there are options that governs its basic functionality:

- Default colour tables used for distinguishing material types and magnitude intensities
- CPU resources allocated to the MediFlex toolkit
- Default units of measurements for graphical plots
- Mesh quality for acoustic propagations models and Thermal response calculations
- Default output settings—Clearing 2D line plots after each new spline is used and the number of points of data to store along the 'Add Line' tool in the 'Define Outputs' section of the toolkit
- Default background colour for interface graphics
- Show Grid and Snap to Grid—show grid of elements when mapping out materials and snap to grid when interacting with grid (useful for placing and resizing material blocks)



SolidWorks Interface Guide

PZFlex is proud to present the initial release of the SolidWorks modeling interface. This interface is designed to aid the user in building or importing models with complex geometry. The initial release allows the user to directly import STL files from SolidWorks and generate and simulate PZFlex models in record time. Once the basic geometry is imported via the interface, the user applies materials, generates the mesh and applies the basic load and boundary conditions to the model. Subsequently, the interface will generate a PZFlex input file that represents the model selected, ready for execution.

The SolidWorks graphics viewport in PZFlex allows the user to manipulate and orientate the model via OpenGL graphics to aid the user in visualization. The rest of this guide will illustrate the simple steps required to import a SolidWorks model and starting simulations.

While the interface allows the user to import and run simple examples, its main advantage comes in the ability to import complex geometrical models and create a structured PZFlex grid. The user will still be required to have a good working knowledge of the PZFlex scripting language to get the most out of these more complex models in terms of defining outputs and accessing the data.

Starting the SolidWorks Interface

Open PZFlex and select the Tools options from the menu bars at the top of the FlexLAB environment, then select “Import SolidWorks Model”, alternatively press “Shift and F9”.

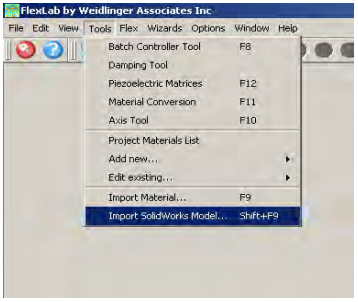


Figure 1 Starting the Interface

Selecting the SolidWorks interface will start a new window in the main program, that is split with a graphics viewport on the right-hand side and an options toolbox on the left like that shown below.

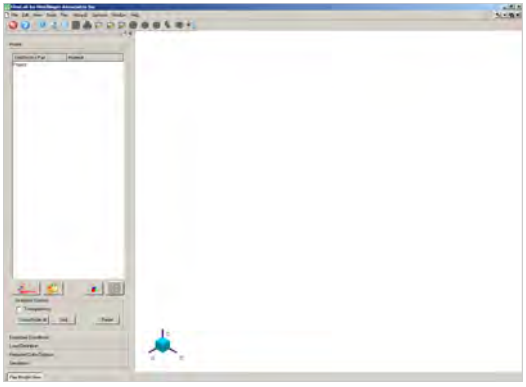


Figure 2 Main Interface

Step 1: Importing SolidWorks models

PZFlex is designed to read STL files produced from SolidWorks, so the user can now import solid models directly into FlexLAB. It also means that users can import models from other solid modeling packages as well, although they will have to use SolidWorks as an additional import step. SolidWorks is capable of converting Pro-E, GID and other commonly used modeling packages to STL format.

In order to import the STL files, select the SolidWorks icon on the interface, illustrated on the main interface controls (Fig 3), and then highlight parts of the model the user wishes to import.

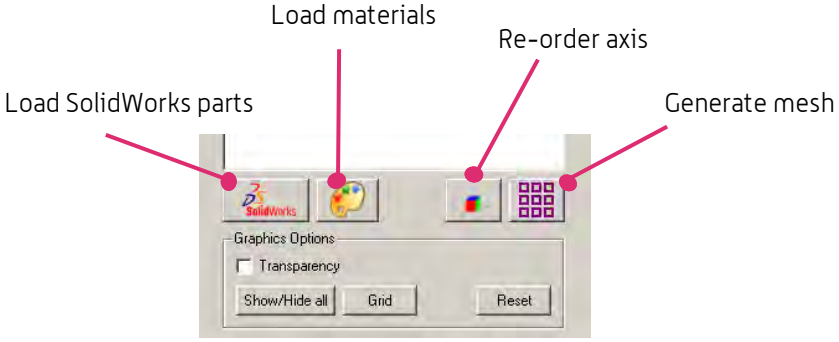


Figure 3 Main Interface controls

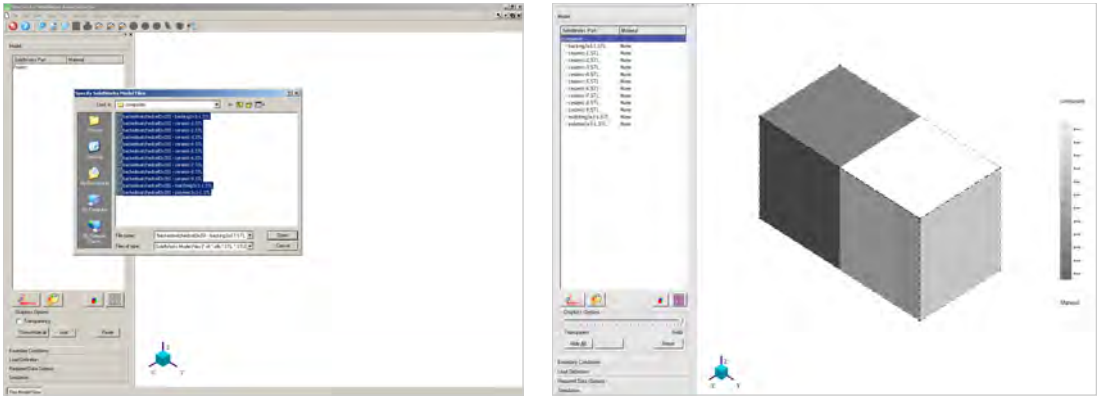


Figure 4 Importing models

After highlighting the parts of interest and importing them, the user will see the model in the graphics viewport on the right of the screen. Note that all the parts are shown in gray-scale. This is to illustrate that there are no materials associated with the model at this stage.

The graphics in the viewport can be manipulated in the normal OpenGL manner to zoom, pan and rotate the model as follows:

Rotate	*Hold down right mouse button and move mouse*
Zoom	*Ctrl + Hold down right mouse button and move mouse*
Pan	*Shift + Hold down right mouse button and move mouse*

If you wish to remove parts of the model, other commands can be accessed by clicking the right mouse button over the parts shown in the left-hand side menu on the interface. Extra parts can be removed, or hidden to allow the user to see internal sections of the model more clearly. Alternatively, the transparency slider on the interface can be used to examine inside the model.

Remember: Since you are now importing parts of the model, the model building is being done outside of PZFlex. Therefore, to model in quarter symmetry, you will only have to build a quarter of the model in SolidWorks. The user cannot manipulate the geometry of the model once it is imported into PZFlex.

Step 2: Applying materials to the model

Now that we have imported the model, we have to apply the correct materials to all the selected parts. To aid in this process, we access the material database supplied by PZFlex and form a materials project file for the importation. By selecting the “Load materials” icon, illustrated in Figure 3, or right clicking the mouse on the material section of the left-hand side toolbox, we can start the material database seen in Figure 5 and select the materials required.

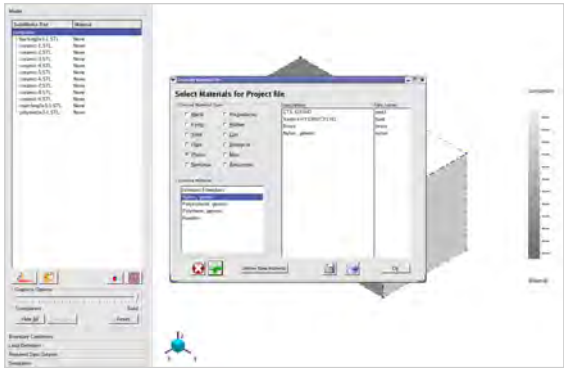


Figure 5 Materials selection

The material database has all the materials supplied by PZFlex divided into material groups, such as Piezoelectrics, Rubbers, Epoxies, etc. The user simply selects the materials of interest for this project and adds them to the material file on the right-hand side of the dialogue. In addition, if new materials are required that are not within the database, they can also be created within this dialogue and will be saved in a separate user defined database for future use. This project materials dialogue should be closed once the selection of the materials for the model has finished.

To apply the materials selected to the model, right click on the part in question and a selection box will appear with the materials in project list. Highlight and select the material required for the part.

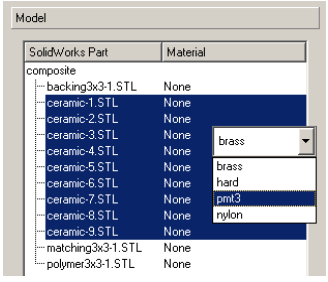


Figure 6 Assigning materials

Note, multiple parts can be selected and assigned the same material by dragging the selection box over the parts or by holding down CTRL or SHFT when selecting parts from the list.

Notice that as parts are assigned materials, the colour of the model changes and the legend on the right-hand side of the model changes to reflect the colours within the model as illustrated in Figure 7.

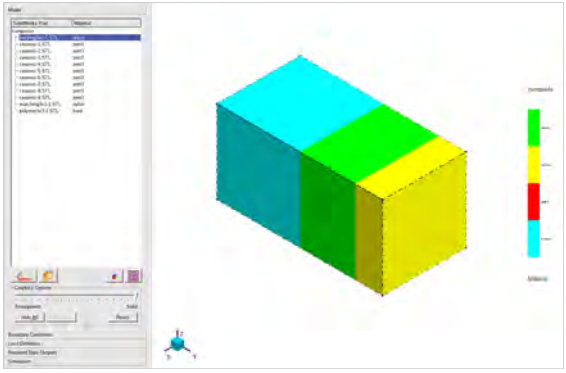


Figure 7 Materials assigned to model

Step 3: Realigning the model

Since it is easier to select nodes/element within PZFlex if the origin point within the model is known, it is recommended that one of the corners of the model is assigned to be the (0,0,0) origin. As SolidWorks tends to build models in arbitrary space and the origin is not necessarily assigned to a convenient point in the model, we have provided a means to translate your model to a known reference point. This will allow easy access to the node and elemental locations generated within the model.

To do this, select the re-order axis icon from the main icons shown previously in Figure 3, then select the one of the corners of the model from the graphics viewport. This point will become the new (0,0,0) of the model. In order to complete the transformation, you will also have to select another corner to indicate which way the z-axis will be orientated within the model.

When first point selection is made, a temporary axis will be shown on the model to show which way the axis is currently going to be orientated, as shown in Figure 8, this will remain until second point is selected.

Note that after the second selection is made the model within the viewport will shift to the new axis and may need refreshed.

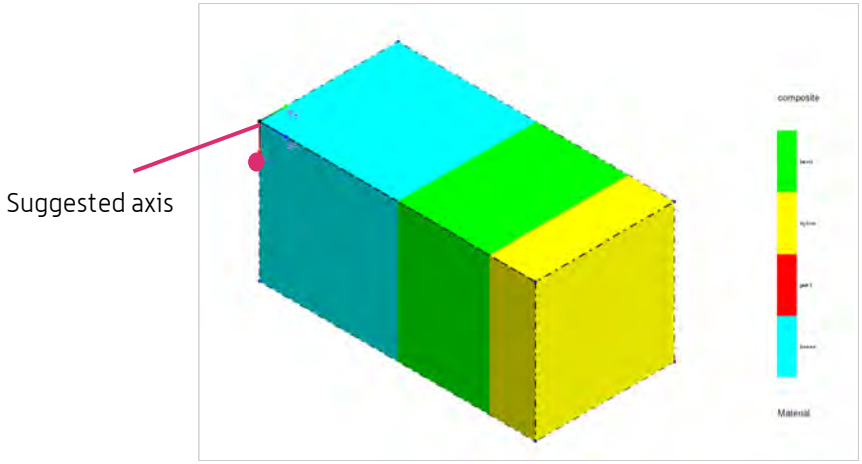


Figure 8 Axis selection

Step 4: Meshing the model

Now that we have completed all the preparation steps, we can now create the mesh for the model. Select the "Generate mesh" icon from the main controls as shown in Figure 3. The meshing dialogue, as shown in Figure 9, will now appear.

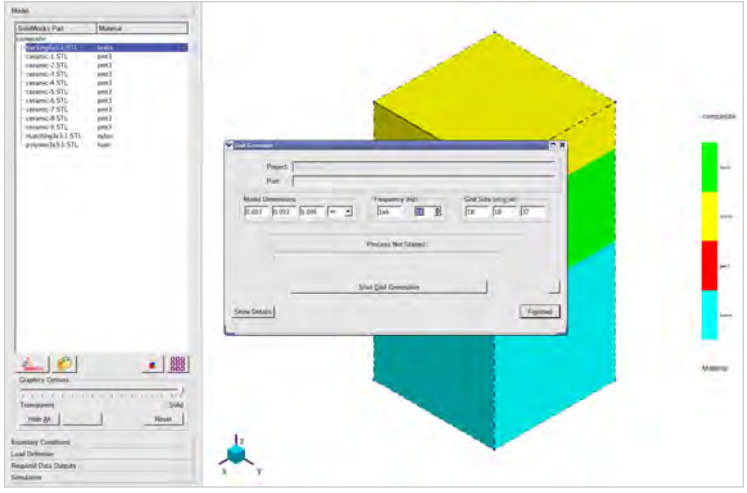


Figure 9 Generating the mesh

The meshing dialogue allows the user to build a grid for the imported model. Since the mesher has direct access to the materials used within the model, it can help generate the correct element size. The user inputs the frequency of interest for the model and the number of elements per wavelength required and the mesher will then calculate the element size required and display the number of elements that will be create in each direction.

Important:

> *Since SolidWorks does not pass the units in which the model was rendered within the STL file, you have to set the units in this section using the combo box provided. Therefore, if the units you used in SolidWorks were inches, please make the appropriate choice here. Incorrect unit selection will lead to false results.*

Once the frequency, model units and elements per wavelength have been selected, click the grid generation button and the model will be meshed to your specifications as shown in Figure 10.

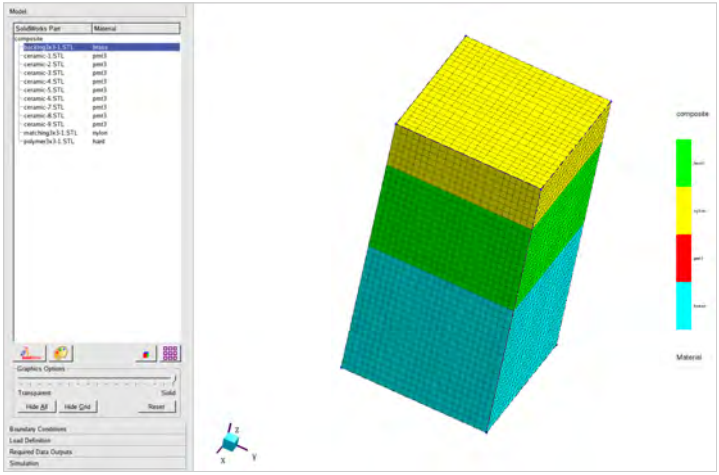


Figure 10 Meshed model

Congratulations, you have now successfully imported and meshed your SolidWorks model.

Now, we will show how to apply symmetry and load conditions to create a complete PZFlex simulation.

Step 5: Applying boundary conditions

Select the “Boundary Conditions” tab from the left-hand side toolbox. Select the side of interest on the graphic viewport and the associated side will be highlighted on the left-hand side toolbox, then simply select the boundary condition that you wish to have applied to this side.
Note that the default condition for any boundary is free.

Step 6: Applying Loads

There are two common ways to apply loads within a PZFlex model, either as a pressure load, or a electric drive function to a piezoelectric element. We have provided a simplified route to applying these loads. To apply loads, select the “Load definition” tab from the toolbox.

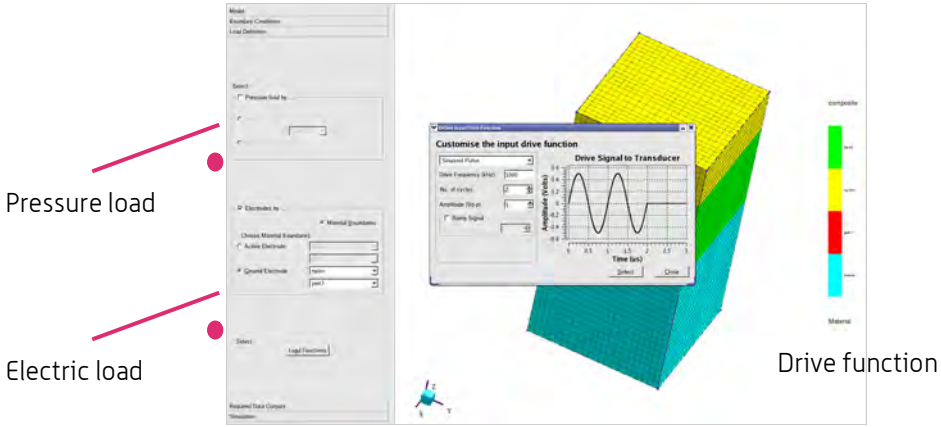


Figure 11 Load definition page

Figure 11 shows the layout of the toolbox with the pressure and electric load sections highlighted. Also shown is the drive function dialogue which allows the user to create different drive conditions, namely: Sine waves, Blackman wavelets, Square wave pulses, chirps and user defined. The user defined function allows the user to import two column ascii data to the model, where the first column is time and the second is amplitude of the drive signal.

Pressure Loads

Pressure loads can be applied to the model in two ways. Firstly, by selecting a side of the model that we wish the pressure load to be applied. Select the side in a similar manner to

the way the boundary conditions were selected, i.e. select the side by clicking the relevant side using the graphics viewport. The pressure loads will be applied normal to the selected surface.

Alternatively, you can select a boundary between different materials. To do this, select the material boundaries option and use the combo boxes to select the two materials between which the load is to be applied. The user then has to select the direction of the pressure force, either by supplying a vector or allocating a point in space that the pressure is focused towards.

Electric Loads

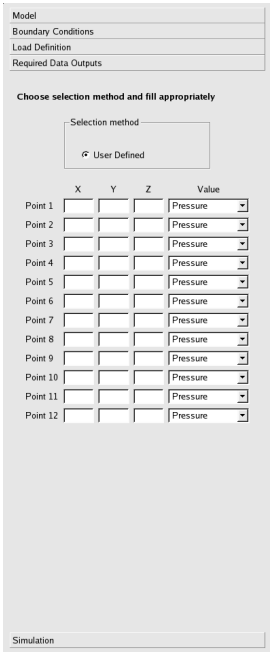
Electric loads are applied in a similar manner to that of the pressure loads just described. The user selects the material boundaries for the ground and drive electrodes to be applied. For example, in a composite the ground plane could be applied between the bottom of the ceramic pillars and the backing layer, while the drive electrodes could be at the top of the ceramic pillars and the matching layer.
When the user selects an electric load, the software automatically computes the electric window for the model and applies this condition.

Step 7: Selecting Outputs

Selection of outputs can be done within the “Required Data Outputs” section of the tool box on the left-hand side.

Currently, the only method for selecting points within the model for generating output data is to manually enter the co-ordinate location of the points of interest. Please ensure that these points are actually within the model, as points selected outside the model will not be stored. Work is continuing to make the selection of output data simpler.

The user should input the x, y and z location of the points of interest within the model and then select the information to be recorded at this point. Information that can be stored is displacement in the x, y or z direction and pressure.



Step 8: Running the model

Once the user has completed the following steps, the final requirement for the simulation run is to define the simulation time. As PZFlex is a time domain code, we have to run the model for a number of time steps to complete the simulation. If the user knows the time period over which they wish the run to simulate, they should enter the time in the dialogue on the “Simulation” page of the interface toolbox.
Alternatively, if the model in question involves piezoelectric materials, they can opt to use the automatic simulation mode. This examines the charge present in the electric window of the simulation and continues until such time as the majority (>95%) of the charge has dissipated.

The user can now export the whole model to PZFlex by pressing the “Start Simulation” button. The model is then converted to PZFlex format and two additional windows open. These windows contain the PZFlex input file for the model and the project material file. This allows the designer to see the format into which the models have be converted and insert any additional commands that are required.

Further Help

If you require any further assistance with the SolidWorks interface, or find any problems during its operation, please feel free to contact us via email at support@pzflex.com.

USING PZFlex

Finite-Element Analysis

PZFlex uses finite-element analysis (FEA) to simulate an object's response to specific loading conditions. The foundation of FEA is the computer model of the object. FEA modeling is based on breaking up the physical reality of a shape into smaller, discrete elements. To model the pressure a computer monitor exerts on a table, for example, we break up the table into smaller elements. We then compute how the pressure is distributed across the table. The overall structure of the table is the same even though it is broken up into smaller elements:

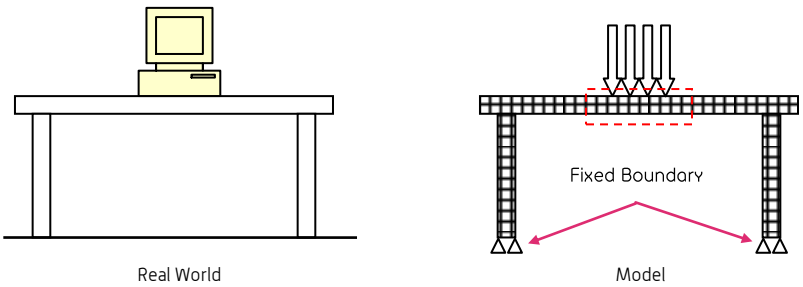


Figure 1

PZFlex calculates the stresses and strains on a single element. It then transfers these stresses and strains proportionally to the surrounding elements. To understand how it does this, it is helpful to consider the structure of the individual element.

An element consists of individual nodes connected together. Every element has either four nodes (for 2D modeling) or eight nodes (for 3D modeling):

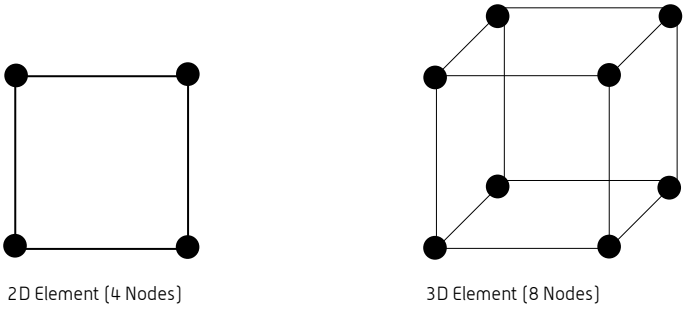


Figure 2

When a load is applied to a node, PZFlex first computes the distribution of that load to the surrounding nodes, then to the next set of nodes, and so on.

In the model in Fig. 1, the computer has been replaced by a pressure load. Although it is possible to model the entire computer, to do so would be a waste of time and resources if we are interested only in the behavior of the table. Instead, we place the equivalent pressure load on the elements of the table's surface where the computer rests (defined by the red dashed line). This produces the same results as building a model of the computer, but in a fraction of the time.

The goal in model building is to make the model both as close to reality and as simple as possible. The more complex a model is, the longer it takes to run. One way to simplify a model is to apply symmetry to it. Using the example of the table, we can reduce the model to:

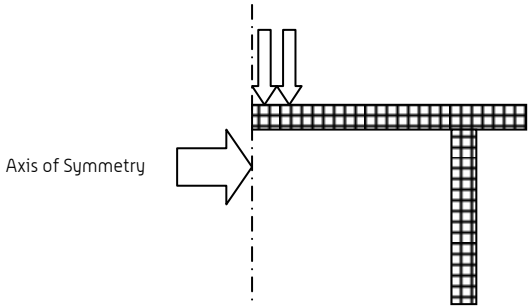


Figure 3

Because the computer load is in the center of the table, and the table is symmetrical about the center axis, we can apply a symmetrical boundary in the middle. Such a boundary is called a "boundary condition." A variety of boundary conditions can be applied to models.

When you run the model of the table, you see how the pressure is distributed across it. This shows where the pressure would be in the real table. A table designer can use this information to determine how strong the legs need to be to resist the pressure of the computer.

PZFlex is based on the same principles as the problem of the computer on the table. Instead of a static load, however, it applies varying loads (e.g., someone jumping up and down on the table) to see how the object reacts. What makes PZFlex different is that you can include the electrical properties of piezoelectric materials, to model the piezoelectric effect and the resultant effect on the model. (When an electric field is applied across a piezoelectric material, it causes a deformation of the material's shape. Conversely, deforming a piezoelectric material generates an electric charge proportional to the degree of deformation.)

PZFlex Basics

PZFlex consists of two main components: 1) PZFlex itself, which does the pre-processing (model building and assignment of material properties) and simulation and 2) Review, which post-processes the results (creates output displays and graphs).

When building a model in PZFlex, we usually start by creating an input file to define aspects of the model such as its geometry, material properties, driving functions, and boundary conditions. This input file is saved under the file name:


```
<jobname>.flxinp
```

where `<jobname>` is a name you assign to the file for ease of reference. The `<jobname>` cannot exceed 20 characters and cannot include spaces.

The simplest way to execute this file is via the FlexLab interface.

As PZFlex processes the input file, it generates a print file, `<jobname>.flxprt`. This is a history of all the commands entered during the pre-processing of the model. If PZFlex encounters a problem while running an input file, it generates a warning file, `<jobname>.flxwrn`, which later can be used in conjunction with the print file for debugging.

PZFlex also generates a symbol file, `<jobname>.symb`, which lists the model variables. This allows you to verify that you have assigned them correctly.

Once PZFlex has processed the information in the input file and carried out the appropriate calculations, it stores the simulation results in a file named `<jobname>.flxhst`. You can then use **Review** or **Insight** to view the results.

Building a Simple Model

Writing an input file for PZFlex is similar to writing any kind of program, whether in C or assembly language.

The input file, which tells PZFlex how the model is constructed, can be broken down into seven sections:

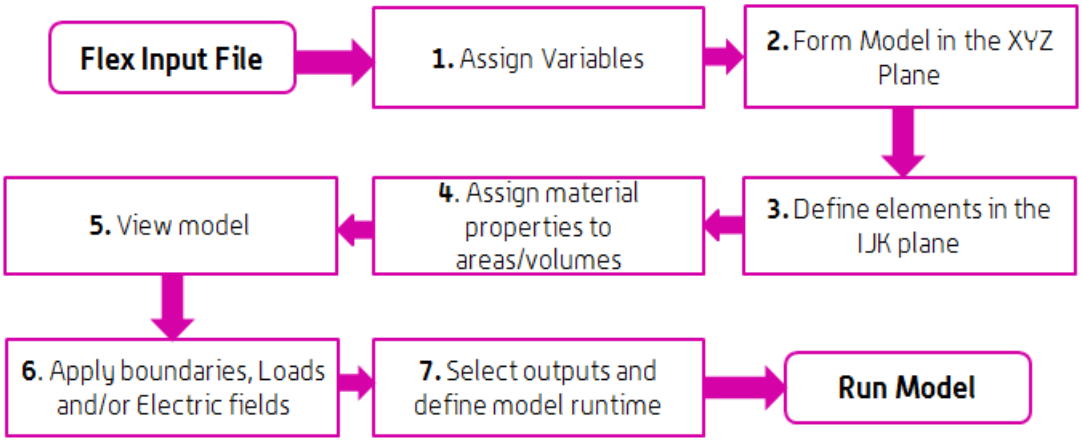


Figure 4

1. Defining the Variables

In all programming, it is best to begin by defining commonly used variables. Once you globally define a common parameter, say, the thickness of a ceramic plate, there is no need to remember the value each time. To define a variable, assign it a name and a value. Although the name can be up to 20 characters long, using as few characters as possible makes it easier to remember; it also takes up less space on the line.

Taking the example of the ceramic plate thickness, we can name the variable “ceramic thickness”—cer_thk.

PZFlex is a user-defined unit system. That means you can work in any parameter system you prefer (for example, lbs. and inches vs. meters and kilograms), as long as you are consistent.

To define a ceramic thickness of 2mm, enter:

```
symb cer_thk = 2.e-3
```

You must include a space between the name of the variable and the equal sign and between the equal sign and the value. Later, we will use“$cer_thk” to refer to this variable. To insert the variable number stored in “cer_thk” we use “$” (which means “the value of”).

For a negative value, enter “-$cer_thk”; there is no space between the negative sign and variable name.

The symbol command “symb” is used here to define a variable. It allows you to place variable names as arguments on any command line; to embed programming statements (variable assignments, mathematical operations, looping, and branching) in the PZFlex input file, and to define procedures (macros), a series of commands that can be called up with a single command. For more on the symbol command, see the Command Reference.

Below is a slightly more complex variable definition:

```
c               MODEL PARAMETERS
c

c List of all model parameters in Imperial units of inches
c ========================================

symb cerwdthy = 0.101               /* ceramic pillar width in x direction
symb cerwdthx = 0.095               /* ceramic pillar width in y direction
symb certhk1 = 0.5                  /* Thickness of first ceramic in z direction
symb certhk2 = 0.5                  /* Thickness of second ceramic in z direction
symb polykerfy = 0.057              /* Polymer kerf width in x axis
symb polykerfx = 0.061              /* Polymer kerf width in y axis
symb backing = 0.72                 /* Thickness of backing material
symb frontml1 = 0.85                /* First front face matching layer thickness
symb txdrsz = 0.75                  /* Overall size of transducer
symb dgtxdrsz = sqrt ( ( $txdrsz * $txdrsz ) / 2 )      /* Diagonal length of transducer
```

The “c” at the start of a line followed by a space allows you to enter a comment in the code, which is ignored by PZFlex. An example might be a section description to facilitate later debugging. You can also insert comment on lines of code you have written by entering “/*” followed by a space.

The mathematical operator for “square root” is “sqrt”; other mathematical operators can be found in the Command Reference. In this case, you could have used

```
symb dgtxdrsz = sqrt ( ( $txdrsz ** 2 ) / 2 )
```

instead of

```
symb dgtxdrsz = sqrt ( ( $txdrsz * $txdrsz ) / 2 )
```

to find the square of “$txdr”; likewise, “$txdr ** 3” provides the cube of the value. Again, all operators are separated by a space, and “$” indicates the values.

As noted above, you can work in whatever units you wish. Here we convert the above distances to metric units:


```
c Conversion to metric
c ================

symb cerwdthxm = $cerwdthx * 0.0254          /* Metric pillar width in  x direction
symb cerwdthym = $cerwdthy * 0.0254          /* Metric pillar width in  y direction
symb certhk1m = $certhk1 * 0.0254            /* Metric ceramic thickness in  z direction
symb certhk2m = $certhk2 * 0.0254            /* Metric ceramic thickness in  z direction
```

For clarity, we use different names for the variables in Imperial units and in metric units. The metric values have the same name except for the suffix “m.” An alternative is simply to overwrite the existing value when converting it:

```
symb cerwdthx = $cerwdthx * 0.0254           /* Metric pillar width in  x direction
```

The previously defined value for “cerwdthx” is multiplied by 0.0254 and stored in the same variable, overwriting the original value.

When starting a model, put any dimensions that you think you may use, even if they will need to be refined. This helps you to visualize the model; it also comes in handy later if there are any problems in the overall geometry.

You also need to define two other variables of global interest: frequency and velocity. A simple rule ensures that the model is precise enough for any situation for which it may be used:

> The frequency is defined as the highest frequency looked at in the simulation and the velocity as the lowest material velocity in the model.

You can then determine the minimum wavelength for the model and set the size of the elements accordingly.

PZFlex uses simple linear elements, i.e., squares and rectangles (2D) and hexahedrons (3D). It is important to use enough elements to accurately represent the object being modeled and to ensure that the forces calculated at the nodes are not overly generalized. Take, for example, three models of a circle:

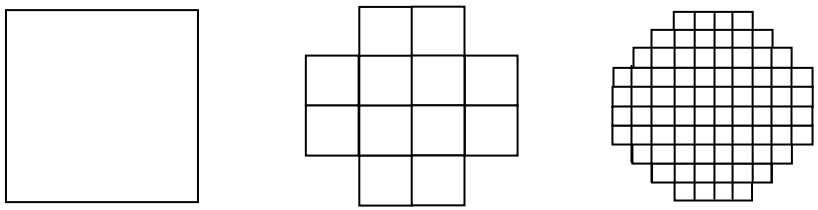


Figure 5

As you can see, the more elements the model contains, the more accurate is the representation of the object. If you use too many elements, however, you run the risk of building a model that takes too long to compute. The process of arriving at the number of elements required is called “meshing.”

> As a rule of thumb, you should use at least 12 elements per wavelength.

Fig. 6 shows the result of trying to model a simple sine wave across a block with different numbers of elements per wavelength.

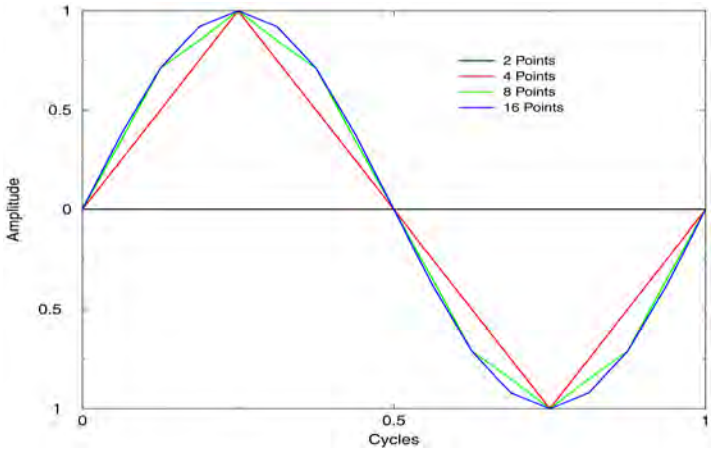


Figure 6

As the number of elements per wavelength increases, the resolution of the sine wave improves.

If you calculate the wavelength at the start of your input file, you can use that information in the meshing process. The code below specifies the velocity and frequency of interest and then calculates the wavelength:

```
c By specifying the wave velocity and frequency of interest
c the wavelength can be easily calculated

symb wavevel = 1600.                    /* Longitudinal wave velocity in material (m/s)
symb freqint = 150.e3                   /* Frequency of interest (Hz)
symb wavelgth = $wavevel / $freqint     /* Wavelength of sound in material (m)

c Model Discretization
c ================

c Determine how large each finite element will be
c by choosing a box size to give a specific
c number of elements per wavelength

symb znumelem = 15                      /* Number of elements per wavelength (>15)
symb zbox = $wavelgth / $znumelem       /* In the Z plane
symb xynumelem = 30                     /* Number of elements per wavelength (>15)
symb xybox = $wavelgth / $xynumelem     /* In the XâY plane
```

The second section of the code then determines how large each element should be in the x -y and z directions, based on the number of elements.

2. Forming the Model in the X-Y-Z Plane

To begin modeling, it is first necessary to convert the real-life object into a format that PZFlex can understand. To do this, we plot the points of the model, similarly to when drawing an object on graph paper. That is, we think of a grid centered on an axis at zero. We then build along the x and y axes for 2D models or the x-y-z axes for 3D models.

The first example in Fig. 7 is a simple rectangle. To build it, we need the points for the four corners. We define two points on the x axis and two points on the y axis:

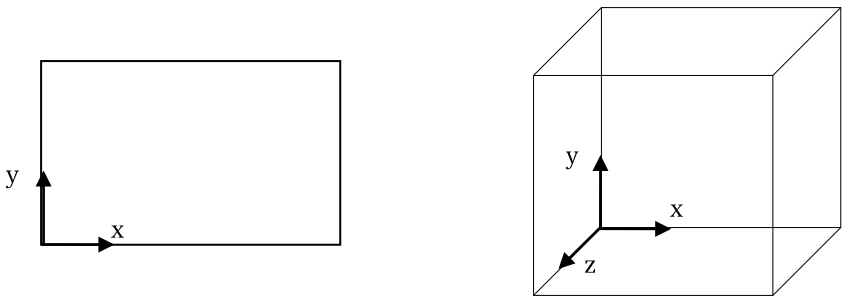


Figure 7

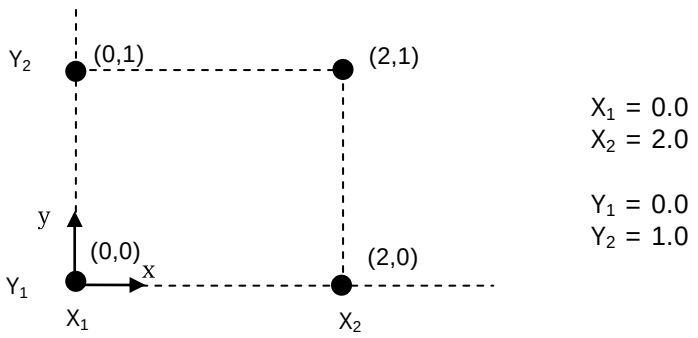


Figure 8

In Fig. 8, the four vertices (corners) of the rectangle are defined in Cartesian space. To define the model so PZFlex can understand it, we insert the following code into the input file:

```
c              BUILD THE COORDINATES SYSTEM
c
c X-direction
c ************

symb x1 = 0.0                    /* Start of model at position 0.
symb x2 = 2.0                    /* End of rectangle at 2

c Y-direction
c ************

symb y1 = 0.0                    /* Start of model at position 0.
symb y2 = 1.0                    /* End of rectangle at 2
```

Or, by referring to the predefined variable table, we can use global design variables to create a slightly more complex geometry. This is an example of how predefined variables can simplify the design process.

```
c            BUILD THE COORDINATES SYSTEM
c

c Now create variable to mark the âend pointsâ in the model

c X-direction
c ***********

symb x1 = 0.0                    /* Start of model at position 0.
symb x2 = $x1 + $polykerfxm          /* Add a polymer width in x-direction
symb x3 = $x2 + $cerwdthxm           /* Width of ceramic pillar in x direction
symb x4 = $x3 + $polykerfxm          /* Polmer kerf width in x direction
symb x5 = $x4 + $cerwdthxm           /* Width of ceramic pillar in x direction
symb x6 = $x5 + $polyedgexm          /* Width of outer polymer in x direction

c Y-Direction
c ***********

symb y1 = 0.0                    /* Start of model at position 0.
symb y2 = $y1 + ( 0.5 * $certhklm )          /* Thickness of transducer
```


This gives the following model geometry:

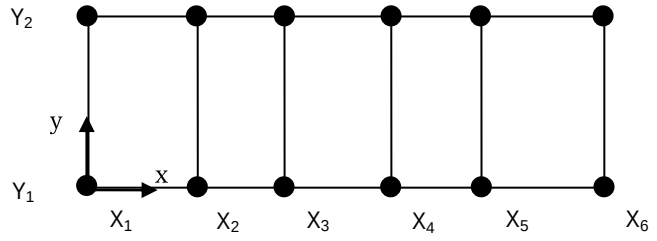


Figure 9

It is not necessary for a model to start at the zero point of the axis. It can start at any point, but for the sake of simplicity it is usually best to locate part of the model on the zero point to serve as a fixed reference point.

The next step in modeling a 3D shape is to add coordinates in the z direction. For a simple cube, we write:

```
c            BUILD THE COORDINATE SYSTEM
c
c X-direction
c **********

symb x1 = 0.0                    /* Start of model at position 0.
symb x2 = 1.0                    /* Width of cube = 1

c Y-direction
c **********

symb y1 = 0.0                    /* Start of model at position 0.
symb y2 = 1.0                    /* Depth of cube = 1

c Z-direction
c **********

symb z1 = 0.0                    /* Start of model at position 0.
symb z2 = 1.0                    /* Height of cube = 1
```


This gives the geometry for the following model:

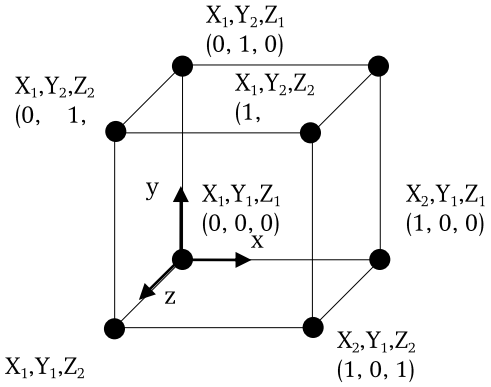


Figure 10

This demonstrates the process of thinking about the object that you intend to model and breaking it down into smaller, more manageable elements. Basically, you visualize the object in terms of points in space and catalogue the points in the input file.

PZFlex can also handle circular and other more complex geometries, but the important thing for now is to understand the basics of building a finite-element model.

Once you have defined the points in space, you must determine the number of elements that can fit into the model.

3. Defining the Elements in the I-J-K Plane—Meshing

You have now defined the geometry of the model in a simple Cartesian coordinate system, i.e., defined the points in space where there are boundaries or transitions between materials. The next step is to mesh the model, i.e, to determine how many nodes and elements fit into the geometry.

Fig. 2 showed the nodes at the vertices of the elements. These nodes transfer the force/stress/displacement/voltage and so on around the model. PZFlex works with a grid structure, which is referred to by integer node numbers.

The first step is to determine the size of the grid for each direction. The grid size is governed by the number of elements per wavelength, calculated above, i.e., “xybox” gave the step size of the elements for 15 elements in one wavelength of interest. This does not mean that there must be 15 elements in that direction, but that the wavelength in question must be covered by at least 15 elements.

For the 2D model in Fig. 8, define the node number points as:

```
c Transducer Internal Structure
c ------------------------------

symb i1 = 1                                     /* Set the first node in the "i" direction as 1
symb i2 = $i1 + nint ( ( $x2 - $x1 ) / $xybox ) /* Determine the number of "i" nodes based upon
                                                /* Model length and element box size
symb indgrd = $i2                               /* Create a variable to define last node in i

symb j1 = 1                                     /* Set the first node in the "j" direction as 1
symb j2 = $i1 + nint ( ( $y2 - $y1 ) / $xybox ) /* Determine the number of "j" nodes based upon

symb indgrd = $j2                               /* Create a variable to define last node in j

                                                /* Model length and element box size
```

As with the geometry, it is simplest to reference the first node point on each axis as 1, so that the node of the x-y origin is 1,1. Next determine the maximum number of node points in the x plane by working out the distance between points x2 and x1 and dividing

it by the size of the grid required (xybox). This may not be a whole number, but using the "nint" command returns an integer number. You then add this number to the previously defined node number, in this case i1, to determine the node number at this boundary.

Repeat this process in the y direction. In both the x and y directions, you have assigned a variable to specify the maximum node number. Just as you define the first node as 1, it is simplest to assign a name to the last node number.

Next enter the "grid" command. This converts the original x-y-z coordinates to i-j-k coordinates to determine the nodes and elements in the grid structure. This is done using:

```
grid $indgrd $jndgrd          /* Set size of grid
```

This forms a grid as large as the predefined variables "indgrd" and "jndgrd." You now should have a grid that looks like the one in Fig. 11, where indgrd = 9 and jndgrd = 7.

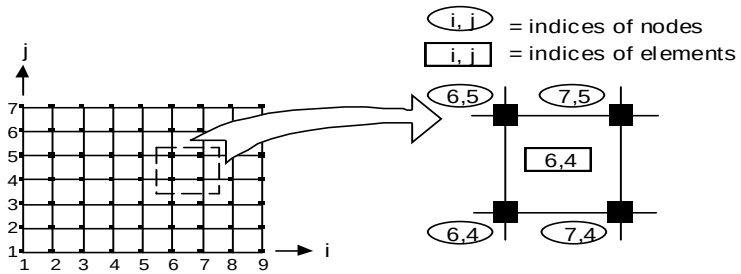


Figure 11

In addition to simple plane strain models in 2D, you can model a variety of other structures by using the constraint option after the grid-size definition. The constraint parameter specifies the type of constraint relations that apply to 1D and 2D models. It is left blank for 3D models.

Fig. 12 shows the constraint options for 2D models. They include plane strain, axisymmetric (about either the x or y axis), and sh-wave propagation, in which the only component of motion is in the z direction (normal to the plane of the model). As with the SH option for 1D, the 2D SH option is available only for linear elastic problems.

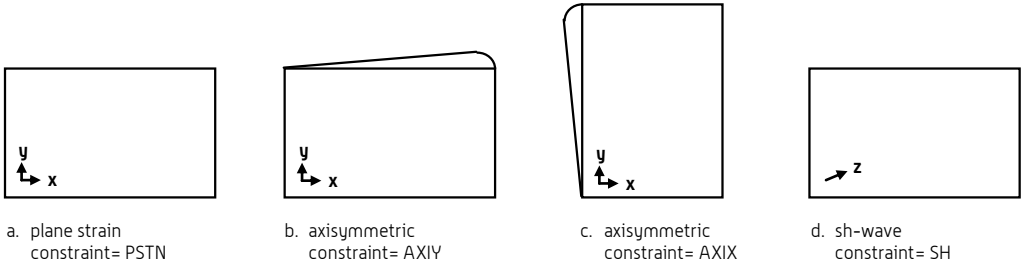


Figure 12—Constraint Options for 2D Models

For a 3D model of a cube, you can generate nodes and elements in the k direction to replace the model definitions in the z direction. This gives the following grid arrangement:

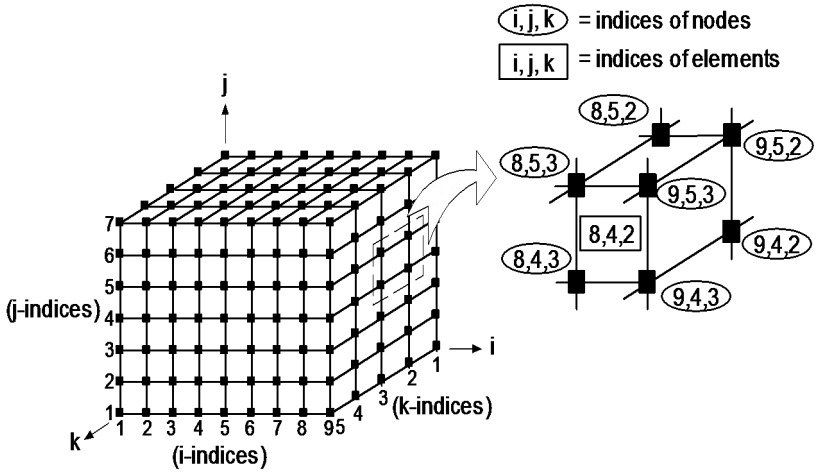


Figure 13

Although PZFlex knows that there is a grid structure, it does not know the position of the nodes with respect to the established x-y-z geometry. To tie the node positions to the geometry, use the “geom” command. Returning to the 2D model, define the structure as:

```
geom
     xcrd $x1 $x2 $i1 $i2

     ycrd $y1 $y2 $j1 $j2
```

Note that “geom” is a primary command; that is, it tells PZFlex that a set of subcommands will be defined after “geom” is recognized. These are indented from the first line of text to indicate to PZFlex that they are subcommands. The Command Reference, under “Command Definitions,” provides all the primary commands and associated subcommands.

After entering the secondary commands, enter the text string “end” to tell PZFlex that you have finished assigning commands within the “geom” structure.

The “xcrd” command is defined as:

xcrd xbegin xend ibegin iend ratio

For example,

xcrd 0.0 1.0 1 11

sets node 1 x-position to 0.0m and node 11 x-position to 1.0m and creates 9 equally spaced nodes between the two.

In this case, we haven’t used the“ratio” option, which can be used to set an element size relative to the previous size, to produce a non-even grid.

The commands “ycrd” and “zcrd” are based on the same principle, but are for definition in the y and z directions.

Below is a more complicated example of the grid system:

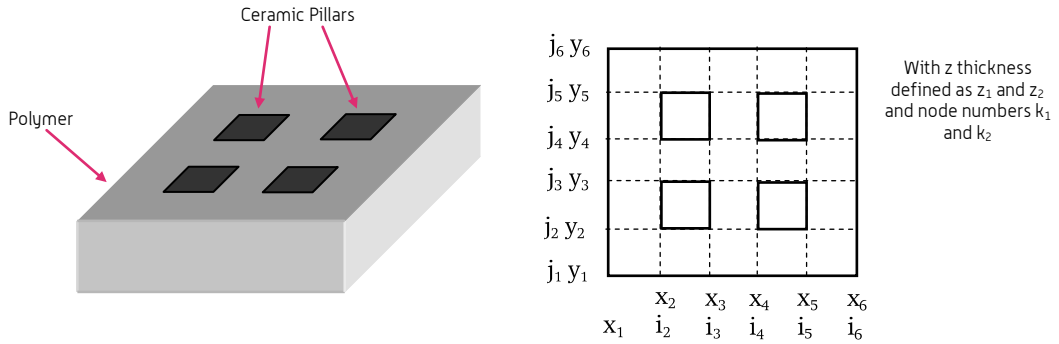


Figure 14

As before, assign the x coordinate positions x1 - x6 and the corresponding i coordinates node numbers i1 - i6. Likewise, the y and z coordinates are predefined as before.

In this case, define the geometry as:

```
geom
     xcrd $x1 $x2 $i1 $i2        /* X axis assignment
     xcrd $x2 $x3 $i2 $i3
     xcrd $x3 $x4 $i3 $i4
     xcrd $x4 $x5 $i4 $i5
     xcrd $x5 $x6 $i5 $i6

     ycrd $y1 $y2 $j1 $j2        /* Y axis assignment
     ycrd $y2 $y3 $j2 $j3
     ycrd $y3 $y4 $j3 $j4
     ycrd $y4 $y5 $j4 $j5
     ycrd $y5 $y6 $j5 $j6

     zcrd $z1 $z2 $k1 $k2        /* Z axis assignment
     end
```

Again, you must enter the “end” command, to tell PZFlex that you have completed the primary command. *(See New Commands—P160)*

4. Assigning Material Properties to the Model

This is the most important step for all finite-element models: not the assignment of the material properties to the model, but the values themselves. Good information about the materials is necessary if the results of your simulation are to correlate well with reality.

With materials such as mixed polymers, the properties change depending on how they are mixed and cured. Although it is possible in such cases to design your own materials file and tailor material characteristics to your needs, to do so requires a more advanced understanding of PZFlex; it should not be attempted by a novice. For further detail, see Appendix B.

PZFlex contains a file of most commonly used materials. To load this generic materials properties file into your materials properties file, use the “symb #read” command, for example:

```
symb #read c:\Program Files\WAI\Flex\PZFlex\Templates\matr.general  /* Load materials file
```

The location of the file depends on what system you are using. The file is typically found in `C:\Program Files\WAI\Flex\PZFlex\Templates` on Windows systems and in `/opt/WAI/Flex/PZFlex/Templates` on Linux systems. We strongly recommend that

you copy the file to your working directory and make any desired changes to this backup copy. That way, an uncorrupted copy remains in the original directory.

Once you have built up a database of available materials, you must select and assign them to the model.

Returning to the 2D model in Fig. 11, assign it as a block of stainless steel. Referring to the materials file, you see that "stst" is the name for this material. To assign the material, enter into the input file:

```
site
      regn stst 1 9 1 7     /* Assign stainlesss steel properties to area
      end
```

The primary "site" command tells PZFlex that you are about to assign material properties to the meshed model. The region or "regn" command follows:

regn <material name> ibegin iend jbegin jend kbegin kend

The previous command assigned a region of stainless steel from i = 1 to 9 and j = 1 to 7. Because the model is 2D, there are no k numbers.

Going to the 3D model in Fig. 13, for a PZT-5H ceramic block assign the material as:


```
site
     regn pzt5h 1 9 1 7 1 5     /* Form a 3D block of PZT-5H ceramic
     end
```

This forms a region of PZT-5H from i = 1 to 9, j = 1 to 7, and k = 1 to 5.

Fig. 14 is slightly more complex to mesh. There are two ways to assign materials to such a model:

1. Using the process described above, break the polymer section of the model into smaller regions and assign materials to them.

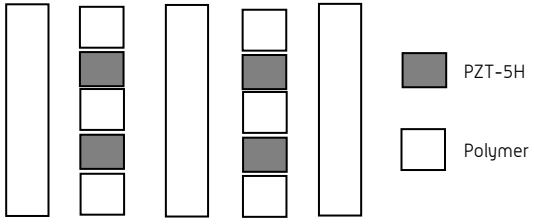


Figure 15

Assuming “poly” to be the name of the polymer in the material file and “pzt5h” to be the PZT-5H ceramic, assign the material properties as:

```
site

c Ceramic Pillars

     regn  pzt5h  $i2   $i3  $j2   $j3   $k2   $k3      /* Set ceramic block
     regn  pzt5h  $i2   $i3  $j4   $j5   $k2   $k3      /* Set ceramic block
     regn  pzt5h  $i4   $i5  $j4   $j5   $k2   $k3      /* Set ceramic block
     regn  pzt5h  $i4   $i5  $j2   $j3   $k2   $k3      /* Set ceramic block

c Polymer filler in rear composite

c The model is built from origin and working right along x-plane,

     regn  poly   $i1   $i2   $j1  $j6   $k1   $k2      /* Polymer block column 1
     regn  poly   $i2   $i3   $j1  $j2   $k1   $k2      /* Polymer block column 2 - row 1
     regn  poly   $i2   $i3   $j3  $j4   $k1   $k2      /* Polymer block column 2 - row 2
     regn  poly   $i2   $i3   $j5  $j6   $k1   $k2      /* Polymer block column 2 - row 3
     regn  poly   $i3   $i4   $j1  $j6   $k1   $k2      /* Polymer block column 3
     regn  poly   $i4   $i5   $j1  $j2   $k1   $k2      /* Polymer block column 4 - row 1
     regn  poly   $i4   $i5   $j3  $j4   $k1   $k2      /* Polymer block column 4 - row 2
     regn  poly   $i4   $i5   $j5  $j6   $k1   $k2      /* Polymer block column 4 - row 3
     regn  poly   $i5   $i6   $j1  $j6   $k1   $k2      /* Polymer block column 5
     end
```


2. Although an element can be assigned only one material property, PZFlex allows you to change the material by overwriting it. In this case, you can make the whole model polymer and reassign the sections you want as ceramic pillars:

```
site

c Assign whole model as polymer

     regn  poly   $i1   $i6   $j1   $j6   $k1   $k2     /* Make one large polymer block

c Reassign the elements you want to be ceramic

     regn  pzt5h   $i2   $i3   $j2   $j3   $k2   $k3        /* Set ceramic block
     regn  pzt5h   $i2   $i3   $j4   $j5   $k2   $k3        /* Set ceramic block
     regn  pzt5h   $i4   $i5   $j4   $j5   $k2   $k3        /* Set ceramic block
     regn  pzt5h   $i4   $i5   $j2   $j3   $k2   $k3        /* Set ceramic block
     end
```

This accomplishes the same thing as the first method, but simplifies the coding by reassigning the material properties from polymer to ceramic.

Further Examples

Not all elements need be a solid material. If you enter:

```
site
        regn mat1                   /* Assign all as material 1
        regn mat2 5 10 1 10         /* Assign lower half as material 2
        regn void 3 8 3 8           /* Remove region in center of block
        end
```

you form the following model:

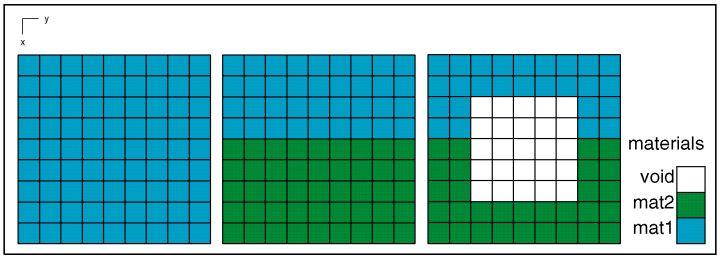


Figure 16

The material “void” is a generic command used to clear all properties from the defined elements. It is usually used to model air or space. The elements are still part of the model, but because they have no real properties, they act like a void and transfer no incident energy.

Complex Shapes

So far, you have looked only at the simple “regn” command. The Command Reference contains several other secondary commands for the “site” protocol.

To form a cylinder, for example, first use the above methods to build an appropriate mesh size, then use the “cyln” command to assign the material properties. The command is structured as:

cyln <mat name> <axis name> iaxis cbegin cend center1 center2 radbegin radend holbegin holend thetabeg thetaend ibegin iend jbegin jend kbegin kend

Although this may appear confusing initially, it is straightforward once you understand the variables.

You are already familiar with <mat name> or material name. <axis name> allows you to change the axis. Use “stnd” or “*” to keep the basic one. In PXFlex, “*” indicates that you do not want to change the default value. Other variables are:

iaxis	Longitudinal axis of the cylinder, i.e., the axis along which the cylinder is formed. Default is Z.
cbegin	Point on the axis, specified above, at which the cylinder begins.
cend	Point on the axis at which the cylinder ends.
center1, center2	Coordinates for the center of the cylinder.
radbegin	Outer radius at the bottom of the cylinder. Default is 0.
radend	Outer radius at the top of the cylinder. Default is radbegin.
holbegin	Inner radius at the bottom of the cylinder. Default is 0.
holend	Outer radius at the bottom of the cylinder. Default is holbegin.
thetabegin	Beginning angle for cylinder sweep. Default is 0.
thetaend	Final angle for cylinder sweep. Default is 360.
ibegin – kend	Indices of all nodes within the region of interest. Default condition is to encompass all the nodes.

The abundance of variables gives PZFlex great flexibility (hence the name). To create a simple cylinder in the z direction (on the original global axi) with a radius of 1.0m, enter:

```
site
     regn void
     cyln mat1 stnd z $z1 $z2 0. 0. 1. 1. 0. 0. 0. 360. $i1 $i2 $j1 $j2 $k1 $k2
     end
```

The first step is to void the entire region so that all elements have material properties. It is not necessary to include “$i1 $i2…” when applying the command to the entire node area. It is included above for completeness. The predefined mesh size is a 1m cube, split into 30 x 30 x 30 elements. Fig. 17 shows the materials model:

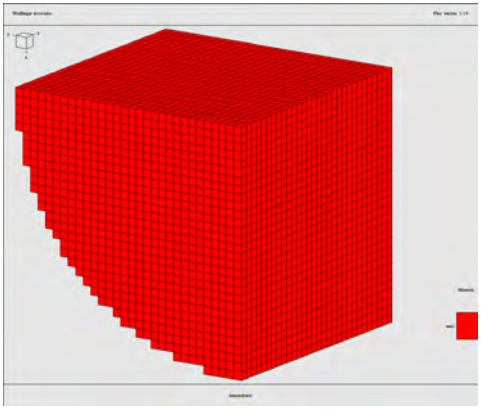


Figure 17

By default, void elements are not shown. The reason only a quarter of the cylinder is shown is that elements are built only in the positive x-y plane (Fig. 18a). To model an entire cylinder, the meshed area must encompass the entire sweep (Fig. 18b):

Adding elements in the negative x and y planes and entering the same “cyln” command

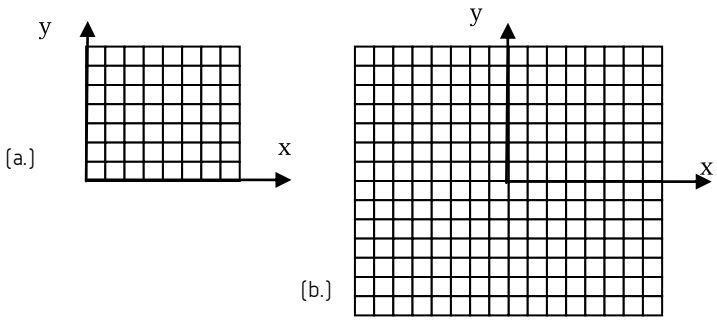


Figure 18

produces the full cylindrical volume (Fig. 19). The axis origin is located in the center of the cylinder.

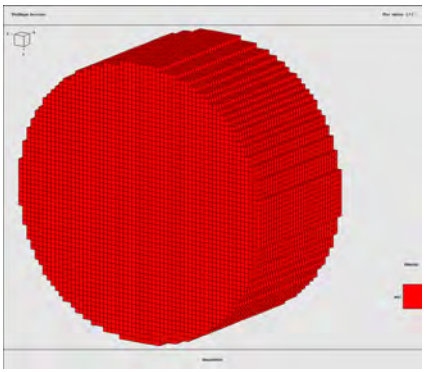


Figure 19

Consider what happens when you alter the radius at the cylinder top and bottom. Using the same command, we change the values for “radbegin” and “radend.” The remaining “cyln” examples assume a full element array on both the negative and positive x-y planes.


```
site
     regn void
     cyln mat1 stnd z $z1 $z2 0. 0. 1. 0.8 0. 0. 0. 360. $i1 $i2 $j1 $j2 $k1 $k2
     end
```

This gives a tapered cylinder:

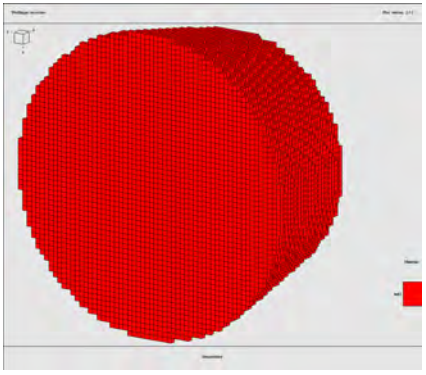


Figure 20

Reverting to a nontapered cylinder (for simplicity), you can turn the cylinder into a tube by increasing the values of “holbegin” and “holend” to 0.7m:

```
site
     regn void
     cyln mat1 stnd z $z1 $z2 0. 0. 1. 1. 0.7 0.7 0. 360. $i1 $i2 $j1 $j2 $k1 $k2
     end
```

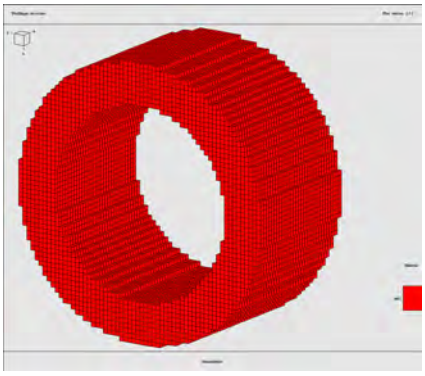


Figure 21

Finally, you can vary the extent of the cylinder by changing the sweep angle. Because PZFlex is based on the grid system, the angular-sweep command imposes a cylindrical sweep over the square geometry, doing a best fit. To create a pie-shaped slice, for example, we enter the following angular-sweep command:

```
site
     regn void
     cyln mat1 stnd z $z1 $z2 0. 0. 1. 1. 0. 0. 0. 45. $i1 $i2 $j1 $j2 $k1 $k2
     end
```

Although these are 3D models, the “cyln” command can be used to define curved geometries in 2D. The only difference is that during the meshing phase we designate a grid only in the x-y plane. It is still necessary, however, to give the cylinder a finite distance in the z plane to allow it to sweep around. We use the following command:

```
site
     regn void
     cyln mat1 stnd z -1. 1. 0. 0. 1. 1. 0. 0. 0. 90. $i1 $i2 $j1 $j2
     end
```

This gives:

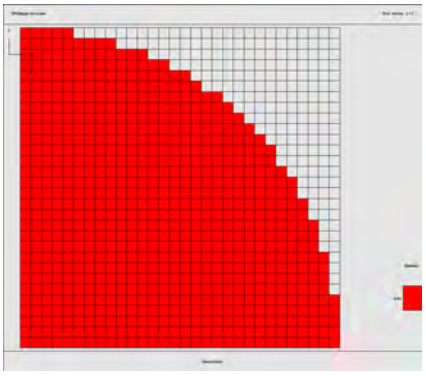


Figure 22

This 2D model shows the void elements (due to the operation of the graphics engine). There is no need to specify k indices in the command line, as the model is only on the i-j plane.

Finally, look at the “center1” and “center2” commands. These shift the location of the cylinder center; “center1” defines the offset on the x axis and “center2” the offset on the y axis.

Shifting the cylinder center along the axis while keeping everything else the same generates the material area in Figure 23:

```
site
     regn void
     cyln mat1 stnd z -1. 1. 0.2 0. 1. 1. 0. 0. 0. 90. $i1 $i2 $j1 $j2
     end
```

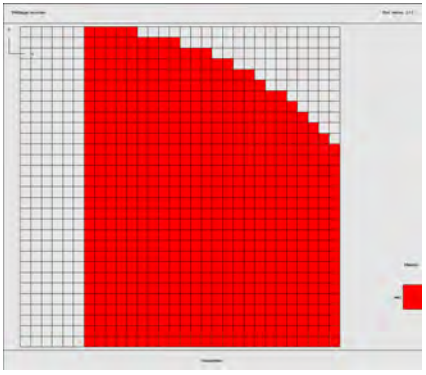


Figure 23

The shift in origin and trimming of the curve are due to the model's having no more elements on that direction. The versatility of this command is that it can be used to create complex and otherwise time-consuming geometries using only "regn" commands, e.g., a sloping block. Using the "cyln" command and the feature described at the start of the "More Examples" section, you can define a material block and then subtract elements from the block using the "cyln" command and assigning a void material. First assign a region of material to cover the element array. Then assign a further point in the x direction as the cylinder origin sweeps a void cylinder from 0° to 135°:

```
site
     regn mat1
     cyln void stnd z -1. 1. 1. 0. 2. 2. 0. 0. 0. 135. $i1 $i2 $j1 $j2
     end
```

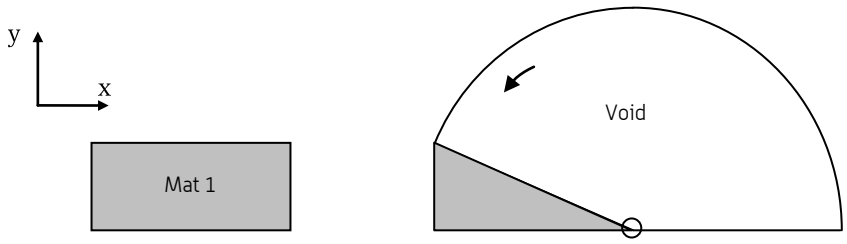


Figure 24

Other commands can be used to create elliptical and spherical shapes, as well as diagonal fillets such as that in Fig. 24. With PZFlex, it is possible to model complex geometries with little difficulty.

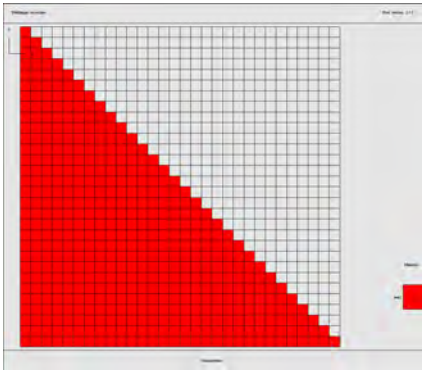


Figure 25

5. Viewing the Model

Viewing a graphic image of the model allows you to confirm that it has been built as you intended and to identify any problems with the input file. The first step is to enter the "grph" or graph primary command. This produces a visual representation of the model on the screen. The next step is to plot the elements that have been assigned material properties:

```
grph                    /* Enter graphing function
     plot matr          /* Plot all materials
     end                /* exit
```

You should now see a graphic representation of the model on the screen.

A range of commands is available within the "grph" function. Although most of them are used at the post-processing stage, a few are relevant here. A basic one is titling the graphic, using the "ttl" command:

```
grph                                /* Enter graphing function
     plot  matr                     /* Plot all materials
     ttl Plot of materials model
                                    /* Assigns "Plot of materials model" as title
     end                            /* exit
```


This titles the graphic window "plot of materials model." The "ttl" function supports titles of up to 200 characters.

When building models with unusual shapes, it can be difficult to see different sections of the model from the static view plotted from the "plot matr" command. Below are the defaults for the viewing window for 2D and 3D models:

2D orientation 3D orientation

Figure 26

To view the model from a different angle, use the "eye" command, which repositions the virtual eye that "sees" the model. Take, for example, the shape below:

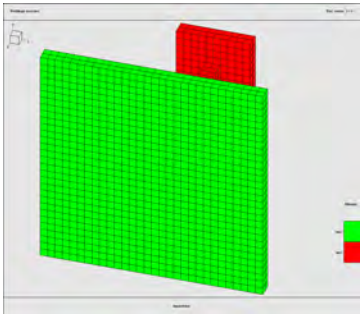


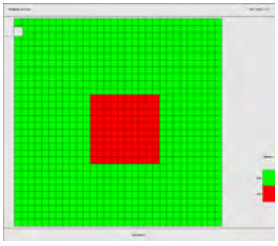
Figure 27

In the default image generated by the “plot” command, the model comprises two materials, and the rear of the model is obscured by the large plate structure in front. Using the “eye” command, rotate the model to view it from a different direction:

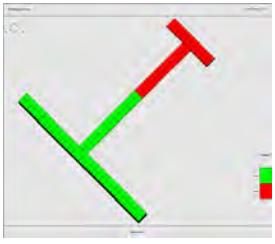
```
grph                                    /* Enter graphing function
    eye <xpoint> <ypoint> <zpoint>      /* Enter integer values to move eye point
    plot matr                           /* Plot all materials
    end                                 /* exit
```


Below are four “eye” positions and their corresponding commands:

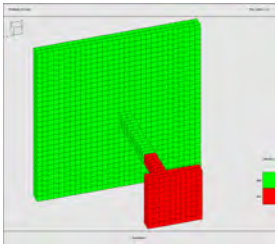
(a.) eye 1. 0. 0. Side View



(b.) eye 0. 0. -1. End View



(c.) eye 0.01 1.01 0.01 Plan



(d.) eye 1. 2. -3. Isometric View from Behind

Figure 28

The values entered for the plan view are slightly more complex. Because the vertical point of the model is defined as the y plane, if the “eye-point” were set as the same, the drawing package would have difficulty resolving the two. To compensate, we shift all the “eye” values slightly.

It is helpful to see all the views simultaneously rather than have to reset the “eye-point” and replot the material. To do this, define the desired number of graphical windows using the “nvew” command. The “nvew” command allows for a wide variety of window configurations:

```
grph
    nvew 4 1            /* Create a display window with four 'sub windows'
    eye 1. 0. 0.        /* Define side view
    plot matr           /* Plot all elements, with a different color for each material
    eye 0. 0. 1.        /* Define rear view
    plot matr           /* Plot all elements, with a different color for each material
    eye 0.01 1.01 0.01  /* Define top down view
    plot matr           /* Plot all elements, with a different color for each material
    eye 1. 2. -3.       /* Define rear isometric
    plot matr           /* Plot all elements, with a different color for each material
    end
```


Here you see all four views from Fig. 28 at once:

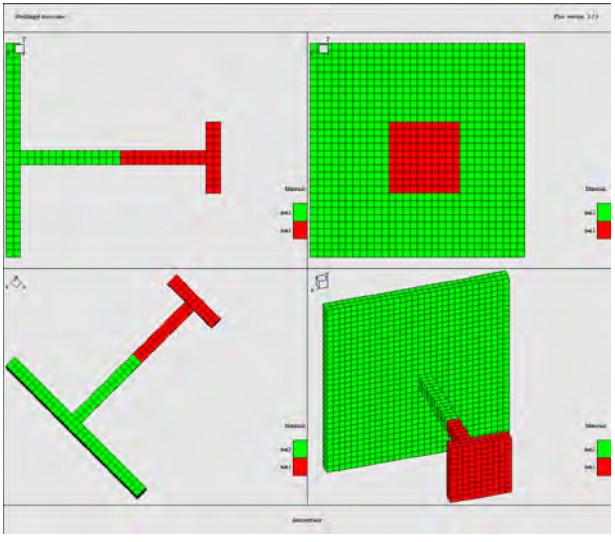


Figure 29

To export the images to a TIFF format for use in reports and manuals, enter the following two commands:

```
grph
    set imag tiff    /* Set the output format as tiff file
    plot matr        /* Plot material file
    imag             /* Export to file
    end
```

The second line informs PZFlex that when “imag” is called, the output file goes to a TIFF format graphic. The “imag” command should be used only after you have plotted the image in the graphics window, as it dumps the graphic output to the TIFF file. As each subsequent “imag” command is entered, the TIFF file is stored with an incremental number, so there is no need to overwrite the previous image file.

Useful Hint

PZFlex runs the input file as designed by you, the user; i.e., the text is read into PZFlex and acted upon. If at any point you wish to pause the inputting of the file, enter a “t” or “term” command:


```
grph
    plot matr    /* Plot material file
    end
```

This stops the reading of the input file, giving you access to the command line again. If the angle of the view is incorrect, for example, you can use the pause to redefine the eye-point. Or you can check the print file to see if the outputted variables are correct. To restart the input file, enter “t” again and the program resumes at the point where it was in the input file. You can also enter “stop” to terminate the run.

Color Tables

There are seven predefined color tables for plotting, Tables 1 through 6 and Table 8. Table 7 is reserved as a user-definable color table. Fig. 30 shows the color tables used for most color graphics. Typically, a spectrally graded color table such as Table 2 or Table 6 is used to display field data, whereas a more varied color sequence such as Table 4 or Table 5 is used to plot materials. Table 3, a gray scale, is useful for reports that will be reproduced in black and white. The color table preference can be changed at any point in the input stream using the “grph colr” command.

With PostScript image files, the colors depend on the state of the PostScript color option (set by the “grph set colr” command). If the PostScript color option is ON, the color image displayed on the screen is recorded in the PostScript output file. If the color option is OFF, a gray-scale table is saved in the PostScript image file.

PZFlex maintains two color tables for plotting, the material color table and the data color table. The material color table displays the material assigned to grid elements and any other data plots that are not field variable. (Note: Review uses the material color table to determine what color to assign to curves plotted on x-y plots.) The data color table plots all field-variable data. Unless otherwise specified in the defaults file, the default color table is Table 4 for material plots and Table 2 for field-variable plots. PXFlex automatically chooses the appropriate color table to use for plotting data when the “plot” subcommand is entered. You can change the color table for plotting materials and field-variable plots by using the “grph colr tabl” command, specifying either the data or matr option.

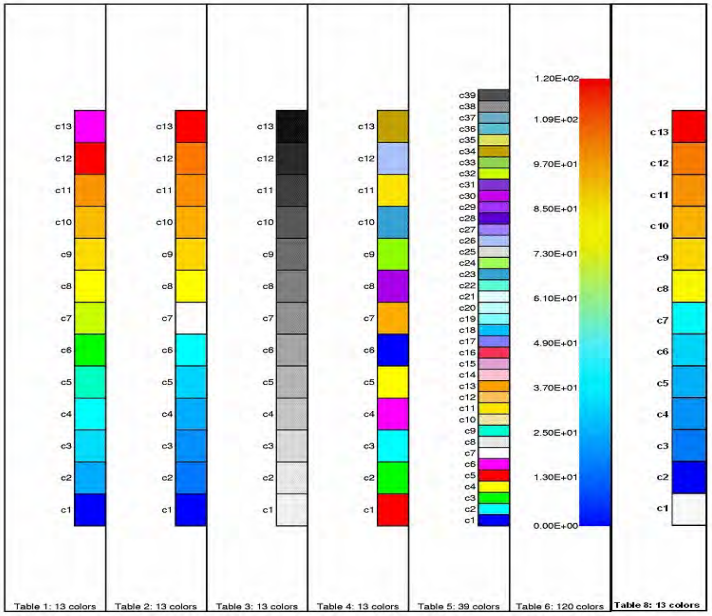


Figure 30 —Principal Color Tables Used for Graphic Display.

6. Applying Boundaries, Loads and Electric Fields

Once you have defined the model geometry and assigned the material properties, it is time to apply boundary conditions and loads, or driving functions, to the model and see how it reacts.

Boundary conditions

It is sometimes useful to apply boundary conditions to simplify the model. Before setting boundary conditions, you must select the nodes. PZFlex allows you to set the following conditions:

symm	symmetry, no motion across plane
fixd	all nodes fixed position, no motion
absr	absorbing boundary, incident waves absorbed
vel	prescribed velocity, model load

It is best to simplify the model as much as possible. If you are modeling a symmetrical object and the same conditions are applied to both sides of the model, there is no reason to model the entire structure. It is more efficient to model half (or less) of the structure and achieve the same results for a fraction of the computing time.

There two ways to do this: by adding symmetry or (in the case of an ultrasound device) by clamping the nodes to simulate the transducer's being in a holder.

1. Select the side of the model to which you wish to apply the symmetry boundary, using the "side" subcommand. PZFlex defines the sides of the model as shown in Fig. 31, where the box defines the entire grid geometry. Selecting Side 2 selects all the nodes on the maximum i-value of the geometry.

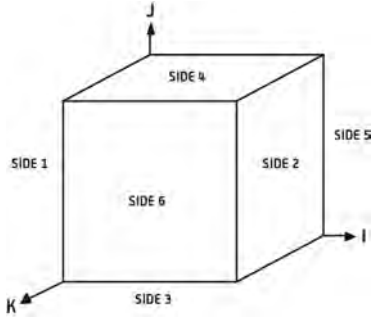


Figure 31

(For 2D models, only Sides 1 through 4 are relevant.) To apply symmetry to a model using the "side" subcommand:

```
boun                    /* Enter boundary commands function
    side 1 symm         /* Apply symmetrical boundary on side 1
    end
```

2. Unless otherwise specified, all nodes on boundaries are assigned as free, or unconstrained. To apply an absorbing boundary to the highlighted section of Fig. 32, it is necessary to specify nodes; using "side" would not work. This is a technically more advanced method, which we will not discuss in detail here.

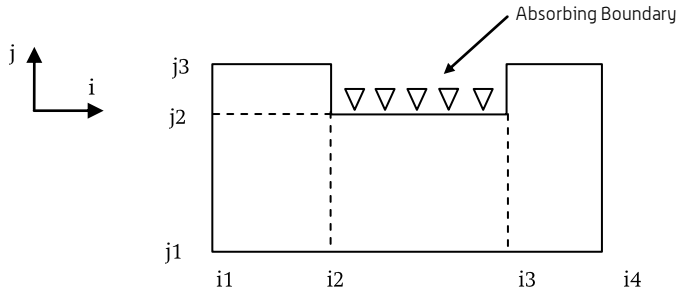


Figure 32

Note that the sides can also be referred to as side '*xmin/imin' (side 1)* which refers to the model edge/plane where the X or I coordinates are constant. This is often simpler than trying to remember which side of the model is *'side 1'*.

Driving Functions

PZFlex uses a time domain, or transient, solver. The model is subjected to a load or loads over time, and PZFlex provides the resultant time-history information. With a single calculation you can use that time-based information to determine such phenomena as electrical impedance, pressure wave output, and mode shapes. Some FEA programs use harmonic, or frequency-based, methods that require multiple calculations for the same frequency range.

There is an art to running models efficiently. Generally, it is best initially to run a smaller and quicker, less well-meshed model to determine the desired simulation period and then rerun the model with the more refined meshing characteristics. This allows you to optimize the model parameters and run the simulation only for as long as necessary. The shorter the running time, the shorter the processing time and the quicker you get results.

In PZFlex, the aim is usually to determine a common parameter such as a device's operational impedance, force output, or acoustic emission pattern (or beam plot). As such, 90% of the time it is necessary to apply only a simple single-cycle sine wave at the frequency of interest (the same frequency of interest assigned at the start of the input file and described in Section 1).

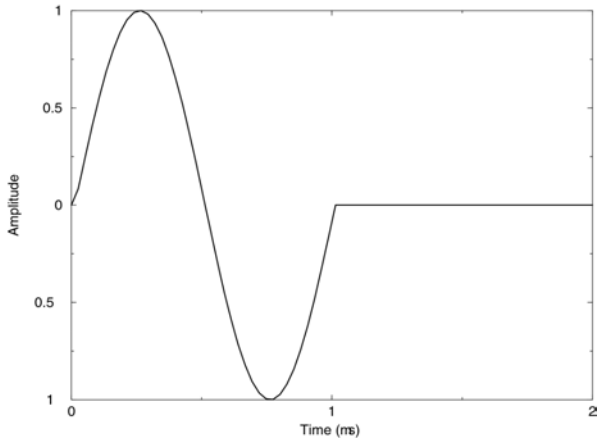


Figure 33

Choosing the Frequency

Usually when designing a new piezoelectric transducer, you will use 1D techniques to determine the likely frequency of operation and then use that information to tailor the simulations. To examine the response of the device over a range of frequencies, however, there must be energy at those frequencies. Using a fast Fourier transform (FFT) on the 1MHz signal in Fig. 33, you see the energy across the frequency spectra:

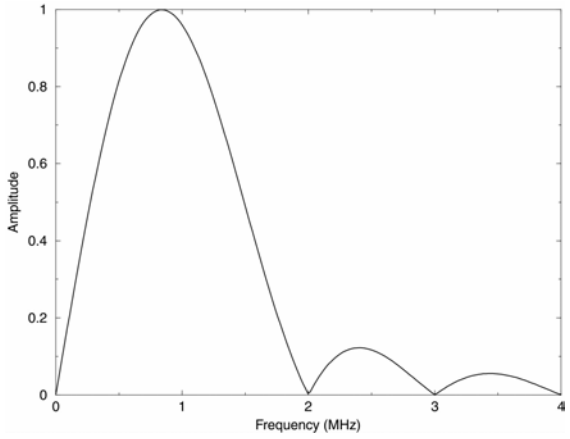


Figure 34

Little or no energy is produced at the multiples of the frequency of interest, in this case 2, 3, and 4MHz. When designing models and assigning drive frequencies, you must make sure that the model is excited across all frequencies of interest.

To generate this sinusoid function, use the “func” command. Only one “func” command can be used per job. To generate the single-cycle sinusoid in Fig. 33, use the following command:

For a fuller description of the “func” command and its variables, see the Command Reference. For this frequency, “$freqint” replaces 1.e6.

After defining the driving function for the model, you must specify what the driving function is to be applied to, and how.

Pressure Loads

Although PZFlex is often used to model piezoelectric devices, sometimes you need to know only how the wave propagates through a material. In that case, it is not necessary to build a complex model that incorporates the transducer structure. You can simply model


```
func sine 1.e6 1. 0. 1.      /* 1MHz single cycle sinusoid
```

the material, apply a direct pressure load to it, and view the wave propagation. (The Annotated Examples section contains examples.)

After assigning the material geometry and meshing the model, apply the pressure load using the “plod” function. The primary function “plod” and its series of subcommands determine where and how the pressure is applied.

Here you apply a predefined pressure load to the model:

```
plod                                /* Enter plod function
    pdef pld1 func                  /* What the function will be (use previously defined "func")
    vctr vct1 1. 0. 0.              /* What direction will it be applied (in vector x, y, z format)
    sdef pld1 vct1 $i1 $i1 $j1 $j2  /* Which elements it will be applied to
                                    /* This is the model lefthand edge
    end
```

As stated above, “pdef” defines the pressure function—in this case, using the single sine wave defined as “func” (see the previous section on driving functions). Use the “vctr” subcommand to tell the model in which axis you want the pressure function to propagate and give it a name for easy reference (vct1, in this case). Say that the propagation will be

in the x direction. Finally, define the surface upon which the load is to be applied. The "sdef" subcommand allows you to apply the defined drive function—pld1, along the given vector path—vctr to the nodes specified by the "sdef" command. As with all commands requiring node number definition, the standard terms apply, namely, the start and end node indices in each of the model ijk dimensions. In this case, the code segment applies to the model a 1MHz, single-cycle sine wave traveling in the x direction and originating at the specified nodes.

Electric Field Loads

PZFlex can also model the displacements that occur from the application of electric fields to piezoelectric materials. Because an electric field window vastly increases computation time, it should be used only when it is required and only around those sections of the model that are affected by the electric field. You can assign more than one window, as when simulating the electric fields on both a transmitter and receiver. Fig. 36 shows the application of an electric field around piezoelectric pillars in an epoxy matrix.

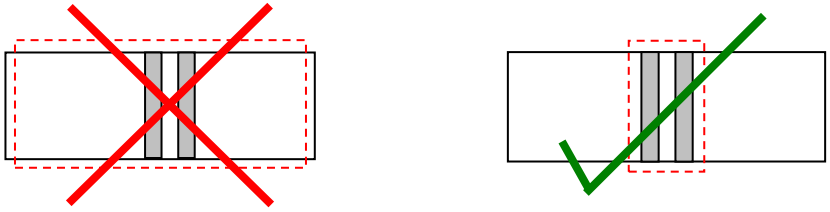


Figure 35

You can assign as many electrodes as you wish. Although you can name the electrodes, for ease of reference, PZFlex numbers them sequentially as they are defined, beginning with "1." (This will be important later.)

The three steps are:

1. Define electric window
2. Specify electrodes
3. Specify voltages on electrodes

First use the "piez" function to define the electric field window and subsequent operations:

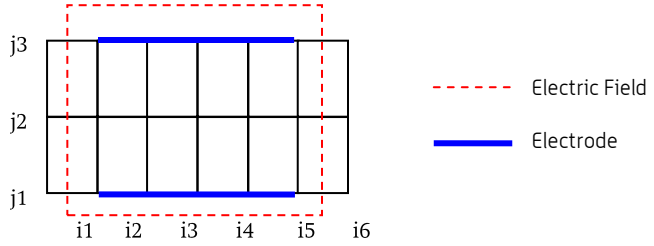


Figure 36

In this case, you need to apply the electric field over the eight central elements and apply electrodes to the top and bottom over these elements. To apply the field, use the following code segment:

```
c Apply an electric window

piez

c Specify piezoelectic calculation window

    wndo  $i2   $i6   $j1   $j3         /* Electrical window over pillars

c Specify electrodes

    defn  top                /* Top electrode
    node  $i2   $i6   $j3   $j3
    defn  bot                /* Bottom electrode
    node  $i2   $6    $j1   $j1

c Specify voltages on electrodes

    bc  top  volt  func         /* Apply voltage to middle electrode
    bc  bot  grnd
    end
```

Define the window over the elements, then specify the nodes on the top of the structure as one electrode and repeat this for the nodes on the bottom. Then assign the boundary conditions (“bc”) to the top and bottom. In this case, apply the predefined single-cycle sine wave function as a voltage to the top electrode and set the bottom electrode as a ground plane.

7. Selecting Model Outputs & Defining Model Runtime

As a time-domain solver, PZFlex generates its results as a set of time histories. A substantial range of time histories is available, including voltage, charge, current, pressure, displacement (in all directions), and velocities. Once you have defined the model geometry and initial conditions, you must specify what results you are seeking.

To use memory most efficiently, by default, PZFlex calculates only the minimum required data set, typically velocities. If you need other properties, such as displacements, stresses, strains, or pressure, you must request them using the "calc" command:

```
calc
    pres        /* Calculate pressure
    disp        /* Calculate displacement
    strn        /* Calculate strain
    strs        /* Calculate stress
    end
```

PZFlex has an internal data manager that organizes all program variables into named data groups. These provide access to the data using the various output options. The name of a data group can be up to four characters long. A data group can be thought of as a triple-subscripted data array. For example, the members of the data group "data" can be identified by its ijk indices, i.e. data (i,j,k). To request output for a particular problem variable, specify the name of the data group to which it belongs and the ijk indices that identify the item within the group. For a 2D data group, for example, only the i and j indices are required to identify an item within the data group. The group can be considered to have a k index always equal to 1. Likewise, 1D data groups are considered to have both j and k indices equal to 1.

Using the "pout" function, you can highlight the results that you want to record. You will use mainly the "hist" subcommand to define the time histories you want. The "hist" command is defined as:

hist <dataname> ibegin iend jbegin jend kbegin kend

where the <dataname> is the name or names of the data arrays you wish to store. For example, "zdsp" is the displacement in the z direction, and "pres" is the pressure front. Time histories can also be output directly in a MATLAB type format as well. See the table at the end of this section for a complete list of available datanames and the required "calc" option. The code segment below shows a common time history definition for the piezoelectric model in Fig. 32.

```
c Which time histories to save(number)?

pout
    hist func                        /* Input function           (1)
    hist pize 1 2 1 1 2 1            /* Volt/charge on ceramic   (2 - 5)
    hist pres $i4 $i4 1 $j3 $j3 1    /* Pressure output at top   (6)
    hist ydsp $i4 $i4 1 $j1 $j1 1    /* Y displacement on bottom (7)
    end
```

This says that you want to store the input function in array 1 of the time history (because it is the first time history specified).

To make it easier to store the electrode time histories, the “histname” command allows you to select a specific electrode by its previously defined name. For the example above, the command is:

```
c Which time histories to save?

pout
    histname electrode vq top   /* Voltage and charge from electrode "top"
    end
```


Replace the electrode name with “all” if you want to capture the voltage and charge for all the specified electrodes. You can select different combinations of voltage, charge, and current by using the options v, q, and i.

```
c Which time histories to save?

pout
    hist func                        /* Input function           (1)
    hist pres $i4 $i4 1 $j3 $j3 1    /* Pressure output at top   (2)
    hist ydsp $i4 $i4 1 $j1 $j1 1    /* Y displacement on bottom (3)
    end
```

Returning to Fig. 36, the line following the electrode definition selects the pressure output on the top/middle node of the model and outputs it to array 6, while the final line selects the displacement in the y direction of the node at the bottom/middle of the model in the file array 7. As each time history is specified, it forms a new array in the overall time history file, which is numbered sequentially.

Such selections can provide the pressure output at a given point in the transducer field or show the deformation of a node in a structure. Specifying time histories for individual nodes gives direct access to the results at that point.

Note that the format of the time histories and dataout arrays can be changed to support operation with MATLAB using the FORM subcommand:

The FORM subcommand requires you to enter the format with the arrays will be saved as (MATLAB) and an ID-tag to reference the data (M1). When using within the DATA command, an extra parameter is required to specify either and input (IN) or output file (OUT).

```
pout
    form MATLAB M1
    hist func /*....
    end
```

```
data
    form out MATLAB M1
    out pres /*....
    end
```

The following table shows the different arrays that it is possible to calculate and store.

PZFLEX data array	*Indices*	*CALC command required*	*Description*
			Basics
xcrd	i,j.k	<done by default>	Original X-Coordinate (Nodal)
ycrd	i,j.k	<done by default>	Original Y-Coordinate (Nodal)
zcrd	i,j.k	<done by default>	Original Z-Coordinate (Nodal)
xdsp	i,j.k	disp	X-Displacement (Nodal)
ydsp	i,j.k	disp	Y-Displacement (Nodal)
zdsp	i,j.k	disp	Z-Displacement (Nodal)
xvel	i,j.k	<done by default>	X-Velocity (Nodal)
yvel	i,j.k	<done by default>	Y-Velocity (Nodal)
zvel	i,j.k	<done by default>	Z-Velocity (Nodal)
			OTHER
velm	i,j.k	velm	Velocity Magnitude
tmas	i,j.k	<done by default>	Current Time step / nodal mass
			ELEMENTAL STRESS
sgxx	i,j.k	strs	XX-Stress
sgyy	i,j.k	strs	YY-Stress
sgzz	i,j.k	strs	ZZ-Stress
sgxy	i,j.k	strs	XY-Stress
sgyz	i,j.k	strs	YZ-Stress
sgxz	i,j.k	strs	XZ-Stress
			ELEMENTAL STRAIN
epxx	i,j.k	strn	XX-Strain
epyy	i,j.k	strn	YY-Strain
epzz	i,j.k	strn	ZZ-Strain
epxy	i,j.k	strn	XY-Strain
epyz	i,j.k	strn	YZ-Strain
epxz	i,j.k	strn	XZ-Strain
			ELEMENTAL MAXIMUM STRAIN
mnxx	i,j.k	maxe	Minimum XX-Strain
mnyy	i,j.k	maxe	Minimum YY-Strain
mnzz	i,j.k	maxe	Minimum ZZ-Strain
mnxy	i,j.k	maxe	Minimum XY-Strain
mnyz	i,j.k	maxe	Minimum YZ-Strain
mnxz	i,j.k	maxe	Minimum XZ-Strain
mxxx	i,j.k	maxe	Maximum XX-Strain
mxyy	i,j.k	maxe	Maximum YY-Strain
mxzz	i,j.k	maxe	Maximum ZZ-Strain
mxxy	i,j.k	maxe	Maximum XY-Strain
mxyz	i,j.k	maxe	Maximum YZ-Strain
mxxz	i,j.k	maxe	Maximum XZ-Strain

PZFLEX data array	*Indices*	*CALC command required*	*Description*
			MISC. ELEMENTAL DATA
pres	i,j.k	pres	Elemental Pressure
aprs	i,j.k	pres acoustic	Elemental Acoustic Pressure
sj2p	i,j.k	sj2p	Shear component of stresses
loss	i,j.k	loss	Accumulated Energy dissipation
xaint	i,j.k	aintn	Acoustic Intensity
yaint	i,j.k	aintn	Acoustic Intensity
zaint	i,j.k	aintn	Acoustic Intensity
maint	i,j.k	aintn	Acoustic Intensity
volr	i,j.k	volr	Volumetric strain rate
xcrl	i,j.k	volr	Displacement field curl vector (X)
ycrl	i,j.k	volr	Displacement field curl vector (Y)
zcrl	i,j.k	volr	Displacement field curl vector (Z)
evol	i,j.k	curl	Volumetric Strain
			PIEZOELECTRIC DATA
potn	i,j.k		Nodal electric potential field (voltage)
qfrc	i,j.k		Nodal electric charge
ex	i,j.k	elec	Elemental electric field X-Component
ey	i,j.k	elec	Elemental electric field Y-Component
ez	i,j.k	elec	Elemental electric field Z-Component
pize			Electrodes
			CIRCUIT DATA
crtv	i = circuit node #		Voltage at circuit node
crtq	i = circuit node #		Charge at circuit node
			DYNAMIC RELAXATION DATA
drlx	i = dataID[a]		Time varying dynamic relaxation
			PULSE/ECHO EXTRAPOLATION
echo			Extrapolated pressure at given point
echp	i = time step		Full time history of “echo”
			OTHER
timsp			Current model timestep
			RIGID SUBSTRUCTURE DATA
rgfr	i = comp[b], j = #		Forces on rigid structure
rgvl	i = comp[c], j = #		Velocities of rigid structure
rgds	i = comp[d], j = #		Displacements of rigid structure

a. 1 = critically damped angular frequency (for "auto" option only), 7 = mass-weighted velocity norm, 8 = prior peak value of drlx(7), 9 = moving average of drlx(7), 10 = drlx(9)/drlx(8), 11 = time of peak drlx (7)
b. 1 = x-force, 2 = y-force, 3 = z-force, 4 = x-rot’l force, 5 = y-rot’l force, 6 = z-rot’l force
c. 1 = x-velocity, 2 = y-velocity, 3 = z-velocity, 4 = x-rot’l vel, 5 = y-rot’l vel, 6 = z-rot’l vel
d. 1 = x-disp, 2 = y-disp, 3 = z-disp, 4 = x-rot’l disp, 5 = y-rot’l disp, 6 = z-rot’l disp

8. Running the Model

Before running the model, you should enter the “prcs” command. In preparation for processing the time-step information, it computes the model time step for stability and opens the necessary arrays for field-data storage. Although the “prcs” command is automatically entered with the first “exec” command if it has not already been so, it is good practice to enter it before beginning. Once it is entered, no alterations to model geometry, material properties, and so on are permitted.

The following code segment shows the use of the “prcs” command:

```
c The model has now been defined. The "prcs" step must be
c used to calculate model timestep, etc. before execution

prcs                                    /* Pre-processor command

c Model time step

symb #get { step } timestep             /* Retrieve calculated time step

c How long to run for?

symb simtime = 1.e-3                    /* Simulate for 1ms (user-defined)

c Total number of time steps to run

symb nruns = nint ( $simtime / $step )  /* Calculate number of models steps required

c Run appropriate number of timesteps

exec $nruns                             /* Run model
```


After entering the “prcs” command, use the “symb #get” and “symb” commands to calculate the model step size and run the analysis. Define how long the model should run and determine the number of steps required to complete the simulation. Because PZFlex is a time-domain solver, once executed, the model calculates its response for each time step consecutively. This is done using the “exec” command, that is, “exec 1” executes one time step in the simulation and “exec 100” 100 time steps, when the time step has previously been calculated in the “prcs” step.

The model runs for the defined number of time steps and calculates the time histories, storing them in the same directory, in a PZFlex history file named: `<jobname>.flxhst`.

Model Optimization

Once the model is ready to run, the next step is to select the simulation runtime. To determine how long a model should run, consider your governing criteria. To examine a

wave interaction with a target in a solid material, for example, requires enough time for the wave to propagate through the material and reach the target. Using simple 1D techniques, it is possible to determine a rough transit time. The situation is more complex when determining, say, the operational impedance of a piezoelectric transducer. In that case, “hit” the device with the drive signal approaching a delta function. Then wait for the charge to ring down in the device, which is dependent on the structure of the transducer and whether or not it is in water or air, backed or unbacked, and so on. Such variables make it more difficult to set the simulation time.

You could, of course, simply set a long simulation time, but that would waste a lot of computer time simulating unnecessary aspects of the model.

The most efficient approach is to run a simplified version of the model, monitor the results, and determine the required simulation time. To do this, reduce the model mesh to speed up the calculations (which, of course, reduces accuracy) and insert a loop into the model input file to plot the results as they are computed. This shows how the model is reacting, allowing you to optimize the runtime.

Once you have determined the optimal runtime, return the mesh to the size required for accurate modeling of the device and run the model as described at the start of Section 8.

> Note
> Altering the mesh size has a dramatic effect on runtime. Halving the element size, for example, increases the runtime 8 times for a 2D model and 16 times for a 3D model.

The following code segment shows the use of the DO loop to complete this operation:

```
c               MODEL RUN OPTIMISER
c

c Run model on reduced number of steps to help define the time period that the model
c requires to run. This section should be "commented out" after use.

symb nloops = 100                          /* Total number of runs
symb nruns2 = nint ( $nruns / $nloops )    /* Calculate the time steps required to do defined runs
do loop1 I 1 $nloops                       /* Loop through FEA runs to show results

                                           /* Once the ring down has been established, the new
                                           /* time should be substituted into simtime and this
                                           /* section "commented out."

exec $nruns2                               /* Run model

grph                                       /* Enter graphing function
      nvew 1                               /* Create a display window with one "subwindow"
      plot 3                               /* Plot the charge time history from the electrode
      end                                  /* specified earlier.
end$ loop1                                 /* End loop
```

This uses the number of loops (“nloops”) specified to calculate the time step size. Configure the DO loop as:

do <loop name> <loop counter name> <start no.> <end no.> <increment>

where <loop name> is the name assigned to track the loop and <loop counter name> is the integer counter (usually defined as "I"), which can be used within the loop to increment the file (create more than one file name), and so on. Next specify how many times the loop will be called, from start number <start no.> to end number <end no.>. The start number need not be 1; both the start and end can be any integer number, i.e., 2 ? 50. The <increment> variable allows you to jump up in steps, e.g., to increase the count in steps of 2 or 3, instead of 1.

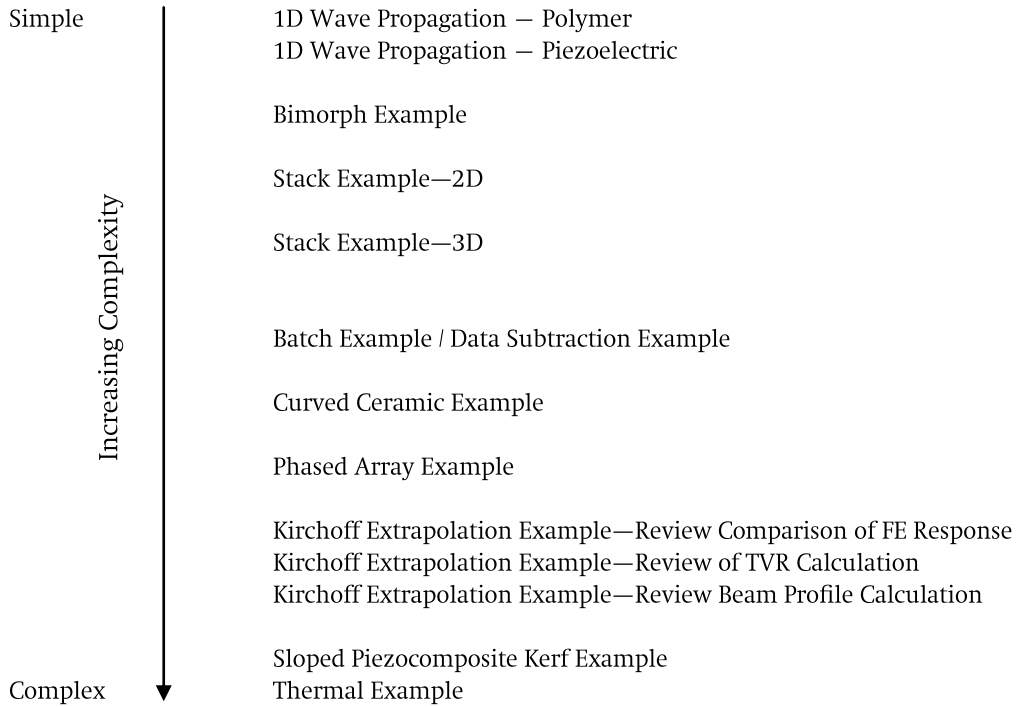
You now have a DO loop called “loop1”; it starts at 1 and goes to nloops (=100). Every time the loop is called, it runs the “exec” command and outputs the charge-time history (time history stored in array 3) to the graphical display as it is computed. This continues until the DO loop is finished.

With this technique, you can run small efficient models to determine the required simulation time and then adjust the input file accordingly.

The Next Step

Having covered the basics for generating a PZFlex input file, you may find it helpful to work through the examples in the Annotated Examples, a series of PZFlex input files using various design approaches. Each example comes with a PDF file that describes the problem and highlights the important points being demonstrated.

The examples begin with a simple wave propagating through a solid epoxy medium and increase in complexity from there. The most complex examples should be left until you are experienced and comfortable with the PZFlex basics. Below is a guide to the annotated examples:

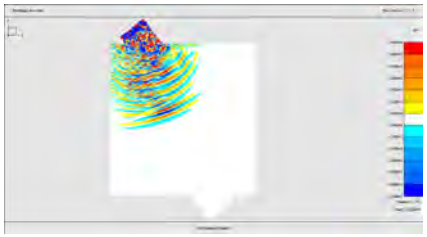


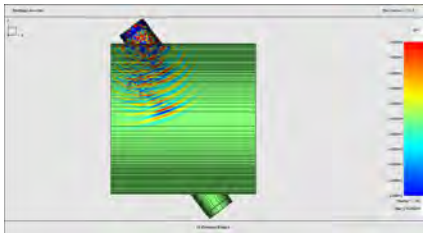
To recap, to run any PZFlex calculation enter:

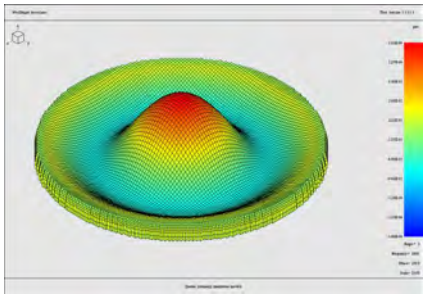
pzflex < jobname>

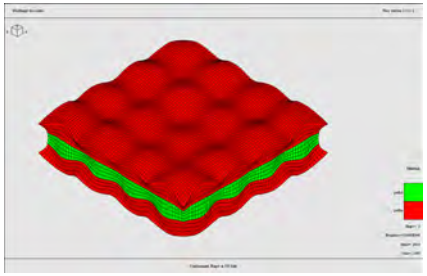
To run any Review post-processing job, enter:

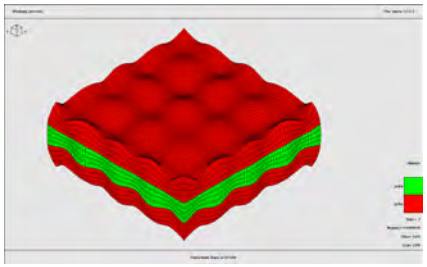
review <jobname>

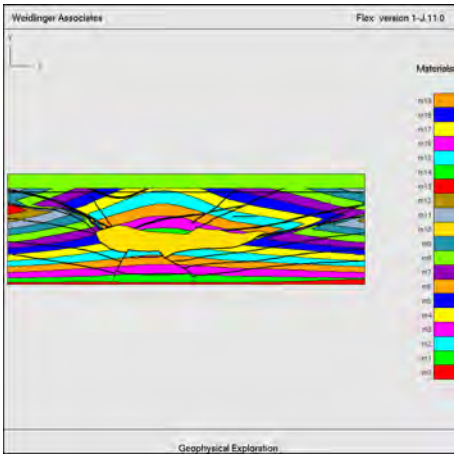


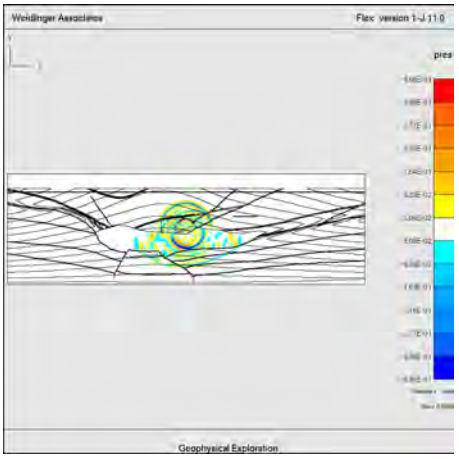


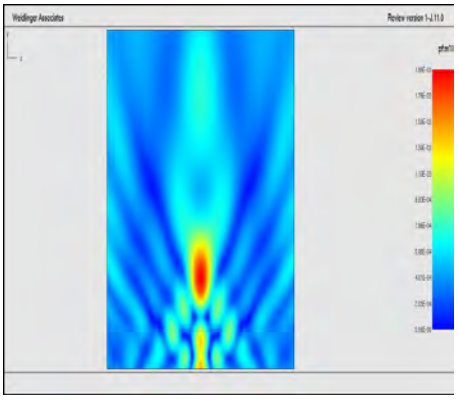







Weidlinger Associates
Flex version 1-J.11.0
Materials
Geophysical Exploration


Weidlinger Associates
Flex version 1-J.11.0
Geophysical Exploration



New Commands

SYMB #KEY

The #KEY options provide an expedient way to quickly define keypoint symbols for standard partition models. The form of the options are:

SYMB #KEYCORD direction indxbegin valuebegin dvalue1 dvalue2 dvalue3 ….

- Direction is x, y or z.
- *indxbegin* is the first index to be defined.
- *valuebegin* is the coordinate value at the first index.
- *dvalue1, dvalue2,…* are coordinate increments to succeeding keypoints.

```
symb  #keycord X  3.  10.  5.  5.
```

Would produce $x3 = 10; $x4 = 15; $x5 = 20. Default for *indxbegin* is 1 greater than the last defined.

SYMB #KEYCOPY direction indxbegin indxend ncopy invertoption

- Direction is x, y or z. *indxbegin, indexend* are the symbols to be copied.
- *Ncopy* is the number of replications
- *invertoption* = INVERT to invert coordinate spacing between each succeeding copy.

If $X1 = 0, $X2 = 1, $X3 = 5, then:

```
symb  #keycopy  X  1.  3.  1.  invert
```

Would produce $X4 = 9, $X5 = 10.

SYMB #KEYINDX direction indxstart indxstop ivalindxstart spacing minval

- Direction is i, j or k. *indxstart, indxstop* are the symbols to be defined.
- *Ivalindxstart* is the node number at *indxstart.*
- *Spacing* is the coordinate spacing between keypoints
- *minval* is the minimum increment between keypoints.
- Default value for *indxstart is 1.*
- Default for *indxstop* is the max defined coordinate keypoint index.

```
symb  #keyindx  I  1.  3.  1.  2.  2.
```

Will define:
$I1 = 1;
$i2 = $i1 + max (2, (nint ($x2 - $x1) / spacing));
$i3 = $i2 + max (2, (nint($x3-$x2) / spacing)).

The coordinate keypoint symbols, $x1, $x2, $X3 must have been previously defined

GEOM—KEYPNT

A shortcut for standard partition models. This option computes the coordinates based on user defined symbols for coordinates and spacings.

KEYPNT nikey njkey nkkey

where:

- *nikey* is the Number of keypoints defined in the I direction
- njkey is the Number of keypoints defined in the J direction
- nkkey is the Number of keypoints defined in the K direction

Using the old coding method, we would have to declare coordinate system in the excessive fashion:

```
geom
     xcrd $x1 $x2 $i1 $i2        /* X axis assignment
     xcrd $x2 $x3 $i2 $i3
     xcrd $x3 $x4 $i3 $i4
     xcrd $x4 $x5 $i4 $i5
     xcrd $x5 $x6 $i5 $i6

     ycrd $y1 $y2 $j1 $j2        /* Y axis assignment
     ycrd $y2 $y3 $j2 $j3
     ycrd $y3 $y4 $j3 $j4
     ycrd $y4 $y5 $j4 $j5
     ycrd $y5 $y6 $j5 $j6

     zcrd $z1 $z2 $k1 $k2        /* Z axis assignment
     end
```


The above is no longer required suing the new *keypnt* command and can simply be reduced to:

```
geom
    keypnt 6 6 2
```

SITE—REGNDUPL

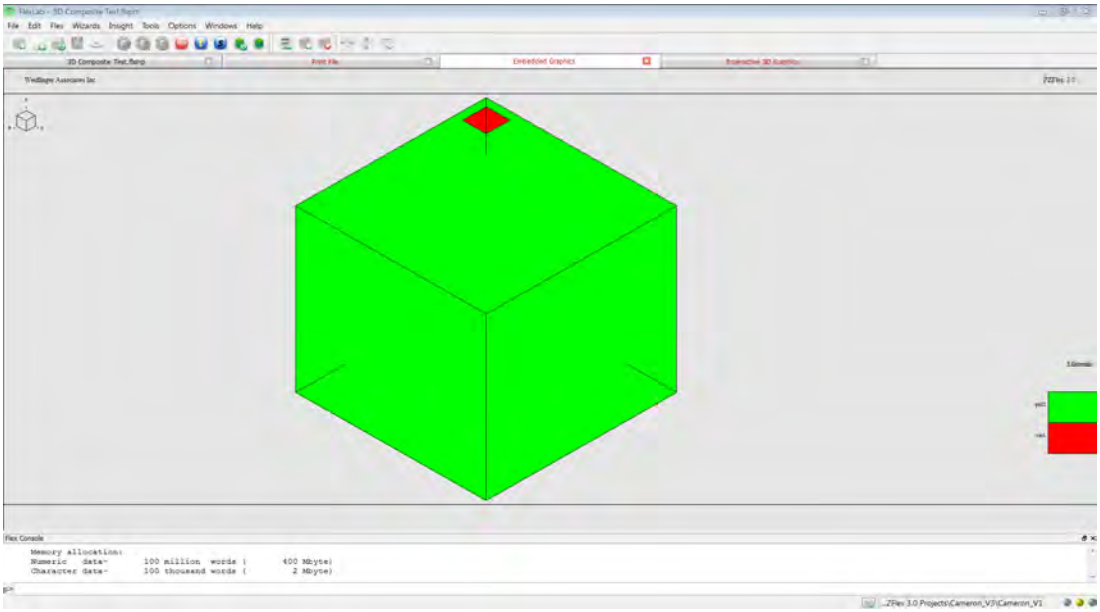
The regndupl command is used to duplicate the materials in a given rectangular region of the computational grid in IJK space. Multiple copies of a periodic structure can therefore be easily generated.

REGNDUPL ibegf iendf jbegf jendf (kbegf) (kendf) dupdir ncopy invertopt

where:

- *ibegf…*, *kendf* are the bounding nodes of the region to be copied. Do not enter *kbef* and *kendf* for 2D models. Default input is the entire grid.
- *dupdir* is the direction of duplication and can be any of *i*, *j* and *k*. Default input is *i*.
- *ncopy* is the number of times to duplicate the region. Default value is *1*.
- *Invertopt* is the option to invert each copy region relative to the previous one.

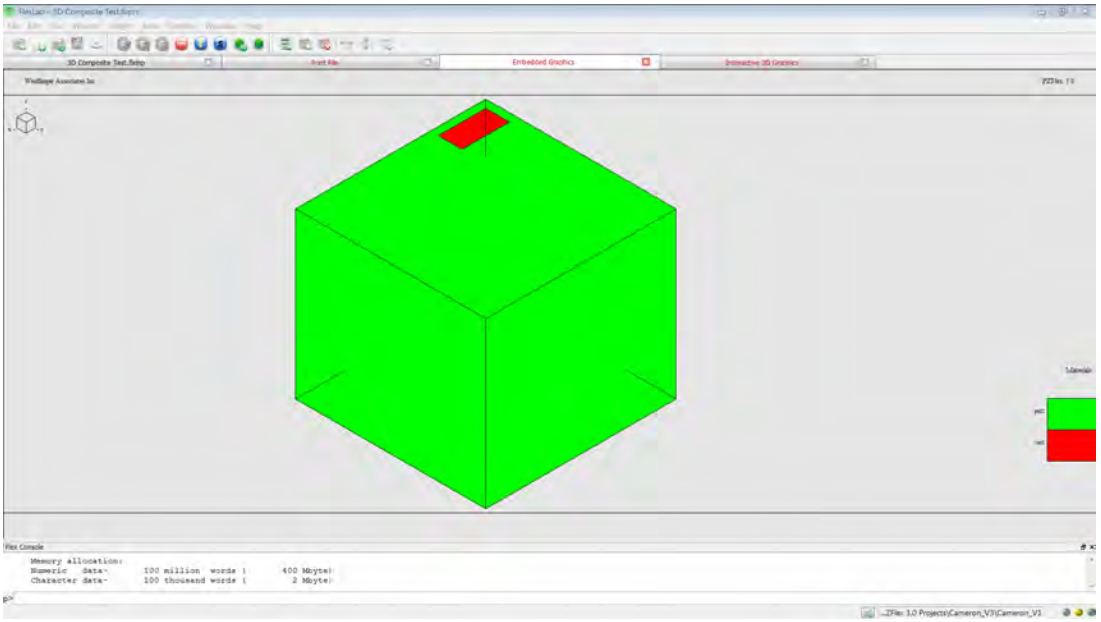
A 1-3 composite structure is a prime example to use the *regndupl* command. Starting off with a basic structure:



Generated from:

```
site
    regn void
    regn pol2
    regn cer1 $i2 $i3 $j2 $j3 $k1 $k2
```

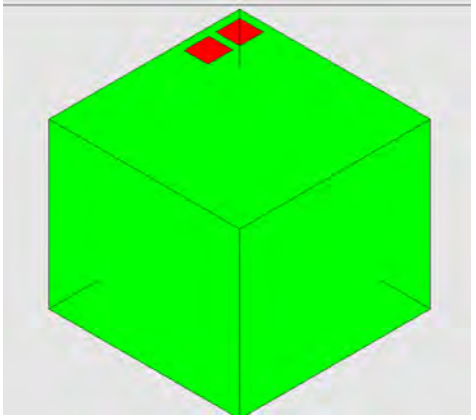
Note that there there must be sufficient IJK coordinates to support the number of times to copy out the repetitive structure along each direction.



The red region has extended out slightly using the following command:

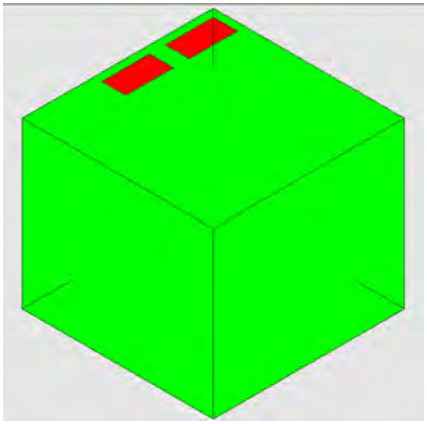
```
regndupl $i1 $i3 * * * * i 1 invert
```

If we remove the *invert* option we, obtain a slightly different structure:



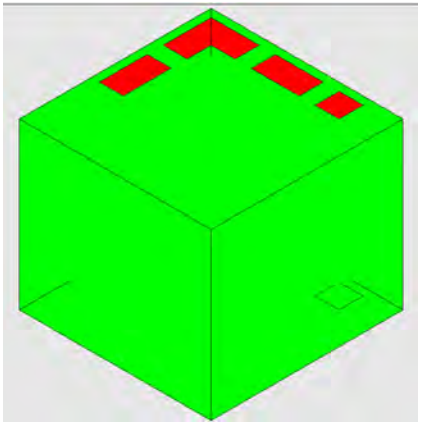
```
regndupl $i1 $i3 * * * * i 3 invert
```

If we change the *ncopy* to 3, we should the structure being copied out a few more times.



Adding another *regndupl* command in the j direction we begin copying out a repetitive structure:

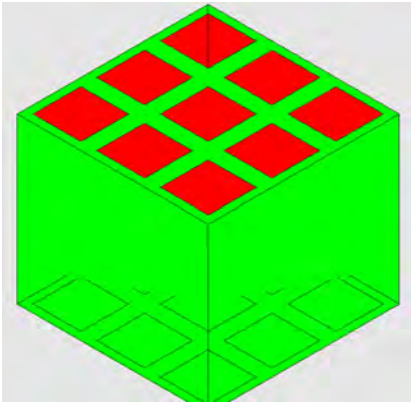
```
regndupl $i1 $i3 * * * * i 3 invert
regndupl $i1 $i3 $j1 $j3 * * j 4 invert
```

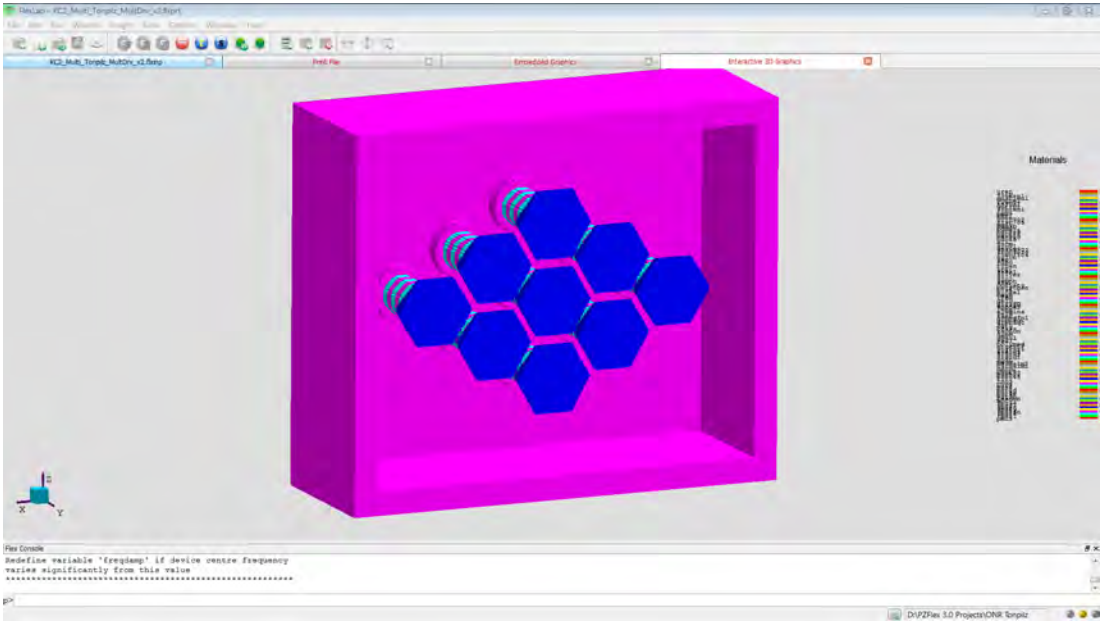
As you can see the region of i1to i2 and j1 to j3 is mapped out 4 time in the j direction.

```
regndupl $i1 $i3 * * * * i 5 invert
regndupl * * $j1 $j3 * * j 5 invert
```

The stars indicate the default values which will represent the all the coordinates along the corresponding direction.



Remember that PZFlex runs code line by line, so be aware that the order of *regndupl* is sometimes very important for more complex structures.



SITE—REGNCOPY

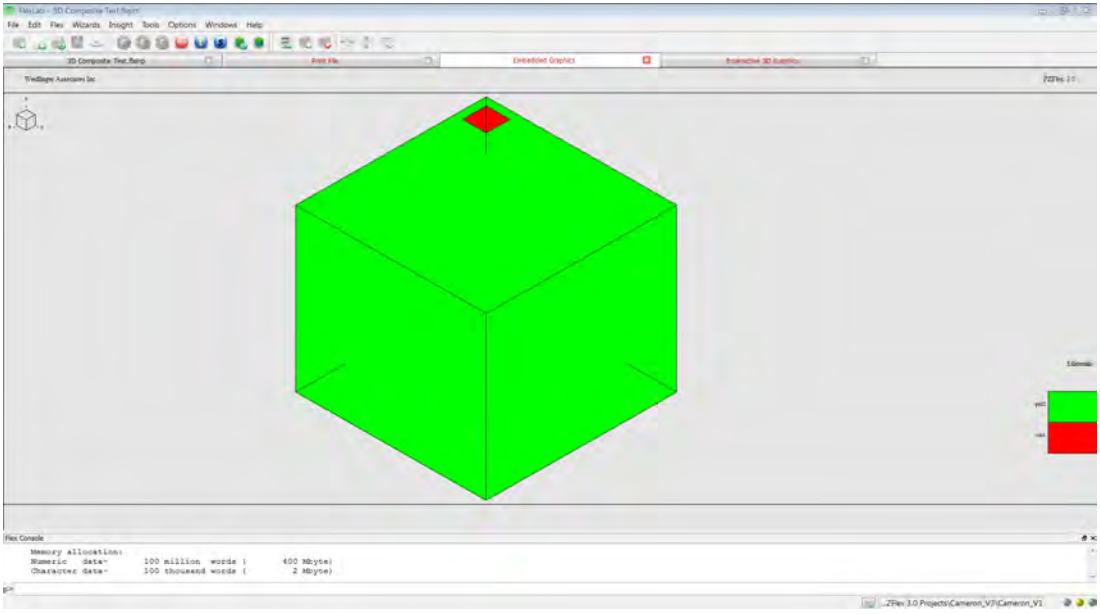
Allows you to copy a material over a rectangular region of the computational grid in IJK space to another space.

REGNCOPY (option) (option) (option) ibegf iendf jbegf jendf kbegf kendf ibegt iendt jbegt jendt kbegt kendt

where:

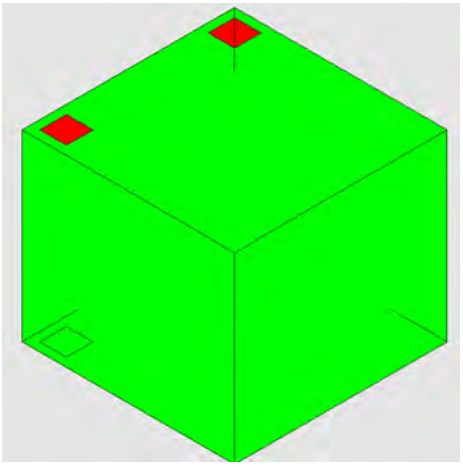
- (option) is the copy directions. Enter any of *revi*, *revj* and *revk* or leave blank otherwise. Combinations of the 3 inputs can be entered i.e. *revi revj*, *revi revk*, *revj revk*, *revi revj revk*
- ...*endf* parameters control the region to be copied
- ...*endt* parameters control where to copy the region to

Using the same structure as before in the *regndupl* we can demonstrate the *regncopy* function very easily.



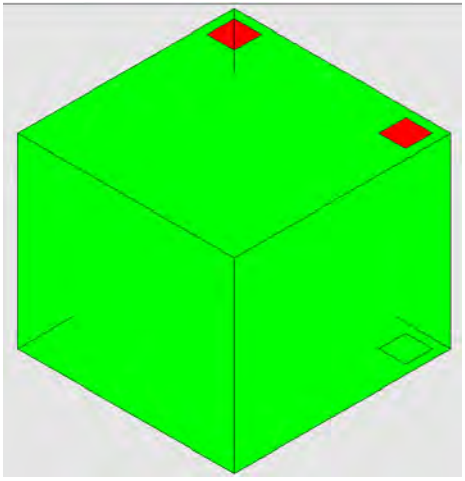
```
regncopy revi $i1 $i3 $j1 $j3 * * * * * * * *
```

The code selects the red square region and copies it to the end I-nodes of the grid.



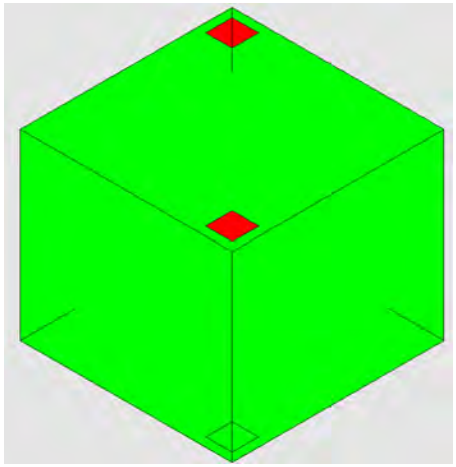

```
regncopy revj $i1 $i3 $j1 $j3 * * * * * * * *
```

Changing the copy direction will copy the small red region in the J direction.



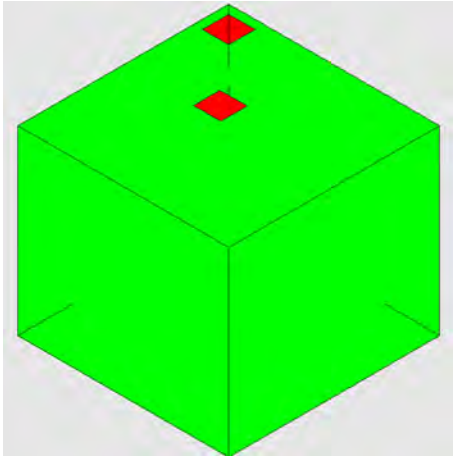
```
regncopy revi revj $i1 $i3 $j1 $j3 * * * * * * * *
```

Adding in another copying direction, it should act like the summing of vectors and move the red region to diagonally opposite corner of the grid.




```
regncopy revj $i1 $i3 $j1 $j3 * * $i5 $i7 $j5 $j7 * *
```

Reverting back to the single copying direction we can define a specific region to copy to and you will find that you have complete control of where the red region can be placed.



Note that IJK grid in use has predefined keypoints.

REVIEW

Introduction

Review is a powerful addition to PZFlex, as well as to the other members of the FLEX software family. It allows you to both display and mathematically manipulate data to determine common design requirements such as operational impedance, acoustic beamplots, transmitting voltage response (TVR), and pressure fields.

Starting in Review

Like PZFlex, Review can be accessed from FlexLAB under Windows, via a desktop shortcut, the quickstart toolbar, or the Start/Programs menu. This brings up Review input file editing functions, an interactive command window, and graphical and text output windows.

Unix users can run Review from a new terminal window by entering the command "review" for interactive sessions or by forming a batch input file (essentially a list of text commands) and running with "review <filename>."

Review input files should be saved under the filename <jobname>.revinp to allow for simple sorting and to enable Review to locate the files in batch mode.

1. Reading in files

Time Histories

Because Review is a separate post-processor for use after a PZFlex run, it is necessary to read in information generated from the PZFlex run and stored in various types of PZFlex output files. The most common type of PZFlex output file is the history file, <jobname>.flxhst, which contains information on individual element or nodal responses at all times, selected using the "pout" function in PZFlex. A typical examples is the time history of the voltage on a transducer electrode, which is analogous to the time trace on an oscilloscope.

When using Review, the first step is usually to read in the aforementioned history file. To do this, direct the external history file into a local file in Review by using the "read" command:

```
p> read  f1  <jobname>.flxhst
```

This reads the PZFlex history into a local file (here called “fl”). You can use up to 8 characters to name the local file. Remember that the PZFlex history file is a series of arrays specified during the model-building phase. Using the example from the PZFlex starter manual:

```
pout
    hist func                          /* Input function           (1)
    histname electrode vq top          /* Volt/Charge on ceramic   (2 - 3)
    hist pres $i4 $i4 1 $j3 $j3 1      /* Pressure output at top   (4)
    hist ydsp $i4 $i4 1 $j1 $j1 1      /* Y displacement on bottom (5)
    end
```

In this case, the local file f1has five entries, all concerning time:

1. Input or drive function applied to the piezoelectric material
2. Voltage on the drive electrode
3. Charge on the drive electrode
4. Pressure output at top of the device
5. Displacement in the y direction at bottom of device

Time histories are saved in the order in which the PZFlex “pout” command requests them. These time histories are now referred to by the local filename (f1here) and the record number in that local file (in this case, 1—5). Use of the “list” command in Review details the time histories in order, with full information on the information stored.

All of the PZFlex annotated examples involve the reading and writing of time histories.

Snapshots

Time-history data are a collection of responses from individual points across the entire simulation time. Snapshot data is the opposite—a collection of responses from many points (e.g., a slice through the model, or even the entire model) at various times. A common snapshot series, for example, would be of pressure in a water-loaded transducer, watching the output pressure wave propagate away from the device. Again, PZFlex saves these data when requested to do so by the “data” command. By default, they are stored in a file with the “flxdato” extension.

To work with the external snapshot file, put it into a local Review file. As with time histories, use the “read” command:

p> read f2 <jobname>.flxdato

The snapshots are loaded into Review in the order in which they were saved. Multiple snapshots are stored in subfiles, i.e., f2/1, f2/2, and so on, and data can be accessed through a combination of this file/subfile structure and the snapshot field name. To plot the second snapshot pressure response, for example, enter:

```
p> grph
s> plot f2/2 pres
```

The “data subtraction” example in the Annotated Examples demonstrates the writing and reading of PZFlex snapshot files.

ASCII Text Data

In Review, in addition to PZFlex input files, it is possible to read in external files (for example, experimental results from an oscilloscope for comparison with simulated results). The data in the external files first must be stored in columns, in tab-delimited format. It is best to format the data in a spreadsheet package such as Excel before attempting to read the information into Review. To read an external file “experimental.dat” into Review, enter the command:

```
p> read  f3  colm  experimental.dat
```

This reads the external file experimental.dat into the local file f3; “colm” tells Review that it consists of columns of ASCII data.

2. Simple Plotting

Although it is possible to view physical data output from the PZFLex model, it is usually easier to interpret data in graphical or visual form. Review allows you to view and manipulate the data in a straightforward manner.

To view the data read into Review above, enter the “grph” command. This takes you to the graphical interface. This is similar to how you viewed models in PZFlex, except here you plot the time history data. To view the drive function applied to the piezoelectric material, enter:

```
p> grph
s>  plot  f1  1
```

When the “grph” command is entered, the prompt changes to “s>” to indicate the subcommand level. The next “plot” subcommand displays the first data set in the local file f1. To view the voltage on the electrode, replace the 1 with a 2. For charge, use 3, and so on. The “plot” command above produces the graph below. Remember that the command “list” gives details of the records in a file.

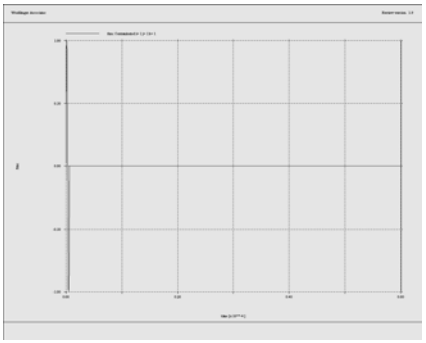


Figure 37

In Fig. 37, it is hard to distinguish the drive function, as the model has been run for a long time period, 0 to 600 microseconds. To zoom in on areas of interest, use the “set” subcommand to set the window size to the desired time frame. The structure of the “set” subcommand is:

s> set wndo <start> <finish>

It is necessary to indicate <start> and <finish>, as this subcommand can be used for any x plane coordinate, such as time or frequency. To trim the time scale down to 0 to 50 microseconds, enter the following command:

s> set wndo 0. 0.5e-5

This changes the graphical display, making it easier to distinguish the drive signal:

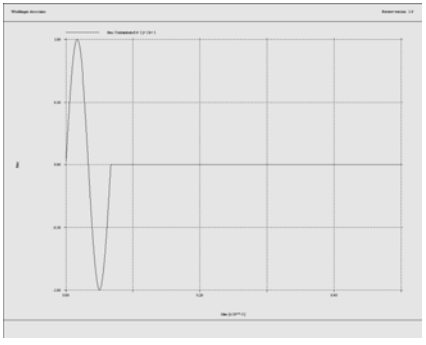


Figure 38

It is possible to display multiple time histories on a single graph. Take, for example, the following series of time histories for a pressure wave traveling through a material (Fig. 38). Each of the three pressure plots has been produced by specifying specific nodes in the model and recording the pressure over the time history. Once the plots have been read into Review, display the plots using the following subcommand:

```
s> plot  f1  1  f1  2  f1  3
```

You can overlay time plots and correlate them on the same time and amplitude scale. Up to six time histories can be plotted on the same graph.

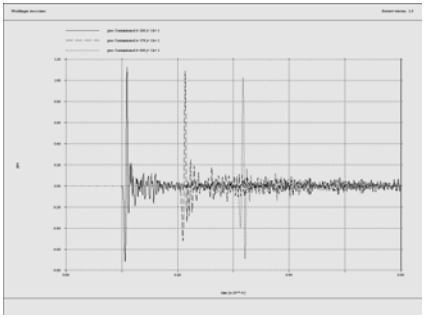


Figure 39

Although the display technique can overlay images, it is sometimes clearer to show multiple images separately in the same window. To do this, again use the “nvew” subcommand (see the Command Reference for the various views available). In this case, enter the following commands to separate the traces:

```
s> nvew  3  5
s> plot  f1  1
s> plot  f1  2
s> plot  f1  3
```

This produces the following graphical display:

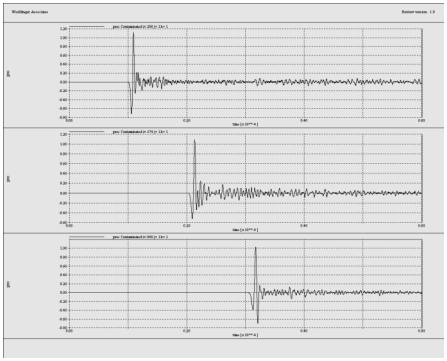


Figure 40

3. Customizing Displays

You can customize graphs by adding titles, changing color tables and legends, and so on.

Note: Unless otherwise stated, all of the following subcommands are under the “grph” primary command.

Adding Titles

To add a title to a drawing to identify the trace, use the “ttl” subcommand. This subcommand differs slightly from many others in that you execute the command and enter the title in the next line, e.g.:

s> ttl

-> pressure plot for wave propagation

The title can be up to 200 characters long. When using this command in a batch file, enter the title in the line below the “ttl” subcommand.

Adding Legend Titles

Review uses the array name and node indices location when forming the default legend for a graphical display. When there is more than one time history on a graph, as in Fig. 3, it is best to define your own legend titles, using the “set” subcommand:

s> set clab 1 ‘Legend 1’

Note

1. The legend title must be in single quotes and contain no more than 80 characters.

2. The number after the “clab” option is the curve number, not the time-history number. The curve number, which is assigned to the graph as the time history is plotted, is not associated with the time history numbers. For example, if you enter the following command:

```
s> plot  f1  3    f1  1    f1  2
```

time history 3 would be defined as curve 1, time history 1 as curve 2, and time history 2 as curve 3, as that is the order in which they were plotted.

Changing X-Axis and Y-Axis Labels

To rename the x-axis and y-axis labels, use the “set” subcommand the same way as for the legend labels. Again, the maximum number of characters is 80. The following code sets the y-axis label as “Pressure in Pascals” and the x-axis label as “Time in microseconds.”

```
s> set  ylab  “Pressure in Pascals”
s> set  xlab  “Time in microseconds”
```

Note

If you include “all” after the title in quotation marks, the labels are specified for all the curves shown; e.g., with multiple curve plots like those in Fig. 40, all the labels are the same.

Changing Line Colors, Thickness, and Style

To change the color of the curves on a graph, use the “lc” option of the “set” subcommand:

```
s> set  lc  <curve number>  <color number>
```

Again, the <curve number> reflects the order in which the curve data were read into the plot between 1 -> 6, and <color number> is the order of the colors already defined from the material color table (see Table 5).

By default, the line width is set to 4/4 inch. To change the width, use the “lw” option of the “set” subcommand. The number after the “lw” option governs the line thickness, with 1 equal to 1/72 inch. The following command sets the line width to 1/36 inch:

```
s> set  lw  2
```

To change the line style, use the “lq” option of the “set” subcommand. This command is formed the same way as the color setting option, with a choice of six line formats:

1 ——————————

2 – – – – – – – – – – – –

3 - - - - - - - - - - - - - - - - - -

4 —·—·—·—·—·—·—·—·—·—

5 —··—··—··—··—··—··—

6 – – – – – – – – – – – –

To change the first curve on the graphical plot to a dot dashed line and the second curve to a short dashed line, enter the following command:

```
s> set  lq  1  4

s> set  lq  2  3
```

The PZFlex Command Reference contains additional options in the “set” command.

Logarithmic and dB Display

If a data set exhibits only small amplitude changes that are hard to distinguish on the graph, you can plot the data set against a logarithmic scale. This is done by setting the x or y axis, depending on your requirements, to log scale:

```
s> set  log  y  on
```

or

```
s> set  log  x  on
```

Similarly, the graph can be plotted in dB:


```
s> set  db  y  on <val>
```

When <val> is left blank, the graph is plotted with the data maximum set as 0dB. Any numerical entry in <val> sets the dB values relative to that number.

To make any of the above changes to the graphical display permanent, substitute “pset” for the “set” subcommand.

4. Mathematical Functions

Review allows you to mathematically manipulate the PZFlex time-history data. Manipulations can range from simple scaling of the data to Fourier transformations of the time-domain data.

Most of the mathematical functions are in Review's "make" primary command. It is common practice when manipulating previous data records to store the results in a new local file. This keeps the modified data separate from the original time history file, preventing data corruption.

Take, for example, a time history file named results.flxhst, which contains seven time histories:

1. x-velocity
2. y-velocity
3. z-velocity
4. x-displacement
5. y-displacement
6. z-displacement
7. pressure

A good way to begin is by learning how to determine the difference between two time histories. Read in the overall time history, create a new local file for the results, and do the calculation:

```
p> read  f1  results.flxhst
p> make
s> file  f2
s> curv  { f1  1 }  –  { f1  2 }
```

Use the "read" command to enter the calculated time-history file and then the "make" primary command. Then use the "file" subcommand to create a new local file named f2 and use "curv" to subtract record 2 in file f1 from record 1 in file f1. The difference between the two files is stored in record 1 of file f2.

Note

1. The "curv" subcommand uses curly brackets.

2. When you generate a text batch file for use in Review, the same rules govern the use of subcommands as in PZFlex, i.e., the subcommand is indented in the text batch file.

The following, slightly more complex example shows how to determine the absolute magnitude of displacement of a node with respect to its initial position. In the time history, the x, y, and z displacements are stored in arrays 4, 5, and 6. Enter:

```
p> read  f1  results.flxhst
p> make
s> file  f2
s> curv  sqrt (  { f1  4 } ** 2. +  { f1  5 } ** 2. +  { f1  6 } ** 2. )
```

Take the square root of the sum of all the displacements squared. The magnitude of displacement is stored in record 1 of file f2. Note again the use of curly brackets to specify the array records.

To calculate, say, the radial velocity (in the y-z plane) of a node at 30 degrees, the “curv” subcommand is:

```
s> curv  cos ( 30. * 3.14159  / 180  ) * {  f1  2  } +  sin  ( 30. * 3.1459 / 180  ) * {  f1 3  }
```

To simplify this large input string, use the symbol or “symb” command to break up the equation:

```
s> symb  angle  =  30.
s> symb  cosval  =  cos  (  $angle  *  3.14159  /  180  )
s> symb  sinval  =  sin  (  $angle  *  3.14159  /  180  )
s> curv  $cosval  *  {  f1  2 }  +  $sinval  *  {  f1  3  }
```


To save the data in the new local file f2, exit the “make” command directory and write the local file to an external file:

```
s> end
p> writ  f2  newfile.dat
```

Here the local file f2 is written to a file newfile.dat.

Converting to Frequency Domain

You can also use Fourier transform techniques to translate the data in the time-history files from the time domain to frequency domain. This is done not with the “make” command, but with the frequency input or “freq” command.

For a time-history file with the drive input function of a single-cycle sinusoid at 1MHz in the first array, you can use the “freq” command and associated subcommands to convert it to the frequency domain:

```
p> read  f1  results.flxhst
p> freq
s> file  f2
s> fft  f1  1
s> end
p> grph
s> nvew  2  1
s> plot  f1  1
s> plot  f2  1
s> end
```

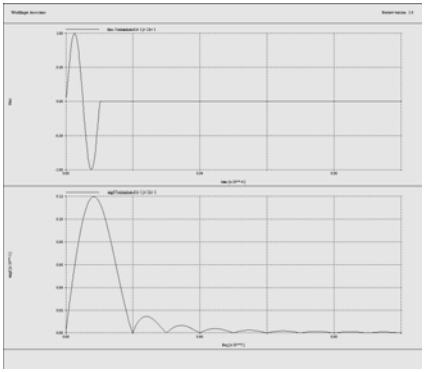


Figure 41

The second graph in Fig. 41 shows the frequency response corresponding to the FFT of the time-domain signal in the first graph. Any single “freq” command creates two frequency domain records: amplitude and phase by default, or, if requested with the “type” subcommand, real and imaginary.

5. Merging files

Review allows you to merge simulation and experimental files so they share a common time or frequency range.

When using a digital oscilloscope or frequency analyzer to store experimental data, you are usually limited to the format in which the equipment stores the information. It is usually based on a binary sampling rate, i.e., the number of data points is to the power of 2—1024, 2048, and so on. As a result, the time step between experimentally derived samples usually differs from those generated by PZFlex in the time history. To merge the two data sets, use the “make” command.

After reading in the two files (see Section 1), use the “freq” or “time” subcommand to define the x axis:

```
p> read  f1  results.flxhst
p> read  f2  colm  experimental.dat
p> make
s> file  f3
s> freq  from  f2
s> curv  {  f1  1  }
s> curv  {  f2  1  }
s> end
```

The PZFlex history file is read into local file f1and the experimental columns of data into local file f2. After entering the “make” primary command and generating a new local file f3, use “freq from f2 to tell Review to use the current frequency step from the experimental file for the new local file. You then generate two new curves for f3, from the PZFlex history file and from the experimental data.

6. Exporting Local Files

It is sometimes useful to output raw data from Review. After using data to generate the acoustic beam pattern of a piezoelectric transducer, for example, you might want to export the raw data for inputting into a spreadsheet.

When exporting the data of displayed images, you must be in the “grph” primary command and graphically displaying the data you wish to export.

First decide whether to form a data column for the x axis for each curve, or for only one. One column for the x-axis data is usually sufficient. To export the data, use the “writ” subcommand:

```
p> grph
p> plot  f2  1  f2  2
s> writ  onex  nolb  output.dat
```

This exports the curves derived from records 1 and 2 in the local file f2 to an external file, output.dat. The external file contains three columns of data; the first for the x-axis data and the other two for the y-axis data for curves 1 and 2.

Note

The “nolb” option exports the data without including the label from the local file. The data are in columns, in tab-delimited format.

7. Calculating Operational Impedances

Operational impedance plots are very useful for piezoelectric transducer designers, as they show at a glance the likely frequency characteristics of the device. This section shows you how to use the information in the PZFlex time history file to generate the operational impedance of the device.

Begin by defining the saved time histories in PZFlex:

```
c Which time histories to save?

pout
    hist func                          /* Input function           (1)
    histname electrode vq top          /* Volt/Charge on ceramic   (2 - 3)
    hist pres $i4 $i4 1 $j3 $j3 1      /* Pressure output at top   (4)
    end
```

The voltage and charge time history on the electrode where the drive function was applied are stored in records 2 and 3. To determine the operational impedance, enter:

```
p> read  f1 results.flxhst
p> freq
s> file  f2
s> impd f1 2   f1 3
s> end
```

The “impd” subcommand calculates the operational impedance based on arrays 2 and 3 in local file f1 and outputs the magnitude and phase to arrays 1 and 2, respectively, in local file f2:

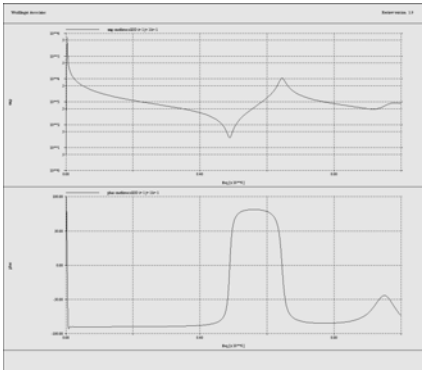


Figure 42

The “list” command, which details the file contents, allows you to determine which records contain the voltage and charge time histories. By default, impedance data are output in magnitude and phase format; use of the “type” subcommand can change this to real and imaginary.

The upper section of Fig. 42 shows the magnitude of the impedance with the main thickness resonance clearly defined, and the lower portion shows the corresponding phase characteristics.

In some cases, the impedance plot is poorly discretized (or fractured):

This is because the number of data points in the time history is insufficient to produce a suitably small frequency step size when the FFT is applied. The frequency step is depend-

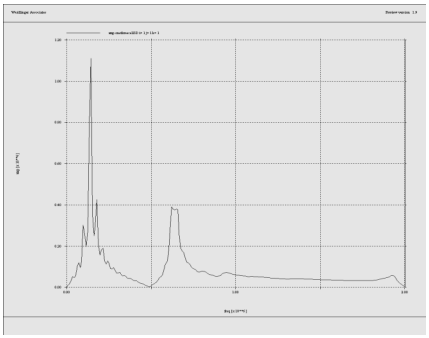


Figure 43

ent on the number of time steps, expressed by the following equation:

Here Δf is the frequency step, N_T is the number of points in the time-history array, N_F is the number of points in the frequency array, and Δt is the time step. Because N_F is half the

$$\Delta f = \frac{1}{N_T \Delta t} = \frac{1}{2N_F \Delta t}$$

size of N_T, when you FFT the data, you halve the number of data points in the set.

The solution is to pad out the time history with extra zeros. Although this may sound like a cheat, it simply adds extra data points after the charge on the electrode in the model has already rung down to zero. This does not change the data; it merely extends the data set. The addition of the zero data points in the time set increases the number of points in the frequency set, thereby decreasing the frequency time step.

For example:

p> freq

s> file f2

s> time pad 3

The number following the “time pad” subcommand determines the size of the multiplier for the time set. Here, the 3 increases the number of points threefold. Fig. 44 compares impedances with (upper trace) and without (lower trace) time padding.

Another common problem with impedance plots is the addition of ringing resonances in

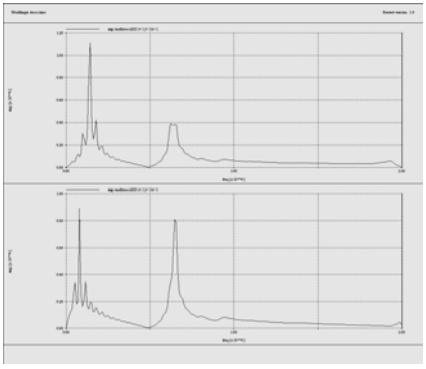


Figure 44

the operational impedance, when the simulation has not run for long enough and the charge has not dissipated from the electrode. This truncation of the charge signal introduces spurious signals into the impedance characteristics. Windowing the new local file prevents generation of these frequencies:

p> freq

s> file f2

s> wndo bart righ 0.9

s> impd 2 3

s> end

This is similar in principle to applying high- or low-pass filters to electronic signals to remove unwanted frequency content. The smoothing is achieved at the expense of removing data from the result; that is, because you have artificially damped the response, the results are less accurate. The preferred method of ensuring smooth frequency-domain results is to run the simulation long enough for the system to reach equilibrium.

Taking the preceding example, apply a Bartlett low-pass filter to the data, smoothing the frequency data:

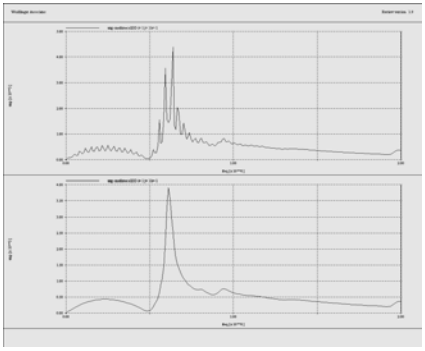


Figure 45

Fig. 45 shows the results for impedance calculations. The subcommand “admt” calculates electrical admittance in the same manner as the impedance calculations.

Most PZFlex examples use Fourier techniques with the “freq” command.

8. Kirchoff Extrapolation

PZFlex works most efficiently when models are on the order of tens of wavelengths in size along each axis. Medical and Sonar devices, however, commonly require predictions of pressure amplitudes at distances many hundreds of wavelengths from the source. These are either impractical or inefficient to model fully. The best approach in such cases is to extrapolate.

The Kirchoff extrapolation technique is a powerful tool incorporated into PZFlex. It uses analytic techniques (essentially a combination of Green’s functions and Huygen’s principle) to take pressure/time data from the finite-element model and extrapolate to a full pressure/time response at any arbitrary location or locations. The technique is valid in both near and far fields, but makes several assumptions:

1) Propagating medium is homogenous, i.e., no intervening layers

2) Propagating medium is fluid, i.e. no shear waves

3) Propagating medium is either lossless or very low loss

4) Response is linear

Fortunately, these assumptions are appropriate for most ultrasonic transducers in water.

Implementation of the Kirchoff method is a two-step process: the first in the PZFlex input file and the second in Review to determine the acoustic beam profile.

The FE model is developed in PZFlex as usual, with the device in the fluid medium. Place an “imaginary box” around the device, so the box is completely contained within the fluid. The box does not interfere with or affect the response in any way; it merely records the pressure and pressure gradients during the simulation, for use during the extrapolation in Review. Although ideally, the box fully encompasses the pressure source, this is not always practical. In such cases, place the box around as much of the device as possible. The box should be close to, but not in contact with, the pressure source; 5 to 10 elements away is typical. The following code segment defines the extrapolation field:

The “ref” command provides a reference point from which to calculate the pressure gradi-

```
extr
    ref in $x1 $y1  /* Set internal reference point for pressure gradient calculation
    defn kirc       /* Kirchoff extrapolation
    node 1 8 1 1    /* Define node surface
    node 1 8 8 8    /* Define node surface
    node 1 1 1 8    /* Define node surface
    node 8 8 1 8    /* Define node surface
    end
```


ent for the extrapolation. Ideally, it is located at the center of the simulated device. The option “in” specifies that the point is inside the device, and the option “out” specifies that it is outside the device. The three variables that follow define the reference point in space.

Next use “defn” to define the Kirchoff extrapolation method and the familiar “node” command to specify the line of nodes that forms the Kirchoff extrapolation field. This produces the following model:

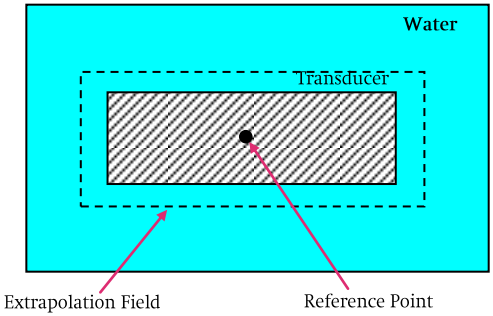


Figure 46

Either place the fluid boundaries far enough from the box that no signals return to contaminate the results or place absorbing boundary conditions at the edge of the fluid. Although absorbers allow for use of a smaller model, they should always be at least 20 elements and 1 wavelength from the pressure source.

The data is saved as the PZFlex extrapolation file jobname.flxext.

The actual extrapolation is performed in Review. There are two forms of extrapolation: time domain and frequency domain extrapolation:

The time-domain solver extrapolates the full transient response to a point in space. Therefore it can be used for pulse-echo responses and broadband responses. Because this method is computationally expensive, however, it is typically used for a small number of extrapolation points, e.g., beam profile arcs and TVRs.

The frequency-domain solver produces a result for a single frequency, i.e., what the field looks like when the device is driven continuously at one frequency. This is similar to producing a mode shape for pressure. Because the computational requirements for the frequency solver are smaller, it is easier to generate 2D and 3D beam profiles.

Beam Profile Arc (Time Domain)

Using the extrapolation field implemented in the PZFlex input file and simulated to produce the field in the file jobname.flxext, you now can determine theoretically the beam profile on an arc at a given radius from the transducer:

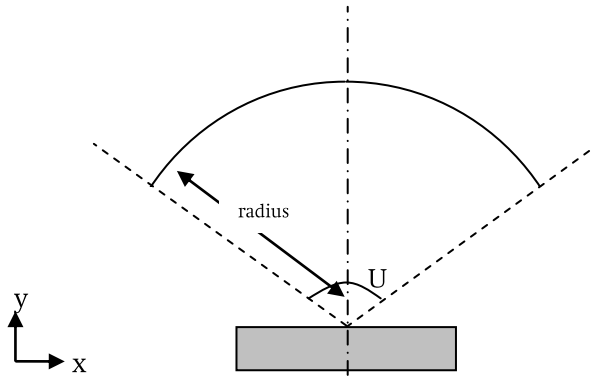


Figure 47

Here, the time-domain version of the Kirchoff solver is used to look at the pressure field along an arc from a 2D model that is symmetrical about the y axis, running through the center of the device:

```
axis
    form angl                  /* Use angle rotation method
    defn beam cyl2 $x1 $y2 0. /* Define âbeamâ axis as front face of transducer
    end

extr
    slvr time             /* Use time domain solution
    data filename.flxext /* Read in extrapolation data
    file f2  cler  kirc  /* New local file f2, cleared and identified as âkircâ
    mirr y- symm         /* List any symmetry conditions (axisymmetry already accounted for)
    show ptop            /* Save peak-to-peak response
    surf cyln beam $radius 0. 0. 1 $strtangl $endangl $incangl  /* specify extrapolation points
    calc 1               /* Calculate
    end
```

First define a new axis at the center of the front face of the transducer; the variables “$x1” and “$y2” indicate the location.

Next define the extrapolation procedure by entering the “extr” primary command and setting the solver for the time-domain method with the “slvr time” command. Specify the file the field extrapolation is stored in from the PZFlex run and start a new local file for extrapolation information.

Inform the solver of any symmetrical boundaries in the model and specify whether to calculate peak or peak-to-peak amplitudes. The solver automatically detects and compensates for axi-symmetrical boundaries.

There are three varieties of the “surf” command:

1. Use “quad” to generate 2D beam profiles or slices through a given transducer pressure output.

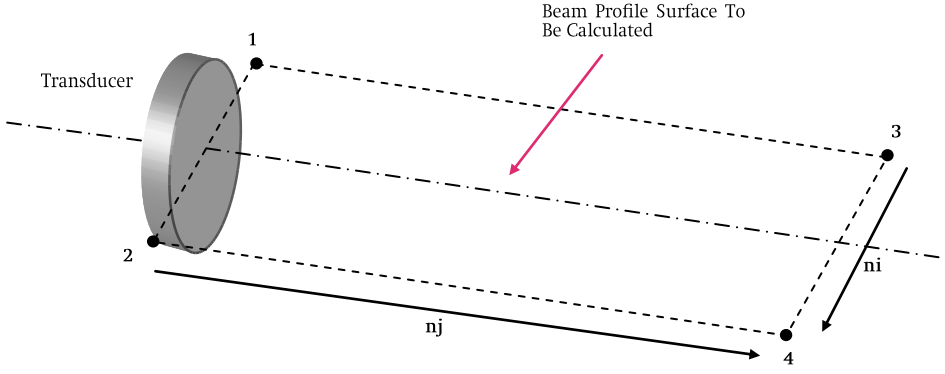


Figure 48

In this case, the surf command is structured as:

surf quad <axisname> ni nj x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4

where, ni and nj are the number of points to be calculated in either plane and x1, y1, z1, and so on are the xyz locations for each point in Fig. 48. Point 1 is always at the minimum i-j location.

2. Use “cyln” to determine the pressure at points along an arc at a specified radius. It can also be used to generate a 3D cylindrical plane.

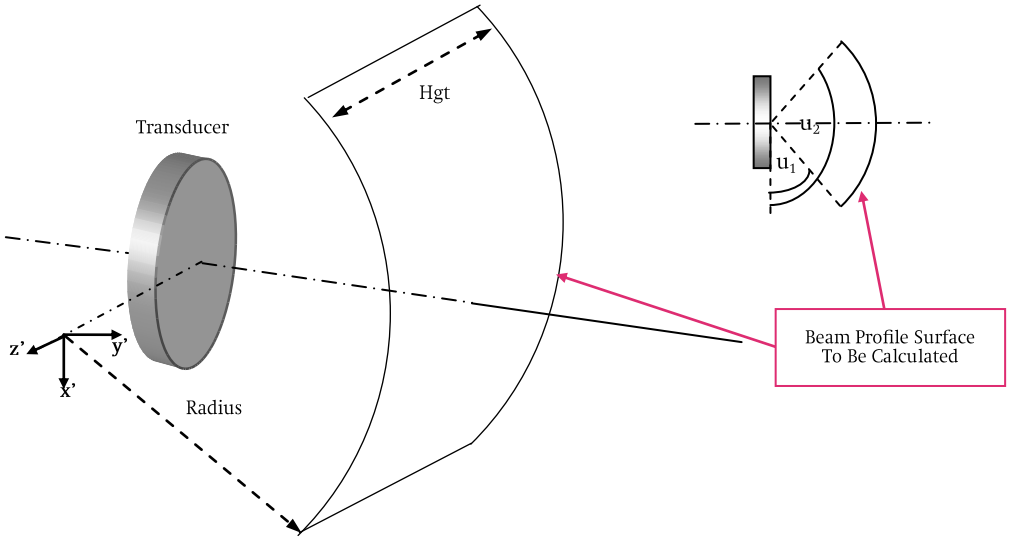


Figure 49

In this case, the “surf” command is structured as:

surf cyln <axisname> $radius $z1 $z2 $incz q_1$ q_2$ $inc_angle

where the axis is defined at the front face of the transducer under the “axis” command and saved as an “axisname.” In Fig. 49, the variable “$radius” is defined as the distance from the front face of the transducer to the cylindrical plane. “$z1” and “$z2” are the z-axis locations that define the height of the cylinder and are specified as “hgt” (for single arcs about the center of the axis, this is set to zero for both Z points), and the variable “$incz” specifies the number of points into which the line is split. Likewise, the variables “q_1$” and “$q$_2$” determine the starting and finishing angles of the arc sweep and “$inc_angle” the number of steps in the arc sweep.

3. Use “sphr” to produce a 3D spherical beam plot, which can be used to determine the entire (or partial) pressure field at a given radius in front of the transducer.

In this case, the “surf” command is structured as:

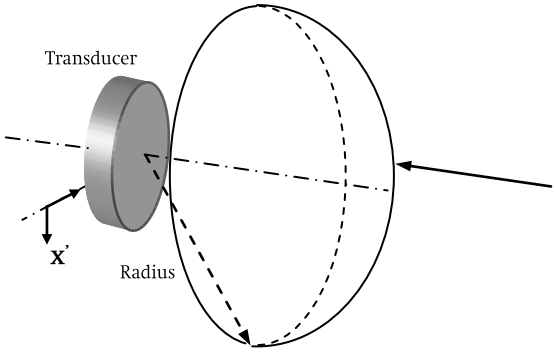


Figure 50

surf cyln <axisname> $radius q_1$ q_2$ $inc_angl1 f_1$ f_2$ $inc_angl2

where the axis has been defined at the front face of the transducer under the “axis” command and saved as an axisname. The variable “$radius” is defined as in Fig. 49. “q_1$” and “$ q$_2$” determine the starting and finishing angles of the arc sweep, and “$inc_angl1” the number of steps in the arc sweep in Fig. 49. Likewise, the variables “f_1$” and “$f$_2$” determine the starting and finishing angles of the arc sweep and “$inc_angl2” the number of steps in the arc sweep.

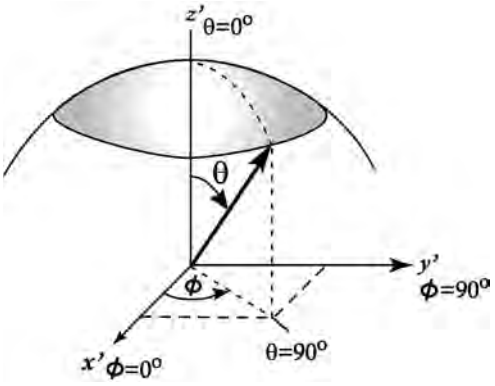


Figure 51

Fig. 51 shows how the extrapolation is referenced from the user-defined axis (note the conventions for q and f).

Finally enter the “calc” command. By default, you need calculate only one step to determine the beam profile of the device.

Fig. 52 shows the beam profile of the transducer at a 1-m radius in front of the transducer, as predicted by the extrapolation method.

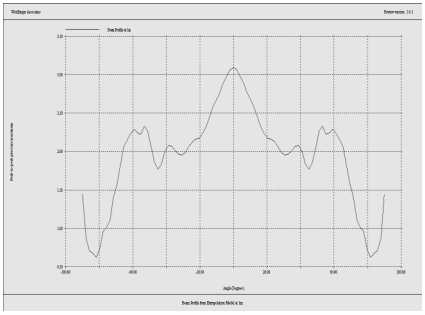


Figure 52

Frequency-Domain Extrapolated Beam Profile

As mentioned above, the frequency-domain extrapolation solver can generate acoustic pressure profiles for a single frequency. These calculations are significantly quicker to solve than those of the time-domain solver and, unlike those of the time-domain solver, can include losses. As with the time-domain solver, you can calculate 2D fields, arc radii, and spherical planes.

The following example calculates a 2D beam profile plot in front of the transducer. As with the previous example, first set the extrapolation field in PZFlex and execute the PZFlex job to generate the flxext file.

The following code segment generates the beam profile in Review:

```
symb xs = $x2             /* Point on front face of transducer
symb xt = $x2 + 250.0e-3 /* Point in far field, 250mm in front of transducer in water
symb ys = $y1             /* Bottom of model
symb yt = $y7             /* Top of model
symb zs = 0.              /* No Z â i.e. 2D model

symb ny = 100            /* No. of steps in y direction
symb nx = 500            /* No. of steps in x direction

symb freqint = 1.0e6     /* Frequency of interest

extr
   slvr freq             /* Use frequency domain solution
    freq $freqint        /* Insert frequency of interest
    type amp             /* Calculate amplitude and phase
    data model.flxext        /* Read in extrapolation data
    file f2  cler kirc       /* Name output file
                             /* Define surface area of interest
    surf quad stnd $nx $ny $xs $ys $zs $xt $ys $zs $xs $yt $zs $xt $yt $zs
    calc 1               /* Calculate the beam profile
    end
```


First set the x and y coordinates for the predicted field. This is usually done by importing the symbol file, symb.<jobname> from the previously run PZFlex job to determine the geometry of the model. As “$x2” is the front face of the transducer, set that as the starting x location and set the end location at 250mm distance. Set the y locations from the model as “$y1” to “$y7” and set the z location as zero to inform the model that you are dealing only with a 2D extrapolation of the beam profile.

Specify the number of sample points for the x and y directions. In this case, as the model was 50mm wide and the propagation distance 250mm, keep the aspect ratio of the sample points even, by setting the x and y samples as 500 and 100, respectively. Define the frequency as a variable for later use.

Enter the “slvr freq” secondary command to tell Review to use the frequency-domain extrapolation solver. Then tell the solver what frequency you are interested in and use “type mag” to say that you want to see results as magnitude and phase. If you prefer, you can view them in real and imaginary. Then read in the previously calculated extrapolation field and set the new local files for the beam profile.

In this case, you want to visualise a 2D beam profile in front of the transducer. Enter the “surf quad” command and then the “calc” subcommand to generate the beam profile. Fig. 53. shows a beam profile:

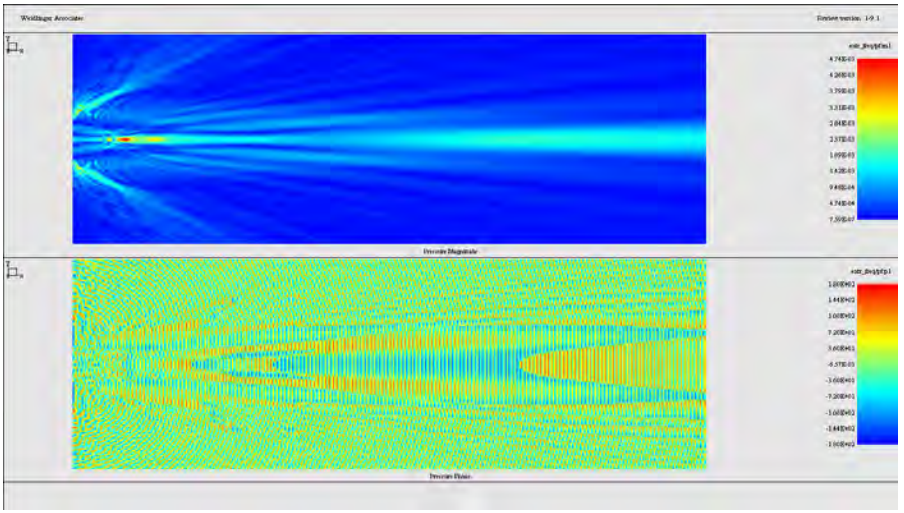


Figure 53

APPENDIX A
ADVANCED MODELING TOPICS

1. Standard and Skewed Partitions

The vast majority of devices are most efficiently modeled using a standard partition, i.e., rectangular elements. In rare cases, skewed partitions, i.e., elements with arbitrary shape, are necessary.

A computational grid can be divided into two types of regions: standard and skewed partitions. The skewed partition is a rectangular region in ijk space that includes all elements of the model that are not perfect rectangles in 2D (right prisms in 3D) and those that do not conform to the standard partition element orientation. Consequently, the skewed partition allows completely general element shapes and orientations. The standard partition is defined as all elements of the model not in the skewed partition. A 2D or 3D model can be composed entirely of standard partition elements or skewed partition elements or a mixture of the two (see Fig. 54 for a 2D model). The subcommands used to define nodal coordinates for elements in the standard partition differ from those used for the skewed partition. When the model has nodes common to both the standard and skewed partitions, the node coordinates must be defined twice, once using skewed partition subcommands and once using standard partition subcommands.

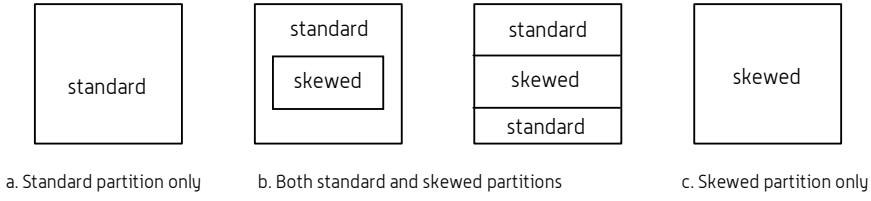


Figure 54

Note: The standard partition includes all elements not in the skewed partition.

For further information on skewed partition models, refer to the Build section of the Command Reference.

2. Problem Time Step

The computational time step in a PZFlex model is restricted by the Courant stability criterion, which sets the maximum time step at which the model can run in a stable manner. This time step is determined by the fastest transit of a wave between any two nodes in the model. During the process (PRCS) step, PZFlex sweeps through all elements, calculating the minimum time step in that element (smallest side/element material velocity); the smallest such value anywhere is the maximum stable time step.

By default, PZFlex sets the actual time step to 80% of this maximum value, as many aspects of modeling, such as the skewed partitions, can reduce the stable operation time step to below the theoretical maximum. Use the "time" command to set the time step to any value up to and including the maximum stable time step. The likelihood of an unstable model increases as the time-stability factor (chosen time step/maximum time step) approaches 1.0. Most 2D and 3D standard partition models work stably up to around 0.9, which results in problems being solved approximately 10% faster. Axisymmetric models should be left at the default value, as they are inherently less stable. To manually alter the time step, use the "time" command:

```
time      time step  timebegin  timefactor
```


where "time step" is the model time step for the analysis, "timebegin" is the initial start time of the analysis (typically zero), and "timefactor" is the safety factor (less than 1.0), which scales the wave transit time across an element to ensure a stable computation. For example:

```
time 1.e-9
```

sets the time step to 1 nanosecond and

```
time * * 0.5
```

sets the time stability factor to 0.5.

3. Defining Computational Zones for the Model

Different regions of the model require different time steps for stability. For example, for equal-sized elements in water (wave velocity 1500 m/s) and steel (wave velocity 5900 m/s), water has a maximum stable time step 3.93 (5900/1500) times larger than the steel. The base time step for the model is set by the faster material, steel, so 3.93 times as many time steps are carried out during a simulation in water than is needed for stability. Zoning a model is a way to use that "overly stable" time step to regain computational speed. By running only every nth time step in a region of the model with a higher stable time step, a stable solution can be obtained in a shorter time. The ratio in each region, N, must be an integer that is less than or equal to the maximum ratio. Therefore you must round down the ratio in each region to the nearest integer. In the example of water and steel, the water can be sub-cycled every 3rd time step (as opposed to every 4th,), as 3.93 is rounded down to 3.

At the interface of two materials, the contacting elements (e.g., water and steel) share a common surface. For the purposes of zoning, PZFlex considers both elements to have the wave velocity of the faster material. This means that a zone must be moved one element into the faster material to gain maximum benefit from zoning.

PZFlex automatically zones the entire model into a single zone by default. Further enhancement must be done manually. Following are guidelines for manual zoning:

1. All elements must be included in a zone.
2. An element can be included only in a single zone (no overlap).
3. Each zone is rectangular in shape.
4. A zone cannot cross the standard/skewed partition boundary (advanced).
5. As there is a small overhead for each zone, you should not create a large number of very small zones.

The “zone” command, which defines a zone, is:

```
zone                    iratio  ibegin  iend  jbegin  jend  kbegin  kend  zonestep
```

where iratio is the integer multiple of the zone time step to the model time step. [Note: We recommend that iratio remain in its default value.] The nodal indices ranges define the region of the model in ijk space to be assigned to the zone. The zonestep parameter is a master override of the zone time step. When entered, this value is used for the zone time step even if it violates the stability criterion. It also overrides the iratio parameter. An example of the “zone” command for a 2D model is:


```
zone                    *  1  20  1  15
```

where the zone time step is default for all elements bounded by nodes with an i-index from 1 to 20 and a j-index from 1 to 15.

Overriding of the default ratio is usually used with a manual entry when an extrapolation (extr) surface is in several different zones, as “extr” surfaces must be in zones of identical time step.

Further detail on zones in a model can be found in the print file (`.flxprt`) just after the “prcs” command is entered.

4. Data Command

The “data” command allows you to import (“in” subcommand) or export (“out” subcommand) any real-number data group in the code. Each of these subcommands transfers an entire data group. The “in” subcommand reads data groups from the data input file. The “out” subcommand writes data groups to the data output file. In general, “data out” is most useful for saving snapshot-style results during an analysis. The simple format of the data output file is described in the section “Files and Formats.” By default, the file saved uses the `jobname` as the filename, with `flxdato` as the suffix. For example, the following lines in the file `transducer.flxinp`

```
data
    out pres
    end
```

save the pressure at every element in the model at the current time step as an array in the file transducer.flxdato. By default, “data out” writes information on the model structure and material for future reference. To change the name of the file, use the “file out” subcommand:

```
data
    file out diffname.out
    out pres
    end
```


saves the data to the file `diffname.out`.

Multiple snapshots of the same field at different time steps can be saved in a single Flxdato file. PZFlex annotates the arrays in the order in which they were requested. This is similar to placing them in subdirectories in the saved file and makes later reference to the array simple. Reading snapshots into Review with the “read” command is similar to reading in time histories, e.g.:

```
read d1 transducer.flxdato
```

reads in all snapshot data in the Flxdato file to the record name `d1`. Multiple pressure snapshots are accessible in subdirectories named 1,2,3, and so on. For example,

```
grph
    nvew 2
    plot d1/1/pres
    plot d1/3/pres
    end
```

plots the 1st and 3rd pressure array in the file `d1`.

5. Restarts

PZFlex allows you to save the current state of the model as it exists in memory at any particular time step to a restart file (flxrsto), for later use. As PZFlex is a time-domain finite-element program, only the current state of the model is necessary to predict the next time step. Therefore, this restart file can be loaded into PZFlex later and the simulation restarted at the same point at which it was saved. By default, PZFlex saves such a restart file upon exit, unless it receives the ”rest no” command.

A restart file can also be saved at any point during a simulation, under any filename. Enter:

```
rest writ  filename.flxrsto
```

To load a previously saved PZFlex restart file, begin a new PZFlex session and in the command prompt enter:

```
rest file filename.flxrsto
rest
```

PZFlex loads the data into memory. You can examine the data or continue executing time steps from this point forward, using the “exec” command.

Restart files contain all data arrays, including mode shape data, as requested with any “shap” command. When you request a restart file with the “rest writ” command, the restart file contains all data including the requested time-history records. Restart files saved by default at the end of a run split off the requested time history data from the `flxrsto` file and save it in a separate PZFlex history file (`flxhst`).

6. Command Override File

It is sometimes necessary to interrupt a simulation while retaining all the data calculated to that point. To do this, use a kill file (`flxkil`), which consists of any “stop/pause” commands. Periodically, PZFlex looks for a kill file of the same base name as the input file (`flxinp`). If PZFlex is running `transducer.flxinp`, for example, it searches for `transducer.flxkil` in the same directory as the `flxinp` file. PZFlex performs this search approximately once per minute.

Once PZFlex finds a kill file, it carries out no further previously issued commands from the `flxinp` file. Instead, it carries out all commands in the `flxkil` file. For example, a `flxkil` file consisting of the single line:

```
term
```

causes control to be returned to the terminal for command input, whereas

```
stop
```

results in the “stop/pause” command cleanly exiting the simulation with all requested data saved.

7. Warning Messsages and Warning File

PZFlex sometimes generates warning messages not because the job needs be terminated, but because a situation has occurred that requires that action be taken. Common examples include requests for color tables for graphics that do not exist, the insertion of real values into integer variables, and requests for plots of time histories that do not exist. Warning messages appear at appropriate points in the print file (`flxprt`). At the end of every simulation, they are combined into a single warning file `flxwrn`, which also notes the number of warnings generated during the run.

Similar warning files are generated during Review and Build runs (`revwrn` and `bldwrn`, respectively).

8. Debugging Error Messages

The majority of errors are code-detected ones. When you receive an error message, check in the printfile (`flxprt`) for the last command issued before the error. This is usually where the error lies.

Following is an example of a code-detected error. The user has tried to print the values of the **σ**yy stress data group prior to the “prcs” command, which defines this data group:


```
 INPUT- PRNT SGYY


        ****CODE HAS DETECTED AN ERROR****
        ****SUBROUTINES TRACKED ARE PRNINP  JOBCTL
        ****ERROR OCCURRED IN SUBROUTINE PRNINP  ****
        ****ERROR MESSAGE--
                           NAME UNDEFINED  ----SGYY


        RESTART WRITTEN FOR TIME=   0.000000        TIME STEP=  0
```

The request for a printout of values of SGYY produced the error. The important line is the “Error Message,” stating that the name “SGYY” is undefined. To correct the problem one must either ensure that SGYY exists before the command is entered (with “calc strs”) or change the name of the requested printout array (e.g., to “pres”).

Any error that occurs when the model is run in batch mode causes the program to terminate. Some command errors, such as requests for unknown variables, cause a prompt for re-entry of the previous command.

Common errors include:

1. Misspelling a command or input parameter
2. Using UPPER CASE text string inputs. Although the manual shows text string parameters as upper case, all commands and text string input parameters must be entered in lower case.
3. Lack of "$" before a "symbol" variable name
4. Lack of spaces between all items on a "symbol" command
5. Request for output for an element data array using nodal subscript ranges in stead of element subscript ranges (e.g., having one fewer element than nodes for each I, J, and K index).

9. Managing Memory

Under Windows, PZFlex automatically allocates memory as needed by the simulation, up to the RAM limit or the maximum allocated by OS. The single-precision (default) version is limited to about 6.5 GB of memory, and the double-precision version has unlimited memory. Double-precision versions use twice the memory of single-precision versions.

Linux versions require manual memory allocation. This is done with the "mem" command and must be the first line of any input batch file. Memory is allocated in millions of words. A word in single-precision (default) PZFlex is 4 bytes. Therefore the command:


```
mem 100
```

allocates 100 million words of memory for numerical calculation, for a total of 400 MB.

The default amount of memory allocated can be altered in the PZFlex defaults section of FlexLab, under "Options>>>Flex>>>Flex Defaults."

Character memory (space for variable names, etc.) is also under user control but is rarely needed. It is allocated in thousands of words and is specified by the second number of the "mem" command:

```
mem 100 20
```

allocates 100 million words for numeric space, but only 20 thousand words for character space. The amount of memory used in a job can be viewed in the last section of the PZFlex print file (`flxprt`).

PZFlex is an "in-core" solver, i.e., all calculations are done in memory rather than on the hard disk. Simulations would run thousands of times slower if the hard disk were used instead.

To give a general idea of the amount of memory required for a simulation, a standard 3D model of mechanical elements uses approximately 1 GB RAM per 10 million elements of the model.

Reducing memory requirements

Each array in PZFlex requires memory. For most arrays, the number of entries equals the number of nodes (velocity, displacement) or the number of elements (pressure, stress). Each value in the array needs 4 bytes of memory, so in a million-element model, almost every array needs 4MB. Given that all direction properties (such as velocity) have 3 components (x,y,z) and others such as stress and strain require up to 6 components, the amount of memory required can expand rapidly. By default, PZFlex saves a minimum of such data and other arrays must be requested explicitly. For example, PZFlex always calculates velocity but not displacement. Use “calc” to request displacement, e.g.:

```
calc
  disp
  end
```

The amount of memory required typically grows linearly with the number of elements in the model. The exception is any section that includes an electric window (“piez wndo”). In that case, the memory needs grow much faster; it is therefore a good idea to minimize electric window size whenever possible. If you run out of memory due to the electric window, consider using an iterative solver instead of the default direct solver. For example:


```
piez
  slvr cgds
  end
```

Other memory drains are shape functions (“shap”), both for every frequency requested and for additional fields to be operated on, and the more advanced damping models (“rdmp” and ”vdmp”).

10. Parallel Processing

PZFlex (beginning with version 1-j.9) supports parallel processing. Computers that contain more than one processor can use this SMP (Symmetric Multi-Processor) capability to reduce the amount of time required for a simulation. This applies to multiple-socket systems and multi-core processors.

Multi-processor systems

There are several advantages to using additional processors with PZFlex. A single simulation using PZFlex (or Review) can be executed while leaving the system available for other tasks, such as wordprocessing or viewing AVI movies. Multiple simulations using PZFlex (or Review) can be executed simultaneously with no significant performance degradation.

Finally, less computation time is required to execute an individual PZFlex simulation.

When a system has more computationally intensive tasks running than the number of processors available, it becomes unresponsive and operates inefficiently. In such cases, it is more efficient to postpone any "extra" tasks until the primary tasks have finished; in other words, to run the tasks in *series* rather than in *parallel*.

Standard PZFlex is licensed for up to two (2) processors on a single system. To expand your license to support more processors, please contact PZFlex support.

Multi-processing commands

To multi-process a model, place the following MP (multi-processing) commands at the start of the PZFlex input file. [Note: You can change the number of processors for a restart run.]

```
mp
  omp 2
  end
```

tells PZFlex to use 2 processors to compute the response for that particular simulation.

Multi-processor efficiency

For the types of problems PZFlex typically solves, MP is not 100% efficient, i.e., the use of a second processor in a PZFlex simulation does not halve the computation time. The reduction in computation time varies with problem type and size, hardware, and operating system. Typical PZFlex simulations run in MP mode with 70% to 90% efficiency.

Problem type

Certain types of models make better use of MP operations than others. Mechanical-only elements that are used for elastic solids, fluids, and gases usually show the greatest gains in performance, while models with large electric or thermal windows show the least gain.

Problem size

The extra overhead to run a calculation in MP mode is usually negligible relative to the gain in performance. In the case of very small models, however, the calculation may not be large enough for the second processor to be effective, and the problem may run no faster than in single-processor mode.

The larger a problem, the larger the potential gain from MP. Note that the number of time steps for a given model does not affect MP efficiency; a model that runs with poor MP efficiency for 100 time steps runs with the same efficiency for 100,000 time steps. "Size" should be thought of as the number of elements or length of time required to solve an individual time step.

11. Thermal analysis

PZFlex includes a transient thermal solver that supports 2D plane strain, 2D axisymmetric, and 3D models. As with piezoelectric simulations and the electric window, the thermal solver uses a single thermal window for thermal analyses. By default, this region includes the entire model, although it can be limited to a subset of the grid. Again, like the piezoelectric window, the region is computationally more intensive than simple mechanical elements and should be kept as small as practical.

Thermal boundary conditions are applied to the thermal window rather than to the entire grid, using the "boun" command. These conditions include temperature, heat flux, linear radiation, nonlinear radiation, convection, and infinite domains. Power deposition per unit volume can also be specified. Thermal properties (specific heat capacity and conductivity) must be defined for materials in the thermal window. This is done using the "matr thrm" command. Perfusion options are available for modeling fluid and blood flow.

Thermal solvers are activated with the "heat" command. Although the thermal and mechanical domains can be coupled (with the "cupl" subcommand), the mechanical and thermal timescales usually differ by a factor of 1 million or more. For most calculations, therefore, it is computationally more efficient to decouple thermal and mechanical domains, i.e., run them separately.

For thermal simulations, mesh density typically does not need to be as fine as for mechanical simulations. The mesh must be dense enough to resolve thermal gradients effectively. It can be coarse in regions of little change while fine in regions of rapid heat change (e.g., focal regions). Unlike mechanical models, where the time step is limited by stability, thermal solvers are unconditionally stable. Use the "time" command to choose the time step of the model. The time step should be small enough to resolve any rises or falls of temperature over time. For example, a 1-second time step would be inappropriate for a model where temperature changes are expected to occur over 2 seconds; a value of 0.1 second per time step would be better.

At each time step, the array TMPR (i,j,k) contains the temperature at each node in the thermal window. As with any array, field plots, time histories, and so on are available.

To perform a decoupled electro-mechanical-thermal analysis:

1. Execute an electromechanical model and record accumulated energy dissipation via the "calc loss" subcommand. This accumulation should be over a representative time interval, e.g., 1 cycle or 1 duty cycle (pulse).
2. Save the model geometry and loss array to a file, using "data out" commands.
3. Build a thermal model with the same geometry. The thermal grid is typically, though not necessarily, coarser than the mechanical grid.
4. Use the "boun defn" command to specify the time variation of power deposition. Although this is usually a constant value, if the duty cycle is long compared to the

thermal time step, the on/off behavior can be represented explicitly. Set the scale factors to divide the loss array by the appropriate time interval (e.g., 1 cycle or 1 duty cycle) for the correct average power in the thermal model.

5. The “boun mshp” subcommand interpolates the loss array onto the thermal grid.
6. The thermal directory in the Annotated Examples describes this procedure in more detail.

12. Further Notes on Extrapolation

Simulating echo return from a rigid reflector

PZFlex can compute the echo response of a transducer to a send signal reflected by a rigid reflector located at some distance from the transducer (see Fig. 55). This applies to both 2D and 3D models. The best approach is to break the problem into separate “send” and “receive” calculations. Both use basically the same physical model but different boundary conditions and other command options. The “send” model computes the return signal at the top of the model. The “receive” model uses the computed return signal as an applied boundary condition on the top surface of the model and computes the model’s response to the signal.

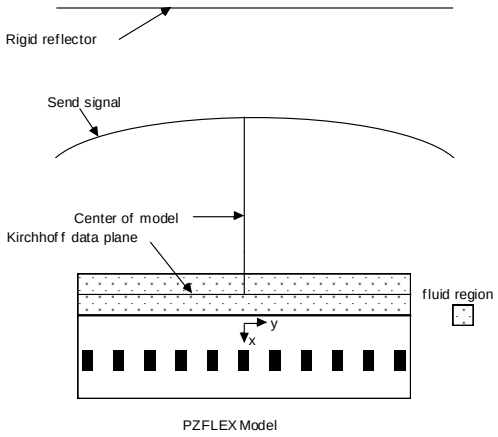


Figure 55—The Pulse/Echo Problem

To compute the return signal during the “send” calculation, define a “Kirchhoff data plane” spanning the entire model, using the “echo” command group. The “echo” option automatically computes the echo return signal using Kirchhoff extrapolation during the “send” calculation and saves a free-field boundary file (with the “flxffl” identifier), which drives the “receive” calculation.

The following approximations are implicit in this pulse/echo analysis:

1. The return signal is assumed to be caused by the presence of a flat, rigid, infinite reflector.
2. The computed return signal is the signal that would occur at the centerline of the model, directly above the driven transducer. It is assumed to be a normally incident plane wave at the top of the model. This assumption requires that the radius of curvature of the wave front of the return signal be large enough that a plane wave assumption is acceptable.
3. The pressure outside the region spanned by the Kirchhoff data plane is assumed to be zero. Therefore, for an accurate evaluation of the return signal, the model should be wide enough for this approximation to be valid.
4. For 2D plane strain models of the transducer region, the computed return signal maintains the 2D plane strain approximation. This results in less attenuation of the signal than if 3D effects were included.

To simulate the “send” and “receive” portions of the problem:

Step 1: Run the “send” model.

1. Construct the model wide enough to capture the primary pressure pulse of a single driven transducer. Zero pressure is assumed beyond the edge of the model.
2. Use the “echo” command group to define a Kirchhoff data plane in the acoustic medium, the reflector location, and other necessary parameters.
3. The Kirchhoff data plane should be located close to the fluid/solid interface to avoid the escape of significant energy through the sides of the model below the data plane. We recommend placing the data plane within a couple of elements of the fluid/solid interface. Also, there should be enough fluid elements above the Kirchhoff data plane that absorbing boundary approximations are not significant at the location of the data plane. We recommend at least one wavelength of fluid from the data plane to the top of the model for the frequencies of interest.
4. Save the time history of the echo return pressure for later plotting. The “pout” command group below is used to request this time history. The “echo” data array contains the last computed value of the extrapolated pressure.

```
pout
hist  echo
<other time history request>
end
```

Note that the “echo” time history saved to the time history (flxhst) file is stored on the file with the same time-record information as all other time histories requested for the model. Thus the echo return pressure is time-shifted from its actual arrival by the time it would take for a wave to travel from the Kirchhoff data plane to the rigid reflector and back to the top of the model. The value of this time shift is printed in the jobs output (flxprt) file. It can also be assigned to a “symbol” variable with the following “symbol” statement:

```
symb #get { timeshft } echotime
```

where the variable “timeshift” contains the value of the time shift.

5. This calculation produces a freefield boundary (`flxffl`) file, which is used as the top surface boundary condition for the “receive” simulation. Note that the “send” simulation should be run long enough to define completely the echo return signal for the “receive” simulation.

Step 2: Run the “receive” calculation.

Use the same model as for the “send” simulation or a reduced model containing fewer transducer elements. The model must have the same top boundary location and same grid spacing in the vicinity of the top boundary. It need include only enough of the horizontal expanse of the transducer array to capture the periodic nature of the array. This is because the return signal is assumed to be a uniform, normally incident plane wave at the top of model.

For the “receive” simulation, make the following changes to the “send” model:

1. Remove any “echo” and “extr“ command groups.
2. Use the “ffld” (free-field) command group to identify the freefield boundary (`flxffl`) file.
3. Change the boundary condition for the top surface of the model to “ffld.”
4. Make both side boundary conditions “symm” (symmetric). This makes the model periodic in the horizontal direction.
5. Remove any time history request for the echo data array, as it does not exist in the “receive” calculation.
6. Deactivate the driving voltage used in the “send” model.
7. Assign “receive” circuits to electrodes if different from send circuits.
8. By default, the “receive” simulation can be run only as long as the `flxffl` file contains boundary data. Setting the “nocheck” option on the “ffld in” command, however, allows the receive simulation to run beyond the time duration on the `flxffl` file, by assuming that the echo return is zero from the last time on the file onward. The time information on the file is time-shifted the same way as the echo time history in Step 1.

Extracting resonance shapes from transient simulations

PZFlex can extract steady-state resonant response shapes (similar to mode shapes) from transient time-domain simulations. The approach is to drive the model with a broadband transient excitation and simulate the model's response in time until it has decayed to about zero. Use the "shap" command to request the specific frequencies for which resonant shape data are to be calculated (you must know the frequencies of interest prior to performing the simulation). PZFlex performs the analysis during the transient response analysis. At the end of the analysis, the resonant shape information is contained in the restart file for the computation. The information can be displayed for each shape requested at any phase angle, and the model's response can be animated to make clear its behavior.

To compute and display resonant shapes:

Run the PZFlex model and request that resonant shapes be computed:

1. Use the "shap" (shape) command group to request that steady-state resonant shapes be computed during the analysis using a discrete Fourier transform.
2. Use the "node" subcommand to specify the regions of the model for which shapes are to be computed. Use the "freq" (frequency) subcommand to specify frequencies. Use the "data" subcommand to specify any fields for which you wish to take a "shap" (e.g., "pres").
3. Save a restart file for the calculation, i.e., do not use the "rest no" command.
4. Display shapes and/or animate a specific resonant response in the "grph" command.

The "plot shap" subcommand allows you to display the deformed shape of the model at a particular phase angle and magnitude of distortion for any shape computed in Step 1., e.g.:

```
grph
    plot shap 1  90
```

plots mode shape 1 at an angle of 90 degrees. The bimorph example in the Annotated Examples provides further detail. The "disp" command controls magnification of the plot.

Displacement shapes are shown by default. To show precalculated field data, use:

```
grph
    plot shap 1 pres 90
```

To plot the pressure mode shape for frequency, request 1 at 90 degrees.

APPENDIX B
MATERIAL DEFINITIONS

Piezoelectric Materials: Properties for Finite-Element Modeling in PZFlex

Piezoelectric materials are anisotropic; that is, they do not have the same properties in all axes. It is therefore important to have a form of notation that distinguishes clearly which axis is being referred to in any statement regarding a piezoelectric material. In Fig. 56, for example, 3 is the axis of poling (z or thickness), and 1 and 2 (x and y, or width) are the axes perpendicular to the 3 axis. A piezoelectric material is typically, although not always, electroded along the planes perpendicular to the upper and lower ends of the 3 axis. The directions 4, 5, and 6 refer to rotations about the 1, 2, and 3 axes, respectively. The numbers are often used in subscripts of material constants, in the form x_{ij}, where "x" is the

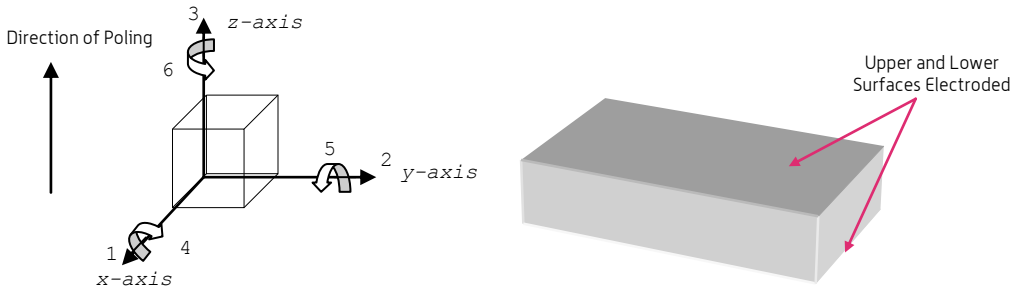


Figure 56—Axis Numbering

constant value, "i" the response, and "j" the applied stimulus.

Two common constants define the piezoelectric behavior of a material under open cir-cuit conditions. The first is the piezoelectric stress constant (e_{ij}), which is the charge produced per unit stress, expressed in units C/m^2. The second is the piezoelectric strain constant (d_{ij}) It is equal to the strain produced by a unit electric field and expressed in units mv^{-1} or CN^{-1}.

Superscript terminology

T	constant stress—mechanically free
S	constant strain—mechanically clamped
E	constant electric field—short circuit
D	constant electrical displacement—open circuit

Piezoelectric material properties

piezoelectric stress constants	[e]	units C/m^2
piezoelectric strain constants	[d]	units C/N or m/V

Less common:

piezoelectric voltage constants	[g]	units Vm/N or m^2/C
piezoelectric current constants	[h]	units V/m

These constants populate the 3-by-6 piezoelectric matrix. Not all 18 constants, however, have non-zero values and some constants are identical. Most piezoelectric matrices have 5 constants with non-zero values, of which 3 are independent values. The values are entered for each material in the "matr" primary command with the "piez" subcommand. The most common form for the matrix are [e] and [d]. PZFlex does not accept [g] or [h] parameters.

Components of [d],[e],[g], and [h] matrices:

	1	2	3	4	5	6
1(or x)	0	0	0	0	15 or x5	0
2(or y)	0	0	0	24 or y4	0	0
3(or z)	31 or z1	32 or z2	33 or z3	0	0	0

Usually the components z1=z2 and y4=x5. Piezoelectric properties are therefore usually provided as three numbers, e.g., for [e]:

$e_{z1} = -9.4$

$e_{z3} = 22.5$

$e_{x5} = 16.0$

The z1 component is a negative number for most piezoelectric materials. By default, PZFlex accepts the piezoelectric stress constants [e], e.g.,

```
piez pmt3  1 5 $ex5  2 4 $ex5  3 1 $ez1  3 2 $ez1  3 3 $ez3
```

or

```
piez pmt3 strs 1 5 $ex5  2 4 $ex5  3 1 $ez1  3 2 $ez1  3 3 $ez3
```

PZFlex also accepts the form of the strain constants [d]. The “matr/piez” line, however, must be altered to reflect this:

```
piez pmt3 strn 1 5 $dx5  2 4 $dx5  3 1 $dz1  3 2 $dz1  3 3 $dz3
```

Dielectric material properties

Dielectric material properties are usually expressed as relative to ε_0, $8.85418782 \times 10^{-12}$ F/m. PZFlex accepts permittivity values only when input as absolute permittivity.

relative permittivity $\varepsilon_r = \varepsilon / \varepsilon_0$

dielectric constant, under constant stress, ε^T, dimensionless

dielectric constant, under constant strain, ε^S, dimensionless

These constants populate the symmetrical 3-by-3 dielectric matrix; 3 parameters have a non-zero value and 2 have a common value. The values are entered for each material in the “matr” primary command with the “elec” subcommand.

	x	y	z
x	xx	0	0
y	0	yy	0
z	0	0	zz

Because usually the components xx=yy, dielectric constants are listed as two numbers, e.g.,

$\varepsilon^S_{xx} = 1306 * \varepsilon_0$

$\varepsilon^S_{zz} = 1200 * \varepsilon_0$

By default, PZFlex accepts the dielectric properties for constant strain [ε^S], e.g.,

```
elec pmt3 $epxx $epxx $epzz
```

or

```
elec pmt3 strn $epxx $epxx $epzz
```

PZFlex also accepts the dielectric properties in the form of the constant stress [ε^T] matrix; the “matr/elec” line, however, must be altered to reflect this:

```
elec pmt3 strs $epxx $epxx $epzz
```

Mechanical properties

Mechanical properties are usually provided under constant electric field (E) but sometimes

under constant electric displacement (D). PZFlex assumes the constant electric field conditions $[c^E]$ and $[s^E]$.

compliance	[s]	units Pa^{-1}
stiffness	[c]	units Pa or N/m^2

The use of [s] for compliance and [c] for stiffness are not typographical errors! These constants populate the symmetrical 6-by-6 stiffness or compliance matrix. Of the 21 possible parameters for most common piezoelectric materials, 9 have a non-zero value and 6 independent values. The values are entered for each material in the "matr" primary command with the "type lean" and "prop" subcommands.

Components of [c], or [s] matrices:

	1	2	3	4	5	6
1	11	12	13	0	0	0
2	21(s)	22	23	0	0	0
3	31(s)	32(s)	33	0	0	0
4	0	0	0	44	0	0
5	0	0	0	0	55	0
6	0	0	0	0	0	66

Usually the components 11=22, 13=23, and 44=55 and stiffness and compliance properties are given as 6 numbers. Symmetrical components are always equal, so 21=12, 31=13, and 32=23. These symmetrical values need not be entered into PZFlex, e.g., for [c]

c11 = 1.14e11
c12 = 0.757e11
c13 = 0.724e11
c33 = 1.11e11
c44 = 2.63e10
c66 = 1.92e10

Note: c_{66} can sometimes be estimated from $((c_{11}-c_{12})/2)$

By default, PZFlex accepts the constant stiffness parameters [c], e.g.,

```
type lean
    prop pmt3 $rho
    $c11   $c12   $c13   0.0   0.0   0.0   $c11
    $c13   0.0    0.0    0.0   $c33  0.0    0.0
    0.0    $c44   0.0    0.0   $c44  0.0    $c66
```

or

```
type lean
stif prop pmt3 $rho
    $c11   $c12   $c13   0.0   0.0   0.0   $c11
    $c13   0.0    0.0    0.0   $c33  0.0   0.0
    0.0    $c44   0.0    0.0   $c44  0.0   $c66
```

PZFlex also accepts material properties in the form of the compliance constants [s]. The ”matr/prop” line, however, must be altered to reflect this:

```
    type lean
comp prop pmt3 $rho
    $s11   $s12   $s13   0.0   0.0   0.0   $s11
    $s13   0.0    0.0    0.0   $s33  0.0   0.0
    0.0    $s44   0.0    0.0   $s44  0.0   $s66
```


A piezoelectric material can therefore be defined with the following:

```
type lean
    prop pmt3 $rho
    $c11   $c12   $c13   0.0   0.0   0.0   $c11
    $c13   0.0    0.0    0.0   $c33  0.0   0.0
    0.0    $c44   0.0    0.0   $c44  0.0   $c66

    piez pmt3  1 5 $ex5  2 4 $ex5  3 1 $ez1  3 2 $ez1  3 3 $ez3
        elec pmt3 $epxx $epxx $epzz
```

where the “$” value indicates that the alphanumeric characters that follow define a variable.

PZFlex also includes an attenuation, or loss, component. Most manufacturers list this as a mechanical quality factor, Q. Materials with high Q, such as PZT4, are usually in the 500 to 1000 range; they are generally used in high-power applications such as naval sonar. PZT5H-type materials, used in medical imaging, are often in the 50 to 100 range, and materials such as lead metaniobates, which have low cross-coupling, often have Q factors below 20. A high-Q material continues to “ring” for a long time after being excited, whereas a low-Q material quickly returns to steady state.

PZFlex usually uses the “rdmp”’ subcommand for Rayleigh damping. This combination of mass and stiffness damping represents piezoelectric material damping. The value can be set for both longitudinal and shear damping; most manufacturers, however, give only one Q value. Q is measured and defined at one frequency (e.g., 1MHz) and usually varies linearly with frequency.

```
rdmp pmt3 $freqdamp q  $qdmp  $qdmp 1.e6 1.
```

In the above attenuation statement for the “pmt3” material, the “freqdamp” variable sets the frequency value for Q. This is necessary, as attenuation cannot be matched perfectly at all frequencies and is usually the center frequency of the device. Q indicates that the numbers that follow refer to a quality factor (alternatives are dB and % of critical damping). The next two variables are “qdmp”; they are Q values for longitudinal and shear. The following number is the value at which the Q factor was measured. The final number is the power to which the Q factor varies with frequency; 1, which is most common, is linear.

Fig. 57 shows the requested damping and actual damping, as well the user’s attempt to best match the two across a range of frequencies of interest.

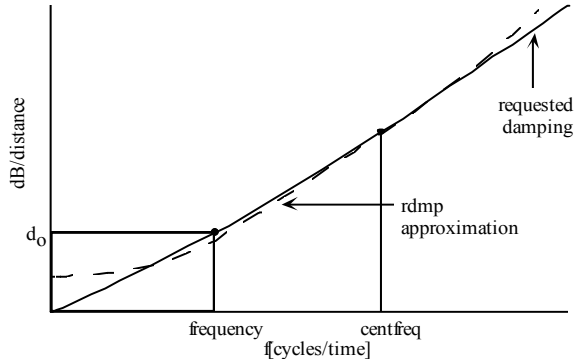


Figure 57—Requested and Actual Damping against Frequency

PZFlex follows the below equation across as wide a frequency range as possible. This equation is the most common way of expressing attenuation in the ultrasound community.

$$dB/dist = d_0\left(\frac{f}{frequency}\right)^{\exp}$$

Polymer materials

Every piezocomposite contains a passive polymer material. This polymer is an isotropic elastic material; that is, the properties are consistent in all directions. The material property specifications are therefore simpler for such materials. The properties that need to be specified are density, longitudinal and shear velocity (or bulk and shear moduli), longitudinal and shear attenuation, and permittivity. In PZFlex they are most often specified by velocities:

```
type elas
wvsp on
prop epoxy $density  $vellong  $velshear
```

or, for specification by bulk and shear moduli:

```
type elas
wvsp off
prop epoxy $density  $bulkmod  $shearmod
```

Permittivity is a single value, again as absolute permittivity:

```
elec epoxy $perm
```


Finally, attenuation must be set. This is generally specified in dB per unit distance at a given frequency, e.g., 2.5 dB/m @ 1 MHz^x. The "x," which is usually between 1 and 2, indicates how the attenuation varies with frequency. Shear attenuation values are often significantly larger than longitudinal values. PZFlex sets the values with:

```
vdmp epoxy db $freqdamp  db  $attlong  $attenshear  1.e6  $x  $dist
```

The polymer usually contributes significantly to attenuation in a piezocomposite.

Piezoelectric constitutive relations

E	electric field (V/m), 3 components (x, y, z)
D	electrical displacement (C/m^2), 3 components (x, y, z)
T	stress (N/m^2), 6 components (xx, yy, zz, xy, yz, xz)
S	strain, 6 components (xx, yy, zz, xy, yz, xz)

$$[T]=[c^E][S]-[e][E]$$
$$[D]=[e][S]+[\varepsilon^S][E]$$

or

$[S]= [s^E][T]+[d][E]$

$[D]=[d][T]+[\varepsilon^T][E]$

Less commonly:

$[S]=[s^D][T]+[g][D]$

$[E]=-[g][T]+[\varepsilon^T]^{-1}[D]$

or

$[T]= [c^D][S]-[h][D]$

$[E]=-[h][S]+[\varepsilon^S]^{-1}[D]$

Mechanical properties

$[c^E]=[s^E]^{-1}$

$[s^D]=[s^E] - [d][g]$

$[c^D]=[c^E] + [e][h]$

Piezoelectric properties

$[e]= [d] [c^E]$

$[d]=[\varepsilon^T][g]$

$[g]=[\varepsilon^T]^{-1}[d]$

$[h]=[g][c^D]$

Dielectric properties

$[\varepsilon^T]= [\varepsilon^S]+[d][e]$

Manufacturers' material properties

Accurate modeling of piezoelectric materials requires 13 independent parameters:

Density—ρ

6 stiffness [c] (or compliance [s]) components: c_{11}, c_{12}, c_{13}, c_{33}, c_{44}, c_{66}

3 piezoelectric components ([d] or [e]): e_{x5}, e_{z1}, e_{z3}

2 dielectric components ($[\varepsilon^T]$ or $[\varepsilon^S]$: ε_{xx} and ε_{zz}

mechanical Q (dimensionless, entered in one of the "matr/xdmp" commands)

It is important to check which values you are using, e.g., stiffness or compliance, or constant stress or constant strain, and to enter the values correctly.

When you enter a matrix in a given form, PZFlex calculates the alternate form and displays those values in the printfile. When you enter the stiffness ([c]) matrix, for example,

PZFlex calculates the compliance [s] matrix and displays it in the printfile (`<jobname>.flxprt`).

Although some manufacturers provide all the necessary information (e.g., Ferroperm, http://www.ferroperm-piezo.com/), others do not. In those cases, it is necessary to fill in the blanks" and approximate unknown values by comparison with values for similar, known materials.

Manufacturers' data usually state a margin of error on the values, e.g.:

stiffness: +/- 5%

piezoelectric: +/- 10%

dielectric: +/- 20%

The values vary from manufacturer to manufacturer.

The methodology for entering material properties assumes that the material in question conforms to a tetragonal (4mm) system. This covers the majority of materials used in piezoelectric modeling, including PZT5H. Some materials, such as quartz, may require additional components from the various matrices.

APPENDIX C
FILES AND FORMATS

Interaction with PZFlex takes place through input and output files. The file options provide flexibility in job setup, execution, monitoring, and post-processing. There are four file types for inputs to the code and eight file types for storing code output. PZFlex uses Fortran “open” and “close” statements to open and close i/o files internally. To develop preprocessing or post-processing capabilities that access the i/o files, you should be familiar with the following information:

Binary files are written with Fortran code. Fortran begins and ends each binary record with a 4-byte integer, which contains the number of bytes in the record. It is automatically accounted for when Fortran is used to read the file, but must be explicitly included for C code.

The following format descriptions indicate integer numbers as type (I), floating point numbers as type (F), and character strings as type (C). Most PZFlex programs use 4-byte floating point and integer values. Certain versions of PZFlex, however, run in full double-precision (8-byte) mode. The double-precision versions use 8-byte words for both real and integer data. Therefore, when file format descriptions show variables of type (F) or type (I), their size will be 4 bytes unless the files were written by a double-precision version of the code, in which case they will be 8 bytes.

Input file types:

1. job
2. grid
3. data
4. site
5. restart

Output file types:

1. job
2. command log
3. result
4. snapshot
5. time history
6. data
7. extrapolation
8. PostScript
9. image
10. restart
11. text
12. Tiff
13. XWD (X window dump)
14. AVI

APPENDIX D
USING COMMANDS

Using the FUNC command

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.

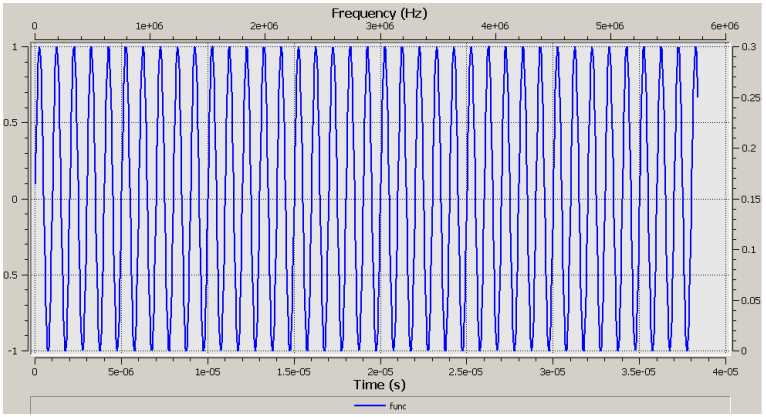
Note: All waveforms can be obtained from time histories of the input function using the INSIGHT tool in PZFlex

Continuous Sine Wave

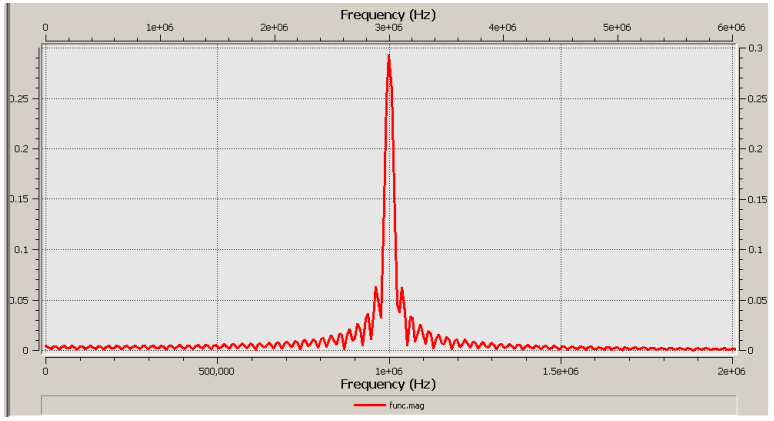
```
func sine $freqint 1. 0. 0.
```

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 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.



Continuous input wave function



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

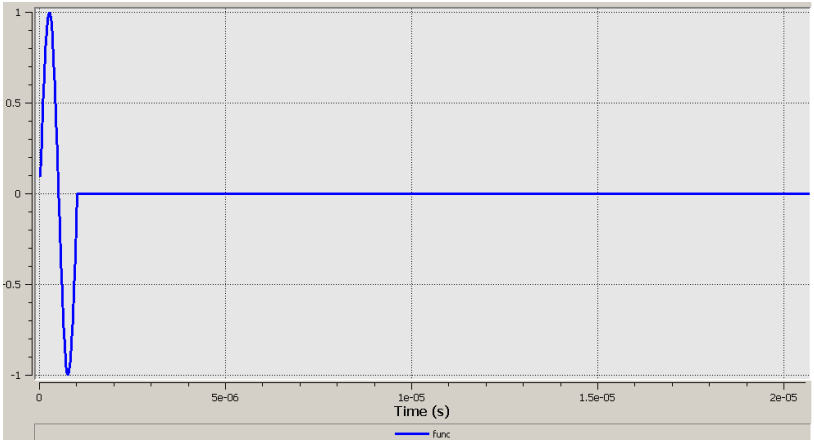
Sine Impulse

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 'Continuous Sine Wave' except from the number of periods/cycles.

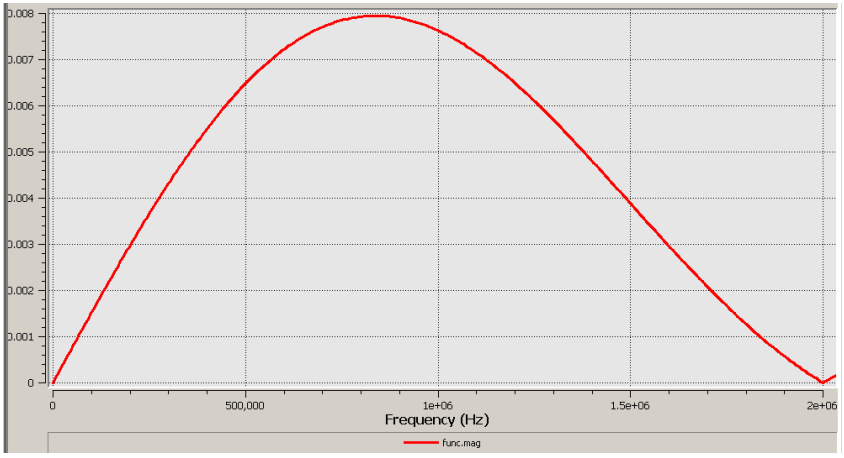
```
func sine $freqint 1. 0. 1.
```

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.



Impulse function consisting of a single sine wave cycle



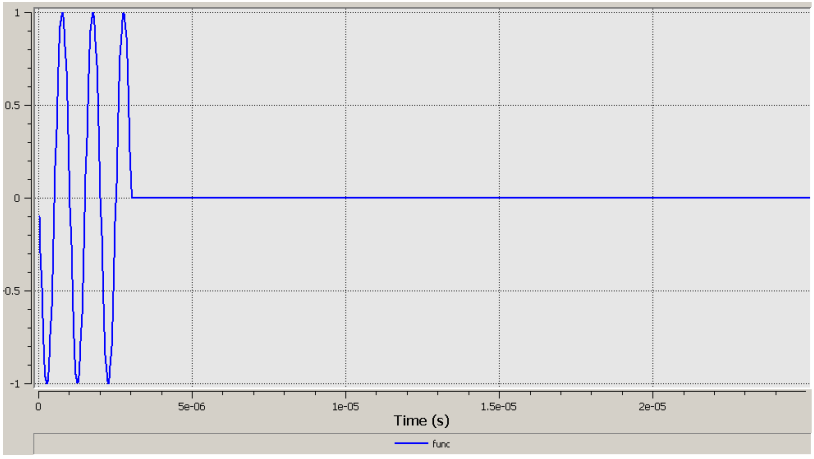
FFT of the sine wave impulse

Three Cycle Burst with Phase Inversion

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

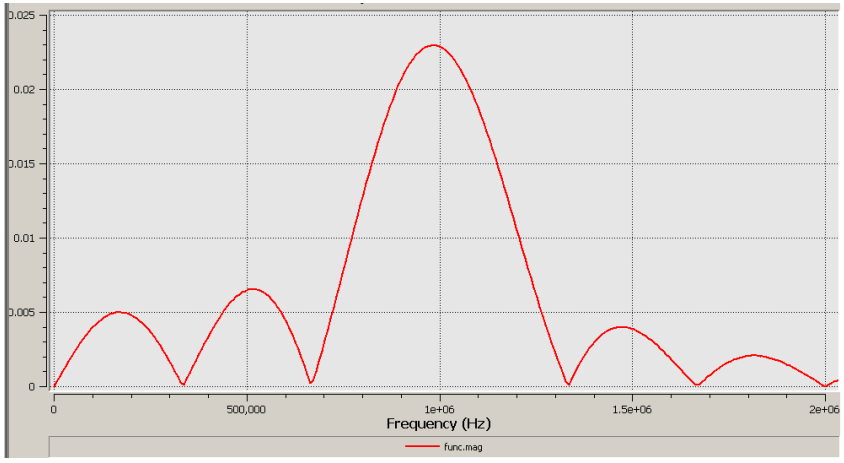
```
func sine $freqint 1. 180. 3.
```

Modified phase and cycles parameters



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.



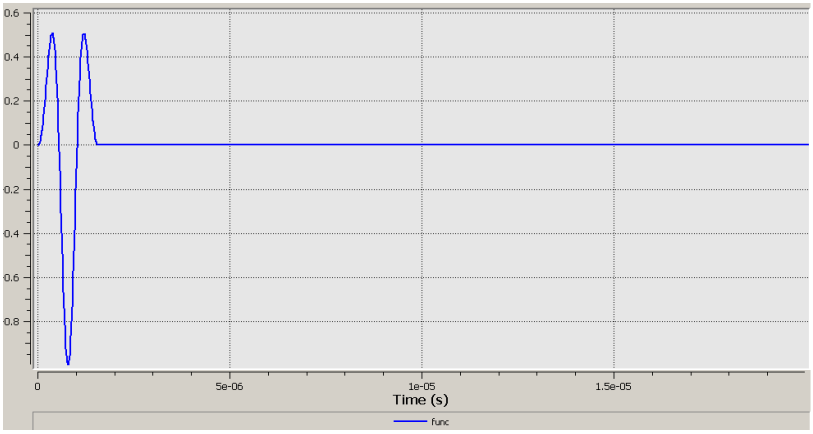
FFT of the 3 cycle burst in Figure 47

Blackman Harris

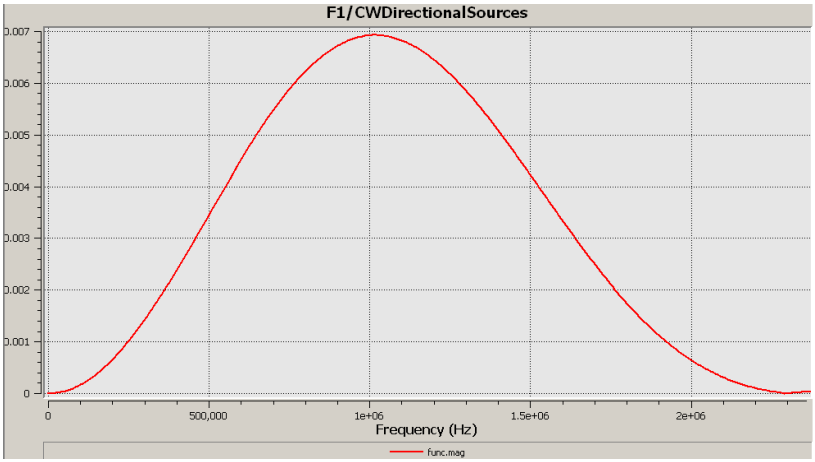
When using a different function, all that is required is to identify the parameters that will change the fundamental properties. For the BLAK function there are only 3 parameters: centre frequency, amplitude and time delay. Time delay does not need to be stated in the line of code as a default value 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.
```

Defining Blackman function



Blackman Harris input signal



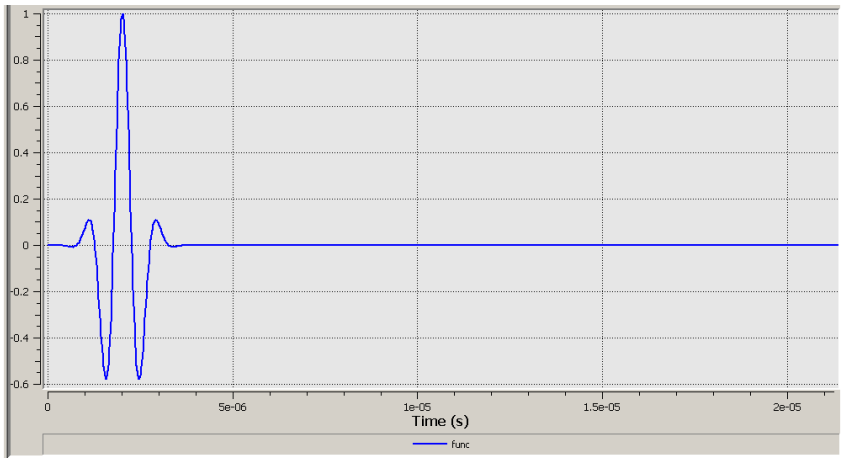
Frequency content of the Blackman Harris signal

Wavelet

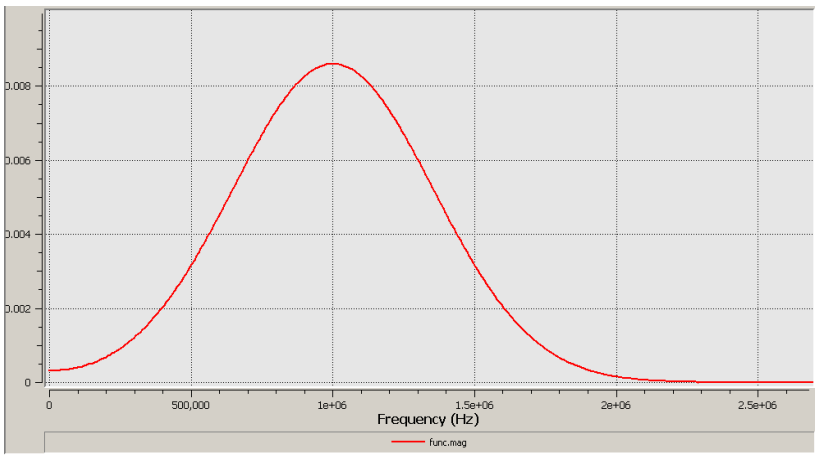
Wavelets are of great use when it comes to signal processing. They have specific properties that when they are convolved with an unknown signal, information can still be extracted. Again, using the wavelet function is similar to all the others: enter the function's name followed by the appropriate parameters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

```
func wvlt $freqint 1
```

Using the wavelet function



Wavelet waveform



FFT of wavelet

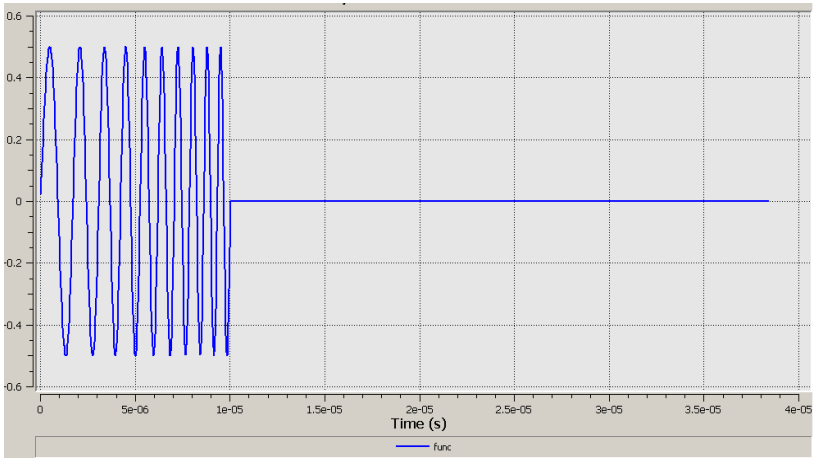
Chirp

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.

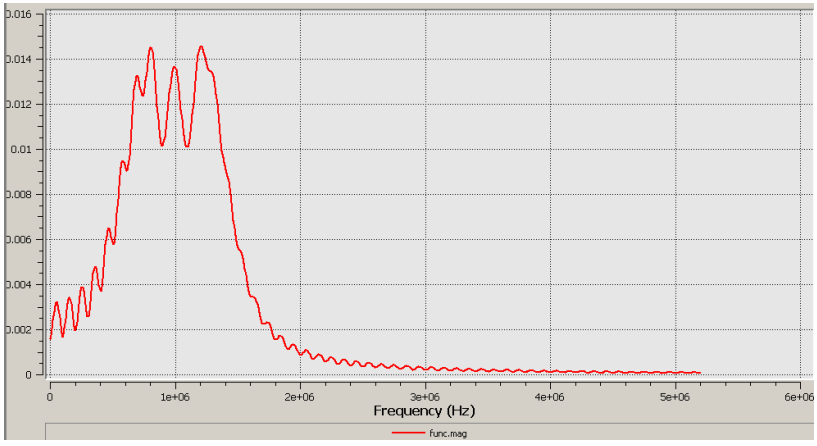
```
func chirp 1. 0. 500e3 1.5e6 10e-6
```

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.



Up-chirp signal beginning at 500 kHz to 1 MHz



FFT of the up-chirp in Figure 55

Using the PLOD command

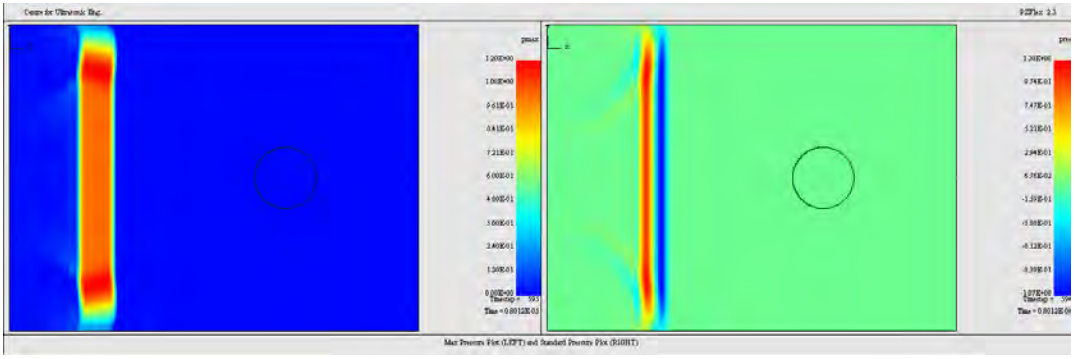
Introduction

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 as 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 are 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 but may refer to other related sections of code.

Wave Applied from Sides of Model



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 above. 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
```

Applies pressure load to the model as shown in Figure 57

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 as the driving function for the model, 'func' should be entered as the pressure time history which is the 2nd parameter.

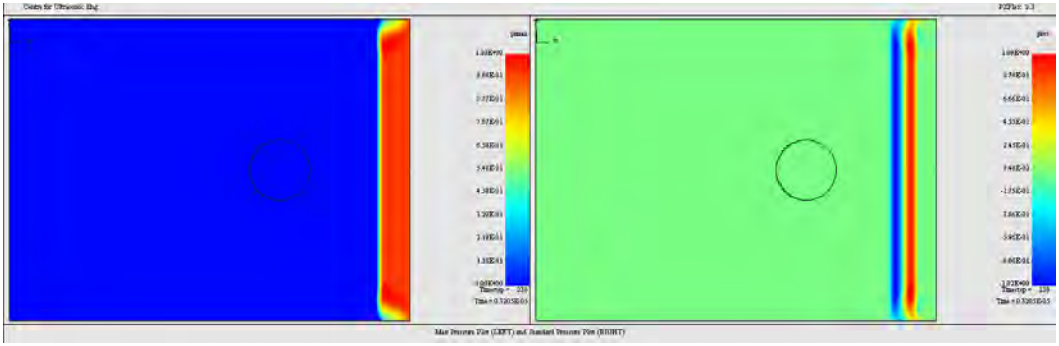
vctr <vectorname> <vectorX> <vectorY> <vectorZ>

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 it should look **similar** to the previous diagram illustrating the propagating wave.

Wave Applied from different side of Model



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 section 'Wave Applied from Sides of Model'.

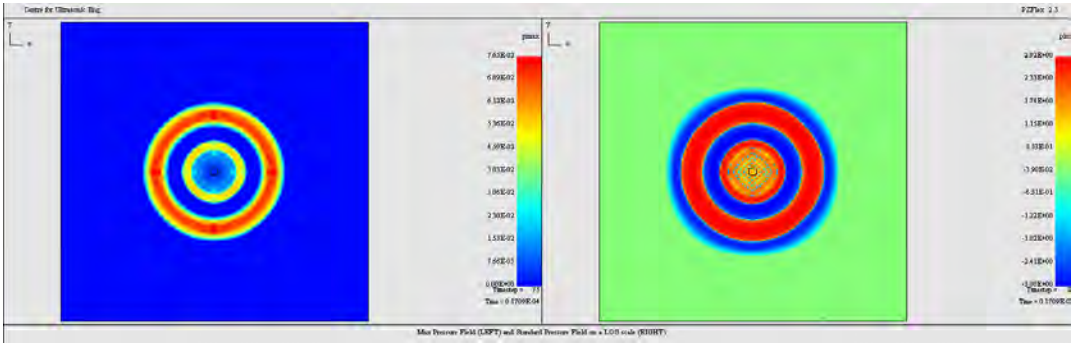
```
plod
        pdef pld1 func
        vctr vct1 -1. 0. 0.
        sdef pld1 vct1 $indgrd $indgrd $j1 $j3
        end
```

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.

Generating a Spherical Wave



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
```

Use of SPOT and SDF2

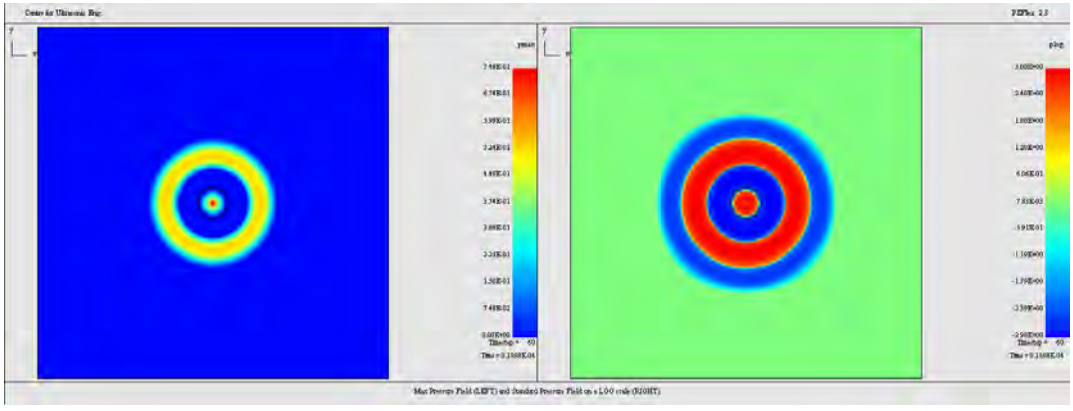
The PDEF command remains the same due to the use of the same driving function. The SPOT subcommand is a tool to identify a surface of nodes to apply a normal pressure loading by simply inputting a known nodal position close to the surface. In this example, the SPOT is situated at the centre point of 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.

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 waves to the defined spheres or cylinders created. This is especially useful when the wedges of the spheres/cylinders are utilized to create more complex boundary shapes.



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 >

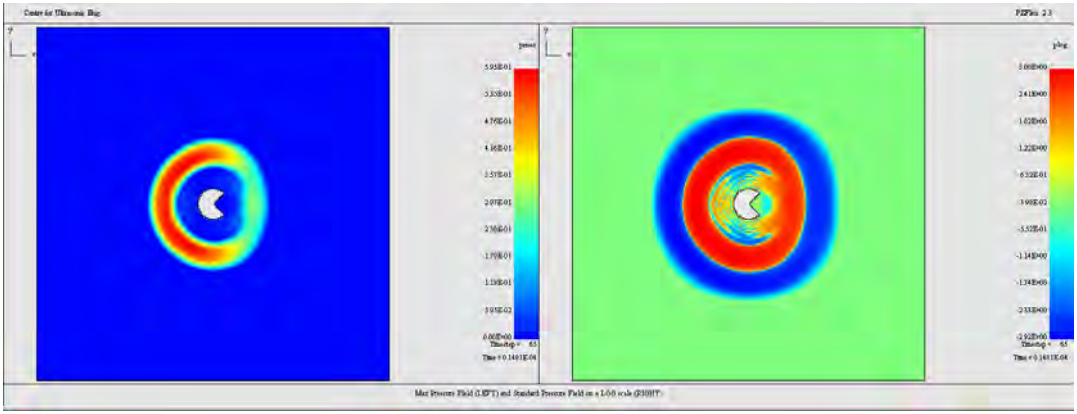
The first 2 parameters are the exact same as the SDEF command.

```
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
```

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 at which the sphere can begin and end with respect to the X and Z axis. This allows the pressure load to be easily applied to both 2D and 3D wedges of a sphere which otherwise would be quite complex.

Using CYLN



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. 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
```

Use of CYLN in both the SITE and PLOD section

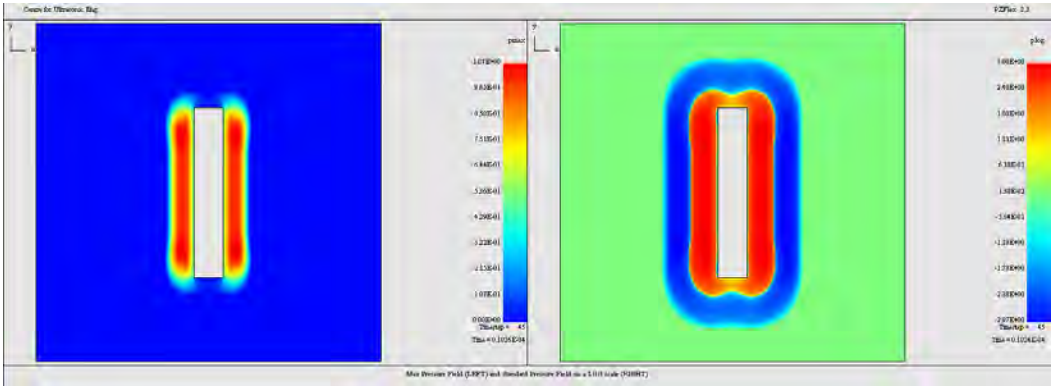
The CYLN subcommand only requires one radius value under the PLOD command and there are no parameters for inserting a hole at the center of the cylinder.

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
```

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.



The cylinder oriented along the Y-axis

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

ANNOTATED EXAMPLES

The Annoted Examples are intended to help the new PZFlex user understand PZFlex code, model-building, and post processing. They cover the following types of problems:

1D Wave Propagation. Introduction to piezoelectric materials. Covers the application of simple pressure loads to a polymer column, symmetrical boundaries, voltage excitation, and common errors.

Bimorph. Construction and evaluation of a simple bimorph cantilever. Covers fixed boundaries, mode shapes, FFTs, and movie generation.

Stack. Construction of a 2D piezoceramic stack model and its conversion to a 3D model. Compares 2D and 3D modeling and discusses the practical value of each. Also discusses determination of the electrical impedance characteristics of the device and normalization of frequency plots.

Batch. Automated parameter sweeps. A simple 2D model is constructed and the thickness varied across an number of automated runs. Highlights the ability to display multiple time histories on a single graph.

Curved Ceramic. Generating curved surfaces using a Cartesian grid. Discusses absorbing boundaries and the application of electrodes to curved geometries.

Kirchoff Extrapolation. A simple 2D model demonstrates use of the Kirchoff extrapolation method to generate beam profiles and determine the pressure response at a far-field point without the need for a large model. Introduces the use of the Kirchoff extrapolation field and determination of a response at a specified distance via Review.

Sloped. The use of Build for more exotic model geometries. Demonstrates a 2D sloped problem and the importation of the models between Build and PZFlex and the Data command.

1D Wave Propagation Example

To run the PZFlex polymer and piezoelectric examples, first open the command prompt. Change to the appropriate directory and load the PZFlex input files into FlexLAB.

Begin by running the 15-element/wavelength polymer example found under Annotated Examples/1D Wave Prop Example/Polymer. In this case, the wavelength is chosen by taking the velocity in the polymer (2565m/s) and the frequency of the driving function (1 MHz) and using the equation $v = f\lambda$. The PZFlex input file is named prop15.flxinp.

Enter "pzflex1j prop15" in the command prompt and click "enter." The command "pzflex1j" starts the PZFlex program; here prop15 is the name of the input file (you do not need to include the flxinp. but it will run the same if you do). You will see Fig. 58, which shows the input end where the pressure will be applied.

Figure 58—Polymer Sample

The command prompt screen contains a "p>." The "p" stands for primary command. Enter "term" (or "t") to continue. The "term" command temporarily interrupts the batch input file, to allow for user interaction before returning control back to the batch file. This requires two "term" commands. The first one is written into the input file, temporarily interrupting the batch input file, and the second is typed manually into the command prompt to return control to the batch file.

Fig. 59 shows the stress dispersion at the output end of the polymer after application of the driving force. Not all energy arrives with the main pulse.

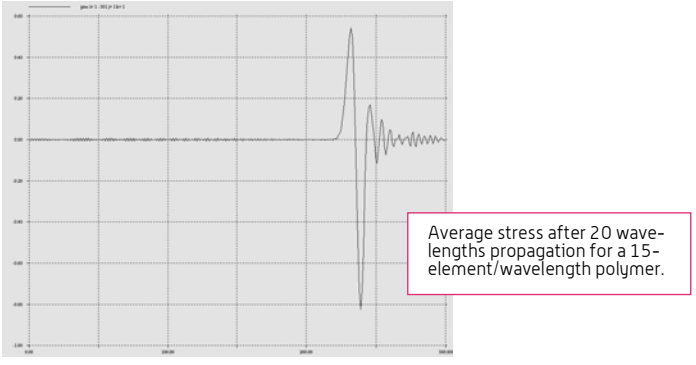


Figure 59—Output End (15 element/wavelength)

You are finished with the `prop15` input file for the time being. Now you will run the input file for the 60-element per wavelength example, `prop60.flxinp`. This is done the same way as the previous example.

Enter "pzflex1j" followed by the name of the input file, in this case `prop60`. First the model will appear, then Fig 60. In Fig. 60 the stress is closer to the ideal sinusoidal function that is the driving function of the system. Although 15 elements per wavelength is usually an adequate discretization, you must ensure that this is the case for all frequencies of interest. Later you will see that the frequency content of the driving function extends significantly past 1 MHz; therefore, the higher discretization of 60 elements per wavelength (at 1 MHz) is necessary to ensure adequate discretization of all propagating waves.

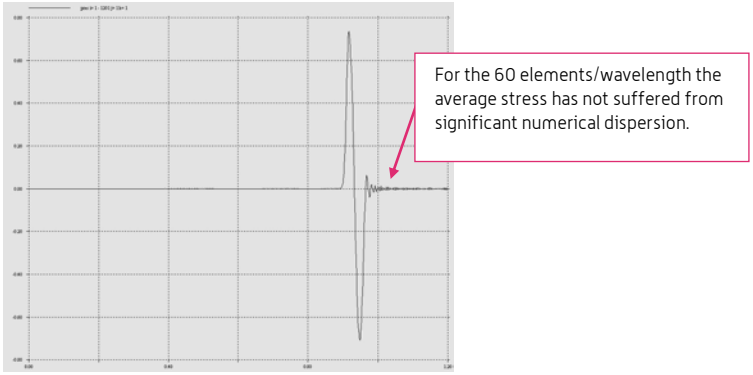


Figure 60—Stress at Output End (60 element/wavelength)

Having run both `prop15 flxinp` and `prop60.flxinp`, you are ready to run the Review file to compare the results of the two simulations. The Windows folder containing the example input files now holds more files than at the beginning. Review files are post-processing files; that is, they are run after the input files. Running Review files is similar to running input files. Instead of entering "pzflex1j," however, enter "review" followed by the name of the Review file. In this case, the review file name is `propcomp.revinp`. In the command prompt, enter "review propcomp" and click "enter."

This runs the Review file. Fig. 61, which contains three graphs, will appear. The solid black lines represent the 15-element/wavelength simulation, and the dashed red lines represent the 60-element/wavelength one. The top graph shows the 1MHz driving function applied to the polymer. Both models have the same driving function. The middle graph shows the displacement at the input end, where the driving function was applied. The bottom graph shows the displacement at the output end, the opposite end from where the driving function was applied. It shows how the displacement of the 15 element/wavelength is dispersed in time, in comparison to the ideal 60 element/wavelength. This is

due to the energy in the higher-frequency signals propagating at different velocities, as a result of inadequate discretization.

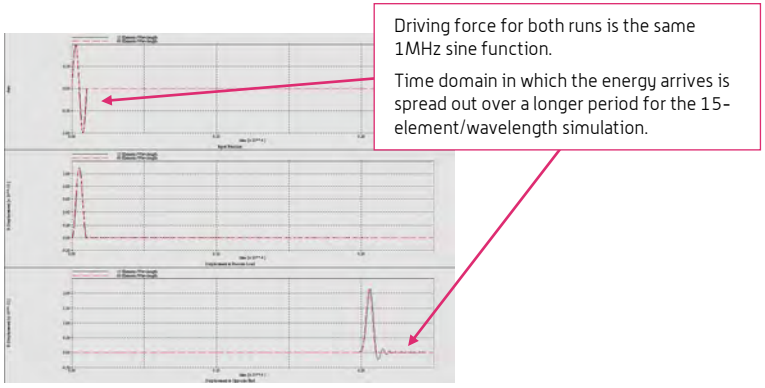


Figure 61—Driving Function and Displacement at Both Ends of Model

The command prompt gives a "p>." Enter "term" to continue. Fig. 62 consists of three graphs, identical to Fig. 61 except that they are "zoomed in" to show detail. The displacement at the near end is the same for both models. At the far end, the 15-element/wavelength model not only moves slightly early compared to the 60-element/wavelength one, it also moves after the 60-element/wavelength model has stopped. Although the dispersion in the 15-element/wavelength model mostly results in later-arriving energy, there is some early-arriving energy, compared to the 60-element/wavelength model

Enter "term" again in the command prompt for Fig. 63. Again there are three graphs with the same solid black line representing the 15 element/wavelength and the dashed red line representing the 60-element/wavelength. These graphs show the frequency domain equivalent of the previous plot. Review has used a fast Fourier transform (FFT) to convert from the time to the frequency domain. The top graph shows the energy input at different frequencies. The nulls occur at multiples of the input frequency or frequency of interest. The middle and bottom graphs show the total displacement for the two models at both ends. Even though the energy in the 15-element/wavelength polymer arrives over a larger time domain, no energy is lost in the system. This shows that the total energy is not only the same at both ends of the polymer, it is the same for the two differently discretized systems. No energy has been lost due to discretization.

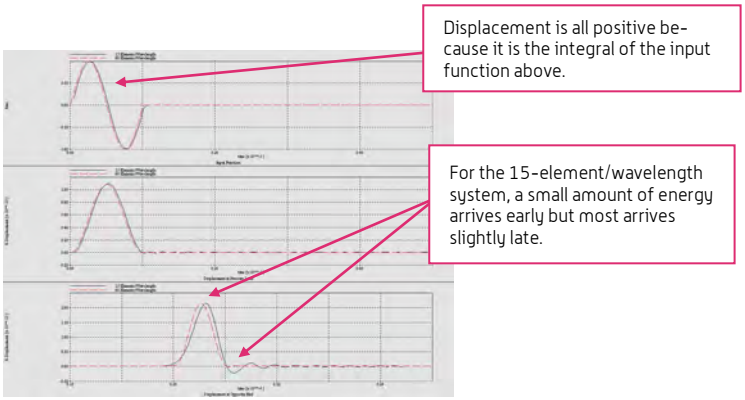


Figure 62—Driving Function and Displacement (zoomed in)

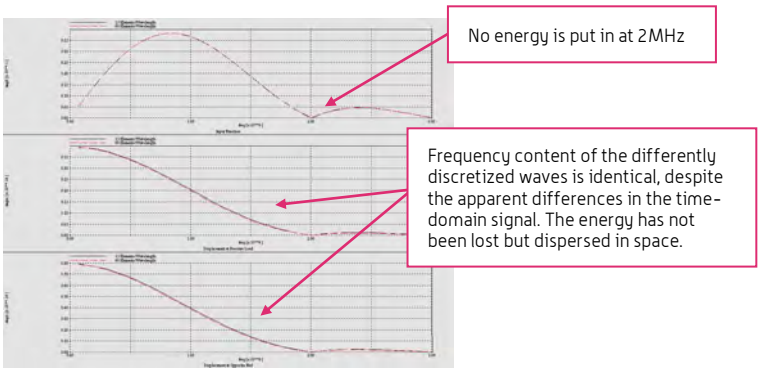


Figure 63—Input Energy and Displacement Spectra

You are now ready to start the piezoelectric example. If the command prompt is not still open from the previous example, open a new command prompt window. Be sure to change the working directory to the folder containing the piezoelectric input and Review files; this is usually `Annotated Examples/1D Wave Prop Example/Piezoelectric`. There are two input files, `propcer.flxinp` and `propcershort.flxinp`; the only difference is that `propcershort` has a much shorter simulation time. There are also two Review files, `propcer.revinp` and `review.propcercomp`. The first is a review of the long runtime input file, and the second compares the short and long runtime simulations.

This example is run the same way as the polymer one. Enter “pzflex1j” followed by the name of the input file, `propcer`. Fig. 64 shows the model, which now has a small piece of ceramic material called “pzt” at the top. This is the piezoelectric part of the model. A voltage will be applied across it to create the driving force.

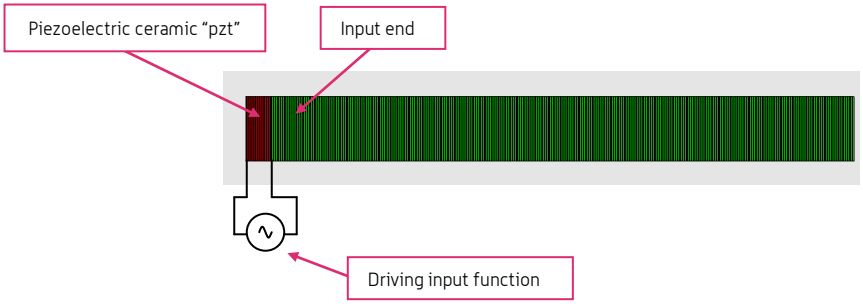


Figure 64—Piezoelectric and Polymer Example

Having run `propcer.flxinp`, you now have the necessary time histories to run the Review part of the simulation. Enter “review” followed by the name of the Review file, `propcer`, in the command prompt. Fig. 65, which contains two graphs, will appear. The upper graph shows the voltage input for the piezoelectric ceramic piece. Like the polymer example, it is a 1MHz sinusoidal function. The lower graph shows the charge on the driving electrode. The charge takes longer to ringdown compared to the period for the voltage cycle. Highly damped devices and devices with well-matched loads ring down faster than lightly damped and poorly matched devices.

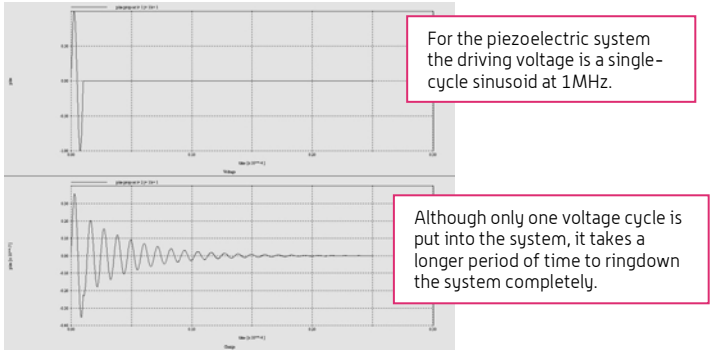


Figure 65—Driving Voltage Function and Charge-Time Plot

Again enter “term” in the command prompt. Fig. 66 consists of two graphs, the electrical

impedance magnitude and phase. The coarse sampling causes a “jagged” appearance, as there are too few data points to represent adequately the impedance curve.

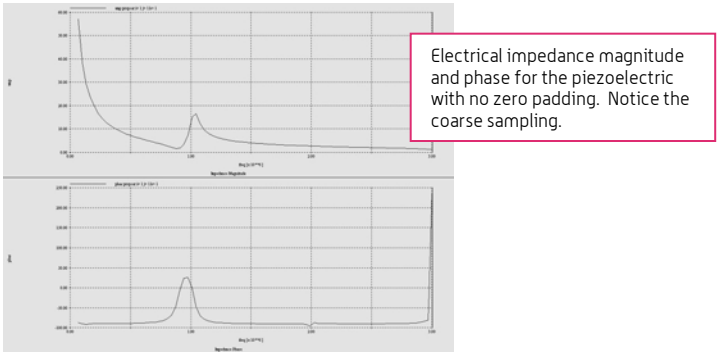


Figure 66—Electrical Impedance Magnitude and Phase

Once again, enter “term” in the command prompt. The graphs in Fig. 67 are similar to those in Fig. 66, except that the curves are smoother. This is the result of zero padding, which adds a number of zeros to the time record before the FFT is performed. This creates more data points when the FFT is calculated. It not only makes the graph smoother but helps to show features that might not otherwise appear. In Fig. 66, for example, in the phase graph at 2 MHz there is a small bump. This is from a gap in the data points, where PZFlex used the points before and after to interpolate how the graph should appear. In Fig. 67, however, zero padding created many more data points and where before there was a bump now there is a large peak. Without zero padding, this peak is less clear. The peak is the result of no energy being put into the system at 2MHz (see Fig. 68). If there is no meaningful input into the model at a given frequency, there is be no meaningful output at that frequency.

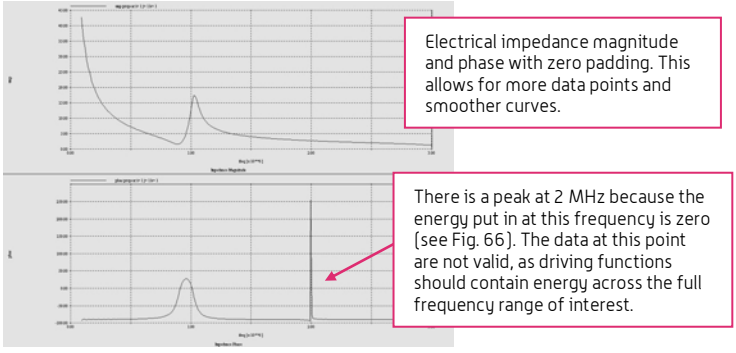


Figure 67—Electrical Impedance Magnitude and Phase (with zero padding)

In the charge time history, NT represents the total number of points. All the points are separated by the same small time step, Δt. Thus the total time in the charge time history can be calculated by multiplying the time step Δt by the number of points NT. For zero padding, zeros are added to the end of the charge time history after the charge has rung down to zero on its own. This does not change the data; it simply increases NT by the same factor as the padding. Adding the subcommand "time pad 3" to the "freq" primary command, for example, increases NT by a factor of three. As the time step remains the same, tMAX, the total time of the charge time history, also increases by a factor of three. The "impd" command uses both the charge time history and voltage time history to generate the electrical impedance frequency response. Both are created by running the initial input file. PZFlex uses three equations to create the electrical impedance graphs:

Equation 1 $$V = I \times Z$$

Equation 2 $$Q = \frac{dI}{dt}$$

Equation 3 $$\int Q dt = \int dI = I$$

In Equation 1,V is the Voltage, I is the current, and Z is the impedance. Equation 2 shows the relationship of charge to current. Equation 3 shows the integration of the second equation, which gives the current. A FFT is then performed on both the current and voltage time histories.

Equation 4 $$Z_{frequency} = \frac{FFT\ V}{FFT\ I}$$

The electrical impedance frequency domain is then calculated using Equation 4, which divides the FFT of the voltage by the FFT of the current. The points on the electrical impedance graph are separated by a small frequency step, Δf. If you call the total number of points on the graph NF, Equations 5, 6, and 7 show how the NF, Δf, and fMAX are calculated:

Equation 5 $$N_F = N_T / 2$$

Equation 6 $$\Delta f = \frac{f_{MAX}}{N_T} = \frac{1}{N_T \Delta t} = \frac{1}{2N_F \Delta t}$$

Equation 7 $$f_{MAX} = 1 / \Delta t$$

In PZFlex, NF is half the size of NT; it may be different with other programs. The frequency step in the FFT result is calculated using Equation 6. The maximum frequency of the elec-

trical impedance graph is determined by taking the inverse of the time step from the charge time history.

The FFT automatically pads NT to ensure a power of 2, 3, or 5 for efficiency (e.g., NT of 1000 is raised to 1024 points). When zero padding is added, both NF and Δf change for any given time history. If the padding is 3, NT is increased by a factor of 3; that is, there are three times as many points in the time-domain signal but Δt remains the same. For the equations above, NF would be 3 times bigger; fMAX would remain the same, as it depends only on the initial time step, Δt, which is constant; and Δf would decrease by a factor of 3, as NT was increased by that much. Thus, although there are now 3 times as many points, the spacing or frequency step between points is reduced. This results in a smoother curve, as data points are now much closer together, while the fMAX remains the same. Thus the graph differs from one without zero padding only in the smoothness not the total size.

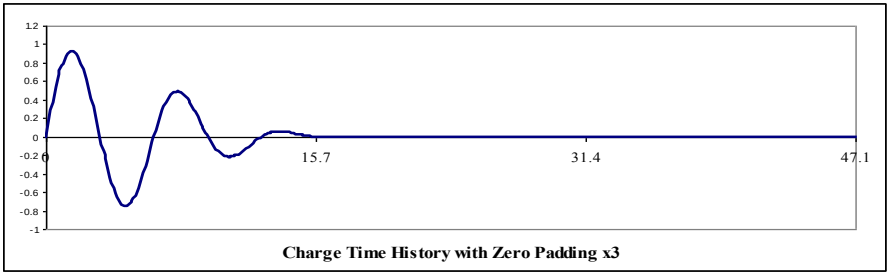


Figure 68

Fig. 68 shows a sample charge time history that has rung down to zero. We have entered a "pad 3" command in the input file, producing 3 times as many data points. If the FFT were displayed as a sinusoidal graph (remember, this is only a hypothetical example, not real data), Fig. 69a shows what it would look like without zero padding. Notice the large Δf between data points. Fig. 69b shows the FFT of charge time history with zero padding shown in Fig. 68. Note that the total period of this sinusoidal graph is the same as in Fig. 69a, except that there are now three times as many data points, resulting in a much smoother curve.

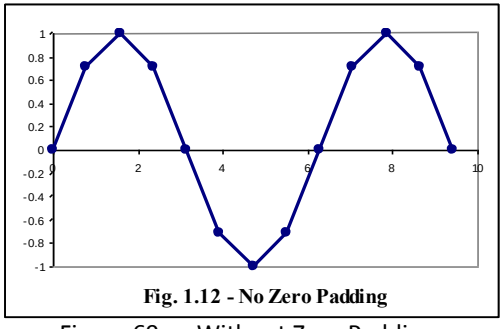


Figure 69a—Without Zero Padding

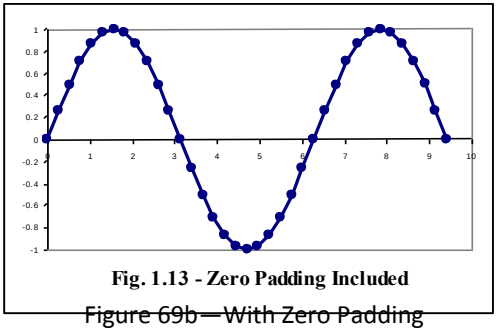
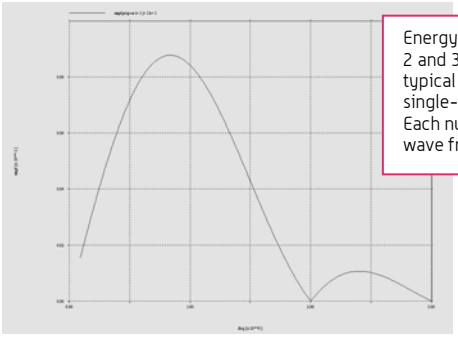


Figure 69b—With Zero Padding

Enter “term” in the command prompt to go to Fig. 70, which is similar to the first graph in Fig. 63. This shows the energy input at various frequencies. Again, there are nulls at 2 and 3 MHz. Each null in the frequency domain graph occurs at multiples of the input frequency. If the input frequency were increased to 1.5 MHz, the first null would be at 3MHz. Fig. 70 is the last graph for this Review file.



Energy input at various frequencies. At 2 and 3 MHz no energy is put in. This is typical frequency domain content of a single-cycle sinusoid driving function. Each null occurs at multiples of the sine wave frequency.

Figure 70—Energy Input for Frequencies 0–3MHz

To end the simulation of the long runtime, enter “term” or “stop” in the command prompt. Before running the Review file that will compare the long and short runtime simulations, it is necessary to run `propcershort.flxinp`. As before, enter “pzflex1j propcershort” in the command prompt. You will see the model you are using (the same model as in the long runtime simulation, Fig. 64). When prompted, enter “stop” to end the input file process. Once the input file has completed, you can run `propcercomp.revinp`. That is the Review file for comparing the long and short runtimes. Enter “review propcercomp” in the command prompt. This produces Fig. 71. The upper graph is the sinusoidal voltage function applied to the model. It is the same graph as the upper graph in Fig. 65. The lower graph in Fig. 71 is the same sinusoidal voltage input function for the short runtime. It is evident that both the long and short runtimes were subject to the same driving force.

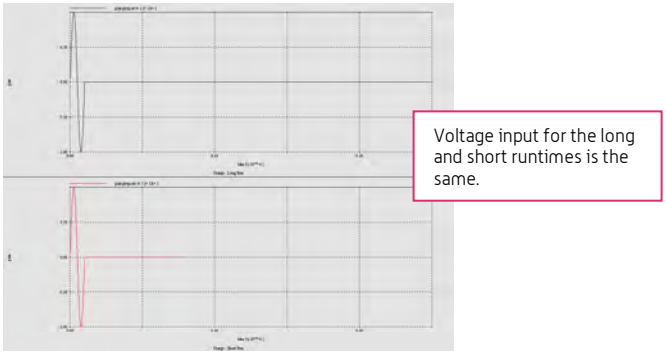


Figure 71—Voltage Input Functions for Long and Short Runtimes

Return to the command prompt and enter “term” again. Fig. 72 shows the charge distribution of the long run and of the earlier, truncated run. In the upper graph, the simulation time is long enough to allow the charge to ringdown to zero with a smooth line. In the lower graph, the simulation time is stopped abruptly, leaving the charge at zero with a sharp angle. As you will see in Fig. 73, this has a dramatic effect.

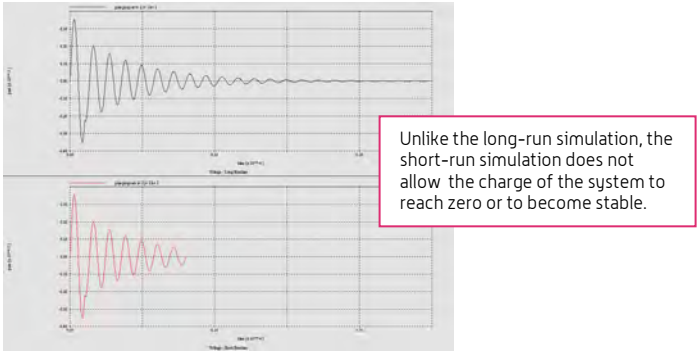


Figure 72—Charge of System versus Time

Again enter “term” in the command prompt. Fig. 73 also consists of two graphs. The red lines represent the truncated charge, and the black lines represent the charge with ample time to ringdown and with zero padding. When the FFT calculates the electrical impedance, it assumes repetition of the charge graph. When the truncated charge stops abruptly, it causes a discontinuity to be introduced into the FFT. This discontinuity results in a “noisy” graph, as in Fig 73. When the charge is left to ringdown, the result is a smooth plot with no discontinuity.

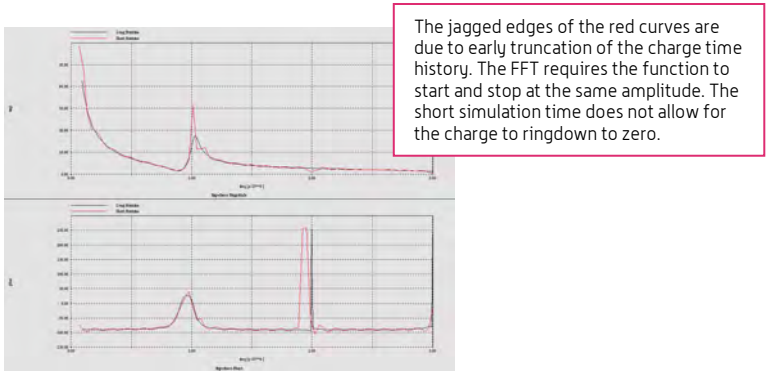


Figure 73—Comparison of Impedance for Long and Short Runtimes

Enter “term” in the command prompt a final time. Fig. 74, again shows the magnitude and phase of the impedance. As in Fig. 73, the black lines represent the long ringdown time with zero padding. Here, the red line again represents the truncated charge, but this time it is a smooth curve. The smoothness results from the application of a windowing function. Windowing functions force the charge to zero by the simulation end time. The windowing function removes from the system the energy remaining when the simulation ends. Although this technique makes a simulation with a truncated charge usable, it is only an approximation. To achieve accurate results it is necessary to let the charge ringdown to zero.

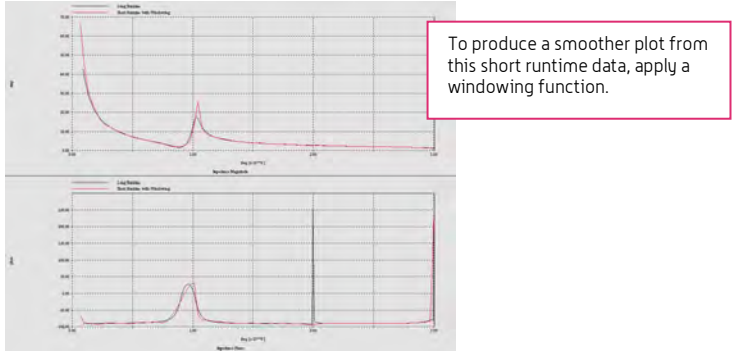


Figure 74—Comparison of Impedance for Long and Short Runtimes (with windowing)

Figs. 75—77 show the effect of a windowing function on a charge time history graph. In Fig. 75 the charge is truncated early and does not completely ringdown to zero. There is still charge in the system when it is cut off. In Fig. 76 the windowing function is applied to the charge time history. The windowing function starts as a straight line at a value of 1, which allow the charge to diminish on its own before being forced to zero from the negative sloped section of the windowing function. Fig. 77 show the result of multiplying the charge time history by this windowing function. The graph starts out the same but is quickly forced to zero. Data points on the zero line continue off the graph. Clearly, the windowing function does a good job of forcing the charge to zero. When the FFT is performed, however, it uses the data points in Fig. 77. These are not the true data of the calculations; they appear on Fig. 75. Thus windowing functions produce only an approximation of the desired results.

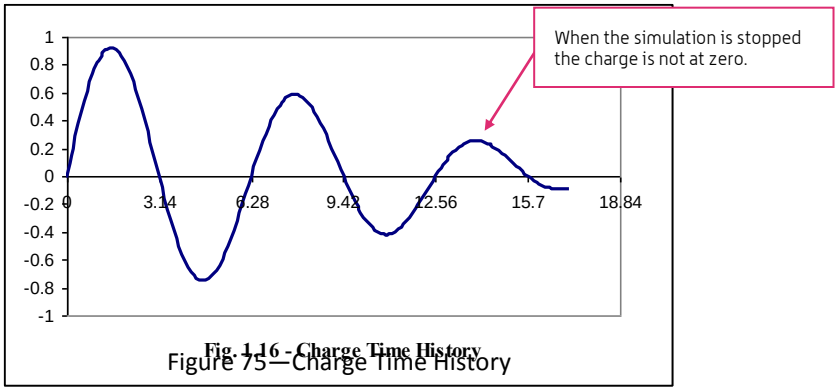


Figure 75—Charge Time History

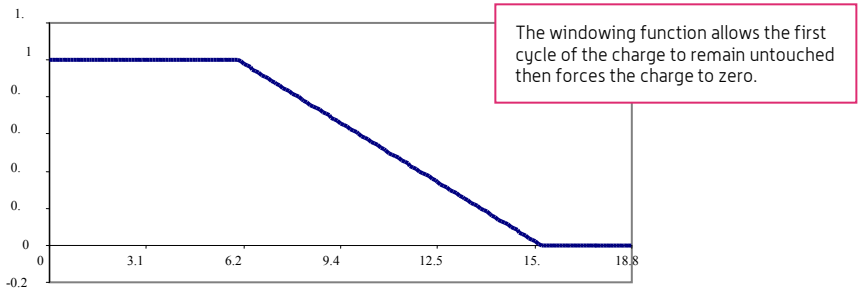


Figure 76—Widowing Function

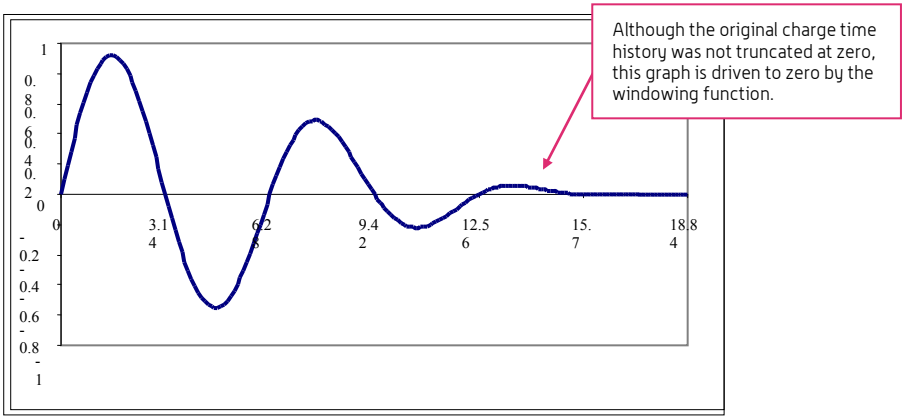


Figure 77—Charge Time History with Windowing

Discussion of ID Wave Propagation Example

The ID wave propagation example demonstrates the importance of proper discretization for accurate results. Without proper discretization, the pressure wave is dispersed through space and does not arrive in one pulse, as it should. It is essential to make sure that the system is discretized for the input frequency of interest. The example also showed that no matter how the system is discretized, the amount of energy at any particular frequency is the same.

The example also demonstrates how zero padding can help smooth out an electrical impedance graph before the FFT is calculated. Sometimes zero padding is needed to show features of the curve that might not be seen as clearly without it.

To obtain results at a particular frequency, energy must be put into the system at that frequency. You cannot put energy in at 1MHz and expect to achieve results at 2MHz. Early truncation of a charge time history can cause a “jagged” electrical impedance curve. The FFT assumes periodic nature of the time-domain signal. If the early truncated charge time history were to repeat itself, there would be a discontinuity between the start and end points. This discontinuity is responsible for the “jagged” curve. One way to solve this problem is to add a windowing function to the charge time history. Windowing functions force the charge to zero before the charge has rung down. It is a way of removing energy from the system to shorten the simulation time. Although this does smooth out the electrical impedance curve, it is preferable to let the charge ringdown on its own. Windowing functions produce a new data set for the FFT to use and an electrical impedance curve that is only an approximation. For accurate results, it is necessary to let the charge ringdown to zero.

Bimorph Example

The bimorph example creates two identical pieces of piezoelectric material, one on top of the other. One end of the model is fixed and the other is free to vibrate. The Review files record the displacement and time histories at the free end.

In the 1D wave propagation example, moving from one graph to another required that you enter "term" in the command prompt. In the biomorph example, the input file is not coded to stop after each graph; it pauses for only a short time before proceeding to the next graph. To control when the program moves to the next graph, you must modify the input file. As you cannot open the input file while PZFlex is running, it is necessary to stop the run if you have already started it. There are several ways to do this. The simplest is to click "Ctrl C" in the command prompt. This terminates the run without saving any of the data that have been created.

A safer way is to create a PZFlex kill file. To do this, create a new text document in the directory that contains the job you are running and enter "stop"; save this file as `flxkil.filename`, where the filename is the name of the input file you wish to stop. Be sure to "save as" without the `.txt` extension. PZFlex checks the directory of the input file for a PZFlex kill file every time step or so and follows the commands in the PZFlex kill file rather than the input file. Using this approach saves the data for all of the completed time steps. Once PZFlex has stopped running, open the PZFlex input file (using a text editor such as WordPad or Notepad). Use the "find" tool under the "edit" drop-down window to search for "grph"; "grph" is the command used by PZFlex to display graphs. The "grph" command will be followed by several subcommands and then finished with the "end" command. After the "end" command, add a new line and enter "term."

The "term" command is used to interrupt temporarily the batch file input, to permit user-interaction before control is returned to the batch file. Do this for all the "grph" commands in the input file. Save the input file and then run `flxinp.bimorp` in the command prompt. You can now move from one graph to the next.

Like the polymer and piezoelectric examples, the bimorph example is run from the command prompt. Open the command prompt, confirm that the directory leads to the folder containing the bimorph input and Review files, and run the input file `flxinp.bimorph`. Enter "pzflex1j bimorph" and click "enter." The first image to appear is Fig. 78, which consists of two identical pieces of piezoelectric material that are colored differently to show that they are separate pieces. The bimorph model is fixed at one end and free to vibrate at the other end. The voltage source is connected to the two electrodes on the top and bottom of the model, and the ground is connected to an electrode between the two pieces of piezoelectric ceramic. All three electrodes cover the entire length of the model.

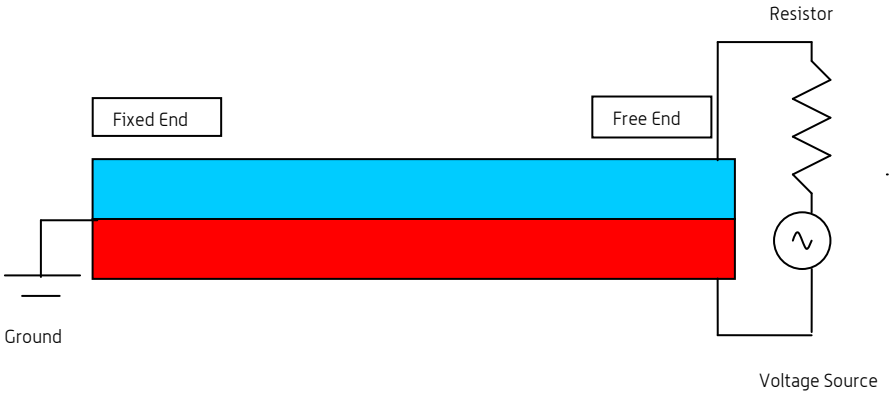


Figure 78—Bimorph Model

Fig. 79 consists of four graphs. The four subwindows were all created using a single "grph" command.

The subcommand "nvew" allows you to create up to eight subwindows in a single window. The left side contains two images. The upper left image is the model being tested. Although the colors are different than in Fig. 78, it is the same model made of the same material. The image at the lower left shows the electric field in the thickness direction. The upper right graph shows the voltage drive function over time. The voltage is applied to the electrodes via an electric source connected to a resistor. The resistor helps the voltage return to zero more quickly than if left to ringdown on its own. The lower right graph shows the displacement of the free end of the model.

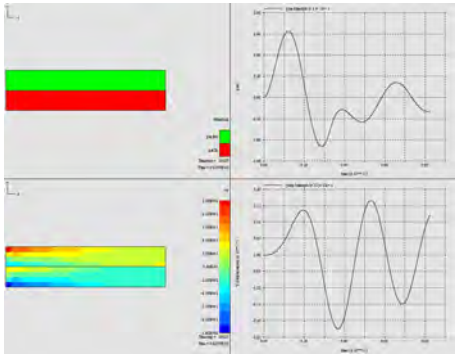


Figure 79—Bimorph Model, Electric Field, Voltage, and Displacement Time Histories

The bimorph example contains ten images similar to Fig. 79. They differ only in terms of the maximum timeframe. Below are two of the ten. On Fig. 80, the fifth graph, the maximum time is 5ms. On Fig. 81, the final graph of the simulation, the maximum time is 9ms. Here the simulation time is long enough to allow the model to reach a steady state in which no further vibrations or voltage are applied to the system.

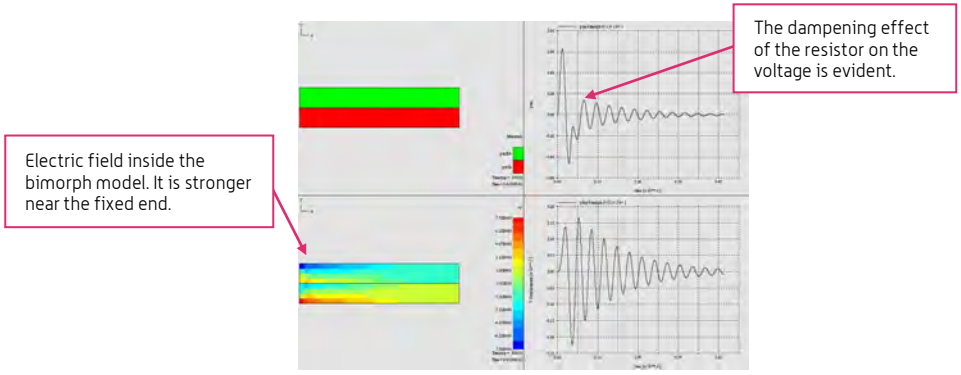


Figure 80—Bimorph Model, Electric Field, Voltage, and Displacement Time Histories

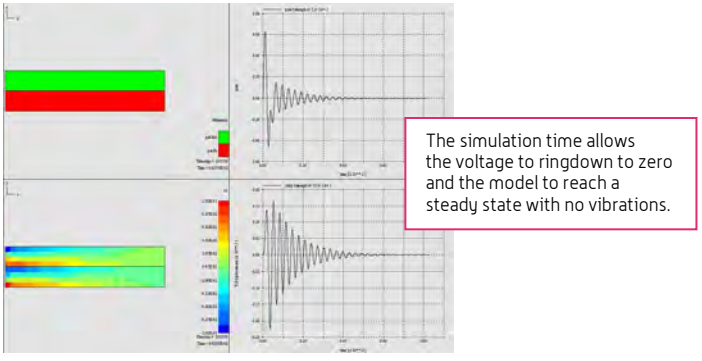


Figure 81—Bimorph Model, Electric Field, Voltage, and Displacement Time Histories

After Fig. 81 a second image of the model appears. showing it being forced upward from the drive function. Enter “term” in the command prompt to see a short animation of the model vibrating under the applied load. The animation can be viewed using Windows Media Player or any other AVI player. This is the end of the input file for the bimorph example.

To run the Review input file `revinp.bimorph`, enter “review bimorph” in the command prompt. It is not necessary to open a new command prompt window, but if you do be sure that the directory leads to the bimorph Review file that you are running. The first figure to appear, Fig. 82, consists of three graphs. The top graph shows the voltage time history, the middle one the charge time history, and the bottom one the displacement at the free end of the model. The simulation has had sufficient time to ringdown the charge and voltage, and the model has reached a steady state with no vibrations.

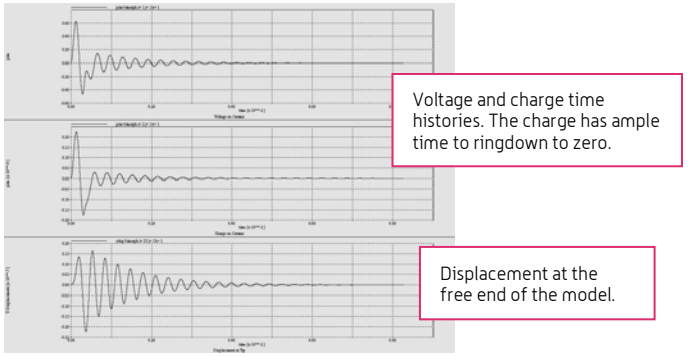


Figure 82—Voltage, Charge, and Displacement Time Histories

Enter “term” in the command prompt. Fig. 83 also consists of three graphs. The first two show the electrical impedance magnitude and phase. The third shows the displacement frequency output. The electrical impedance curves are smooth. Because the charge was allowed to ringdown to zero and then left to run even after reaching zero, no zero padding was needed.

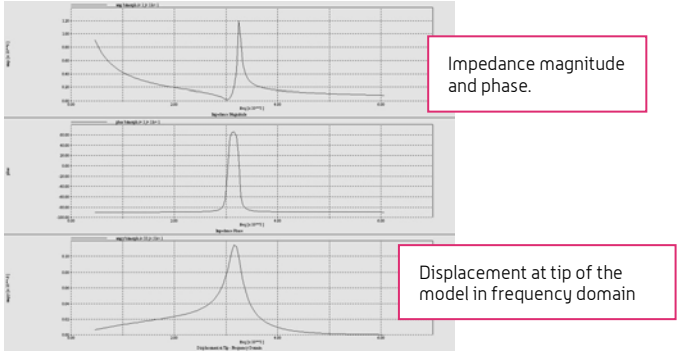
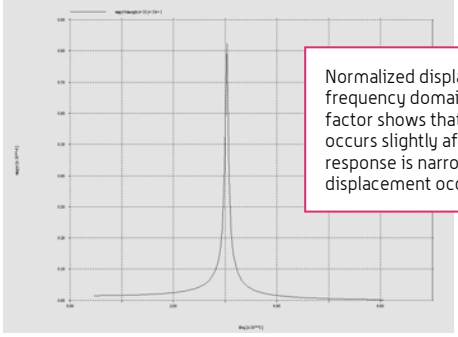


Figure 83—Impedance Magnitude and Phase and Displacement Spectra

Enter the “term” command to go to Fig. 84, which shows the normalization of the displacement FFT. To compare these results with the FFT results for similar models requires the same scaling factor, which is dependent on the total time (tMAX), the total number of points (NT), and therefore the time step between the points (Δt). The FFT of functions with different time bases always show the same shape graph but may have different linear scaling factors. One way to compare two FFTs is to make sure that the time step, total number of points, and maximum time are identical for both simulations. Otherwise the results will be valid only for that specific tMAX, NT, and Δt.

To avoid the effect of the scaling factor, normalize the output function to the input function. In any simulation, the scaling factor is the same for the output and input functions. Simply calculate the FFT of the output function and divide it by the FFT of the input function. This removes the scaling factor, resulting in a magnitude that is the same as the normalization of a different simulation. The units of this magnitude are the units of the output divided by the units of the input. In this case, the normalization magnitude is meters divided by volts (m/V).

The normalized graph shows the frequency ranges in which maximum displacement occurs. It does not, however, show the extent of displacement at those frequencies. In the normalized graph, Fig. 84, a large peak is centered slightly after 3 kHz. Thus for the maximum displacement from the model, you should use a frequency just over 3 kHz. Although the maximum displacement is not evident from this normalized graph, you know that it occurs at this specific frequency. More than one peak in the normalized displacement response would indicate multiple resonances in the structure. Each peak would represent a frequency that created a large displacement. To determine which frequency created the maximum displacement for a particular drive function, the simulation would have to be rerun at each frequency.



Normalized displacement magnitude frequency domain. Removing the scaling factor shows that all of the displacement occurs slightly after 3 MHz. The band response is narrow because all of the displacement occurs at a single frequency.

Figure 84—Normalization of the Displacement Spectra

Discussion of Biomorph Example

The bimorph example shows how adding a resistor to the voltage source can shorten the ringdown time of the voltage time history. The Review file results show the displacement at the free end of the model. The example also shows how allowing the charge to ringdown and become steady at zero for a period of time avoids the need for zero padding. The electrical impedance magnitude and phase graphs are smooth without zero padding. Finally, the example shows how normalization make it possible to compare different results of the same model. One way is to make sure that the tMAX, NT, and Δt are the same for both the input and output functions of the results you wish to compare. The other is to normalize the output function to the input function, thus removing the effect of the scaling factor created by different tMAX, NT, and Δt. Normalizing is a way to create universal data that can be compared with other results without modification. The bimorph example also includes an AVI of the structure vibrating under the drive function.

Stack Example

The stack example contains two models. The first is a 2D model made by stacking three identical pieces of piezoelectric material. The second is the same 2D structure, made of the same materials, extended to a 3D model by adding only a few commands in the input file. Each model has its own input and Review files. Each model also has “snapshots” showing the electric field, displacement, and voltage drive function at different time steps. There are also displacement shapes for both models. A Review file compares the results of the 2D and 3D models. The input files contain a few “grph” commands that are not followed by “term” commands; as with the biomorph example, to pause at each graph requires modification of the input file.

The 2D stack model is created by stacking three identical pieces of piezoelectric material that differ only in terms of poling. The outside pieces are poled positively in the y direction and the middle piece is poled negatively in the y direction. The piezoelectric material is poled parallel to the electric field. The direction of the electric field, together with the direction of the poling, determines whether the material expands or contracts. If the poling and electric field are in the same direction, the material expands; if they are in the opposite direction, it contracts.

In the model, the electric field goes from the live electrode to the ground electrode. In the outer pieces of the stack, the electric field is in a downward direction. In the middle piece, it is in an upward direction. If the poling were in the same direction for all the pieces and the electric field remained the same, the outer pieces would contract and the middle piece would expand. Here, the poling directions are alternated so that all three pieces displace in the same direction. Because the drive function is the same for each piece whether it forces it to contract or to expand, the magnitude of the total displacement for any two pieces is the same. If the poling were all in an upward direction, it would create a canceling effect and the expansion of one piece would cancel out the contraction of the other, as they are displaced by the same amount in opposite directions.

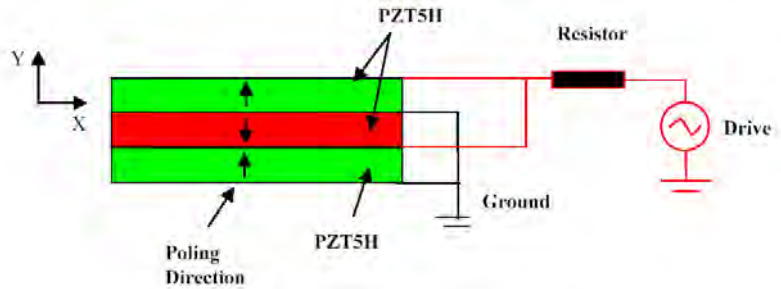


Figure 85

Once the `stack2d.flxinp` input file begins running in the command prompt, you will see Fig. 86, which consists of 4 images. The upper left image is of the model. The image at the lower left shows the electric field through the model. The graph at the upper right shows the voltage drive function applied to the model. The graph at the lower right shows the displacement at the upper center node of the model. The maximum time for the voltage and displacement graphs in Fig. 86 is 30µs.

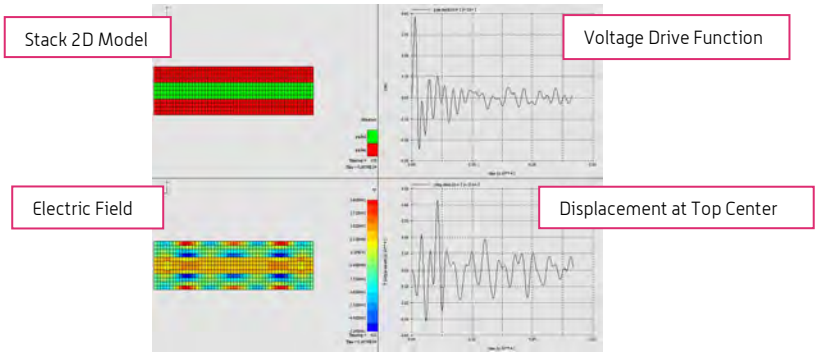


Figure 86—Stack 2D Model, Electric Field, Voltage, and Displacement Time Histories

As with the bimorph example, several graphs are identical to Fig. 86 except for the time elapsed. Figs. 87, 88, and 89 show the model, electric field, voltage drive function, and displacement for longer time intervals than in Fig. 86. Plots occur at 30µs intervals.

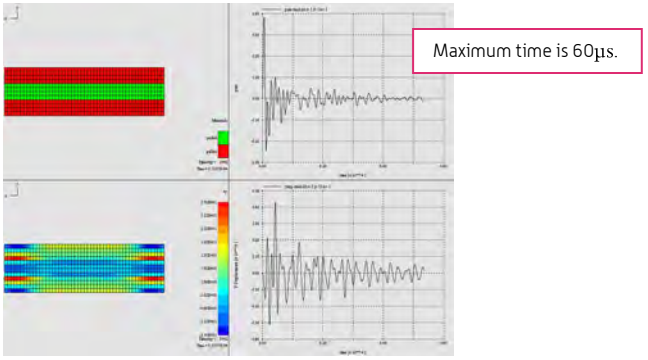


Figure 87—Stack 2D Model, Electric Field, Voltage, and Displacement Time Histories

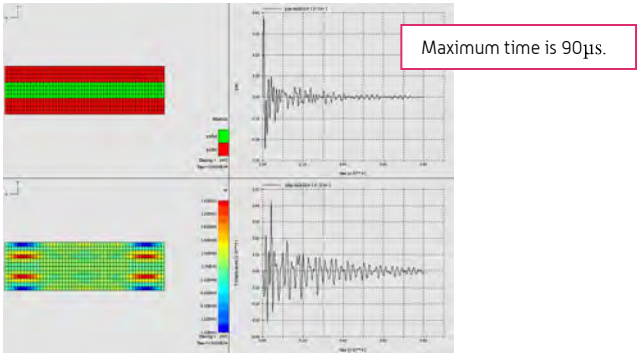


Figure 88—Stack 2D Model, Electric Field, Voltage, and Displacement Time Histories

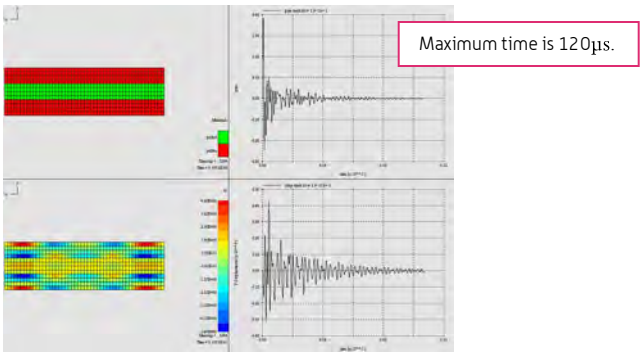


Figure 89—Stack 2D Model, Electric Field, Voltage, and Displacement Time Histories

Following the four graphs showing the electric field, voltage drive function, and displacement, are three images of displacement models. The displacement shapes (often called “mode shapes”) show how the model would react if the drive function were a continuous wave (CW) sinusoidal function at a specific frequency. The displacement shapes are shown after the model has reached a steady state, which is when the model responds in a constant manner to a standing wave vibrating sinusoidally. Due to the broadband nature of transient simulations, PZFlex captures the displacement shapes in a single calculation.

Fig. 90 shows the displacement model at 145 kHz. Each node in the model moves with a sinusoidal function. The differences between the functions are the amplitude and phase. At the center of the model, along the x axis parallel to the boundaries between the three pieces, the amplitude is zero. The largest amplitude, or maximum displacement, is at the

edge of the model. The upper graph in Fig. 90 shows the model at a phase of 0°. The node at the top and bottom in the center part of the model are at a minimum, and the nodes at the top and bottom of the outside of the model are at a maximum. This means that these nodes are out of phase by 180°. In the bottom of Fig. 90 the model is at a phase of 180° and the nodes that were at a maximum are now at a minimum and visa versa.

The same is true for Figs. 91 and 92 but these displacement models are at higher frequencies, meaning a shorter wavelength and more perturbations. Fig. 91 shows the displacement model at a drive function of 358 kHz, and Fig. 92 at a drive function of 690 kHz. The choice of shape frequencies was determined by the results in Fig. 94 (see discussion below).

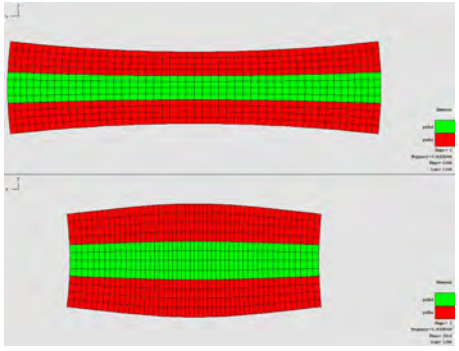


Figure 90—Displacement Shape at 145 kHz

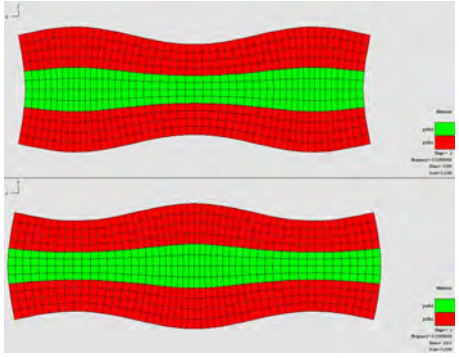


Figure 91—Displacement Shape at 358 kHz

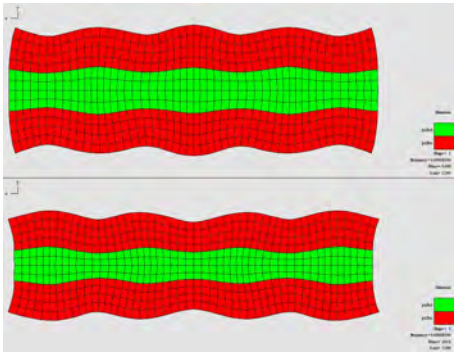


Figure 92—Displacement Shape at 690 kHz

The final graphs of the 2D stack input file show displacement shapes.

Run the Review file for the 2D stack model. The first image to appear is Fig. 93, which is similar to the previous figures. The top graph is the voltage time history, the middle graph is the charge time history, and the bottom graph is the displacement time history. Again the simulation time allows for the voltage and charge to ringdown to zero, creating a smooth electrical impedance magnitude and phase.

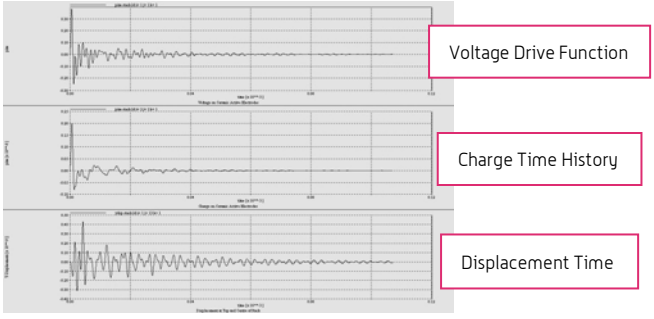


Figure 93—Voltage, Charge, and Displacement Time Histories

Fig. 94 shows the electrical impedance magnitude and phase in the top and middle graph, and the displacement in the frequency domain in the bottom graph. This is where the frequencies for the displacement models, Figs. 90—92 were chosen. In the electrical impedance magnitude graph (upper graph in Fig. 94) the first minima maxima pair occurs before 200 kHz and the second just before 400 kHz. The displacement peaks associated with these resonance pairs are lined up with the minima points. The frequencies of the first two minima are 145 kHz and 358 kHz, the frequencies of the displacement models. There is also a resonance pair with a minima at 690 kHz further up the frequency spectrum in the graph. Unfortunately, the resonant behavior of a structure is rarely available before simulation. To create the displacement models, it is necessary to run the simulation twice—the first time to determine which frequencies are important and the second time to create the displacement models at those frequencies. It is possible to guess the frequency at which the displacement will occur, but to be exact you must run the simulation twice.

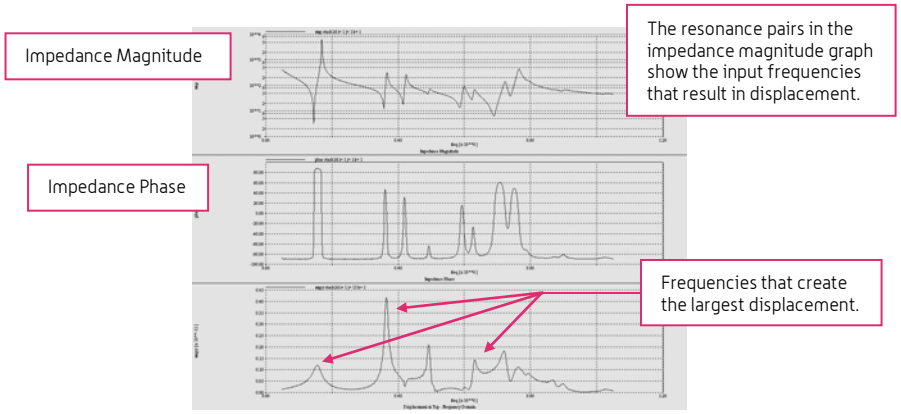


Figure 94—Impedance Magnitude and Phase and Displacement Spectra

As in the bimorph example, Fig. 95 shows the normalization of the output function (bottom graph of Fig. 93) to the input function (top graph of Fig. 93). This is created by dividing the FFT of the displacement time history by the FFT of the voltage time history. The purpose of this normalized graph is to allow comparisons of simulations that are the same except for the scaling factor, or tMAX, NT, and Δt. The units of this graph are meters divided by volts (m/V). Fig. 95 shows clearly the frequencies at which the maximum displacements occur. There are three peaks: at 145 kHz, 358 kHz, and 690 kHz.

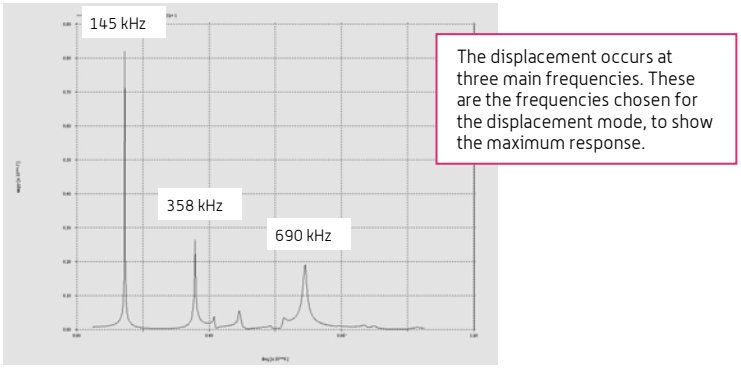


Figure 95—Output Function Normalized to Input Function

This is the end of the input and Review files for the 2D stack model. You will now run the 3D stack input and Review files before running a Review file to compare the 2D and 3D results.

The input file for the 3D stack model is similar to that for the 2D stack model. The 3D stack model is created by copying the 2D stack model and adding z coordinates and k indices. The width of the two models is identical, and the depth of the 3D model is equal to the width. The 2D stack model is a side view of the 3D stack model. The first three graphs, Figs. 96, 97, and 98, give the same plots as for the 2D stack model. The upper left shows the model, the lower left shows the electric field through the model, the upper right shows the voltage drive function, and the lower right shows the displacement. The only differences are that Figs. 97 and 98 have longer time domains.

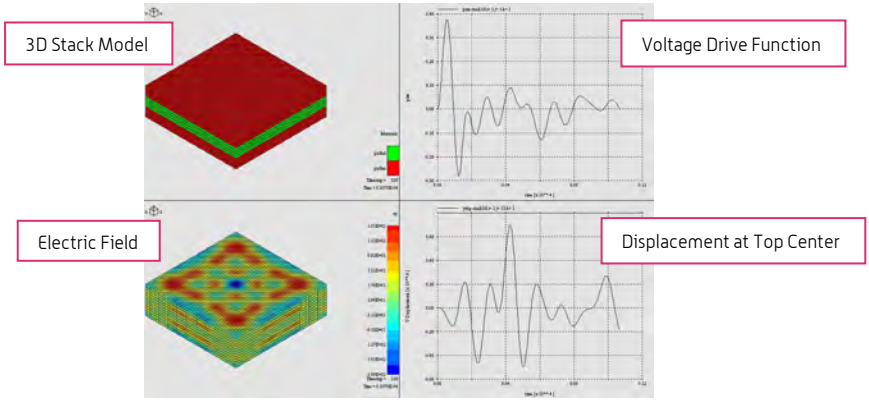


Figure 96—3D Stack Model, Electric Field, Voltage, and Displacement Time Histories

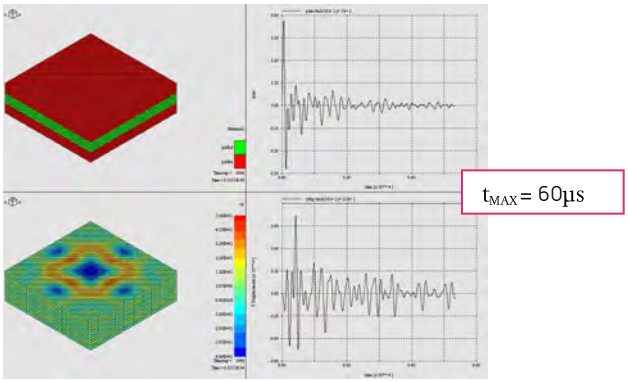


Figure 97—3D Stack Model, Electric Field, Voltage, and Displacement Time Histories

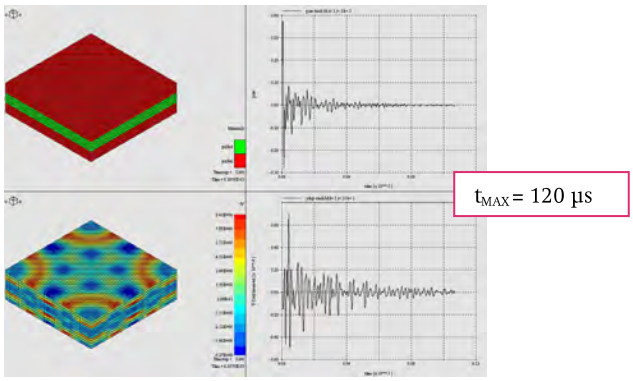


Figure 98—Stack 3D Model, Electric Field, Voltage, and Displacement Time Histories

The 3D model reacts similarly to the 2D model when driven by an impulse drive function at a specific frequency. In Fig. 99 the displacement model is shown at 158 kHz. At this frequency the model behaves similarly to the 2D model at 145 kHz (Fig. 90). Again the upper image shows the model at a phase of 0° and the lower one shows the model at a phase of 180°. The two images are essentially opposites of each other. Where one node was at a minimum, it is now a maximum, and where there was a maximum there is now a minimum.

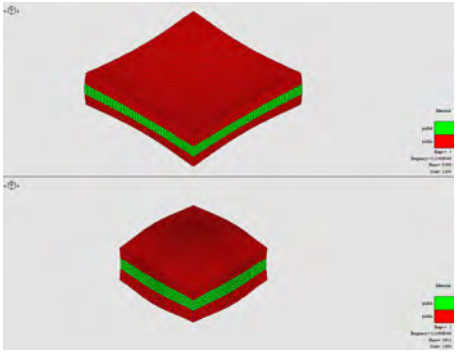


Figure 99—Displacement Shape at 158 kHz

The displacement model in Fig. 100 is run at 356 kHz, and the displacement model in Fig. 101 is driven by a sinusoidal function at 696 kHz. Again, the frequencies were not known before the simulation ran. After the Review file has been run, the electrical impedance magnitude and phase graphs indicate which frequencies create responses.

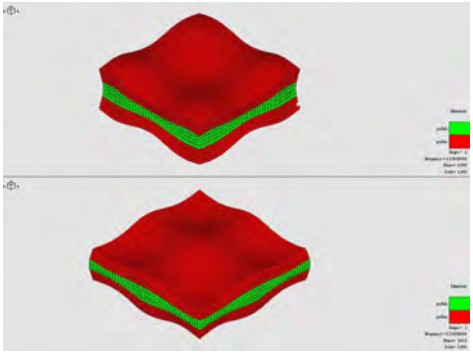


Figure 100—Displacement Shape at 356 kHz

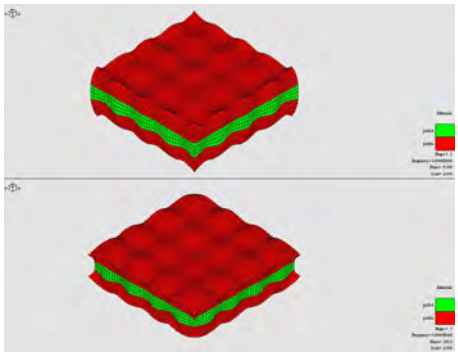


Figure 101—Displacement Shape at 696 kHz

This is the end of the input file for the 3D stack model. Now run the Review file for the model. The first graphs in the Review File, Fig 102 and 103, are similar to previous ones. Fig. 102 contains three graphs. The top graph shows the voltage time history, the middle one shows the charge time history, and the bottom one shows the displacement time history.

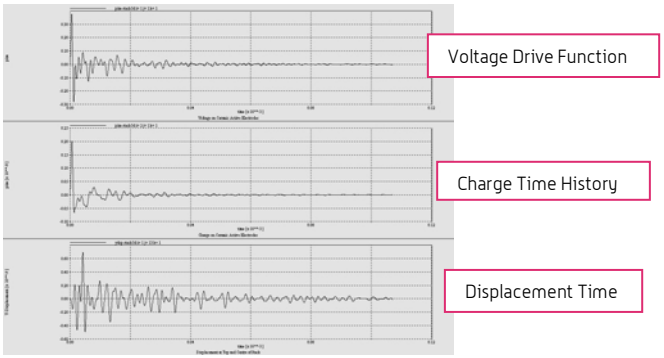


Figure 102—Voltage, Charge, and Displacement Time Histories

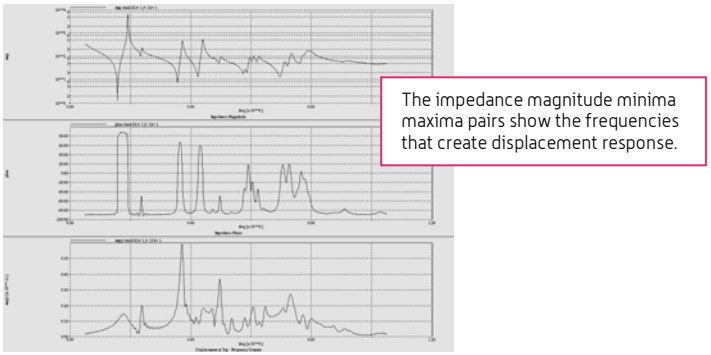


Figure 103—Impedance Magnitude and Phase and Displacement Spectra

Fig. 103 also contains three graphs. The top graph shows the impedance magnitude, the middle one shows the impedance phase, and the bottom one shows the displacement in the frequency domain. The resonance pairs in the impedance magnitude graph are clearly associated with the displacement peaks in the bottom graph.

Fig. 104 shows the normalization of the output function to the input function. As with the 2D model, the normalized graph shows precisely at which frequencies the displacements occur. The frequencies associated with the three peaks are those at which the driving function causes resonant displacement of the model. The displacement models use these three frequencies for the driving function to obtain maximum displacement.

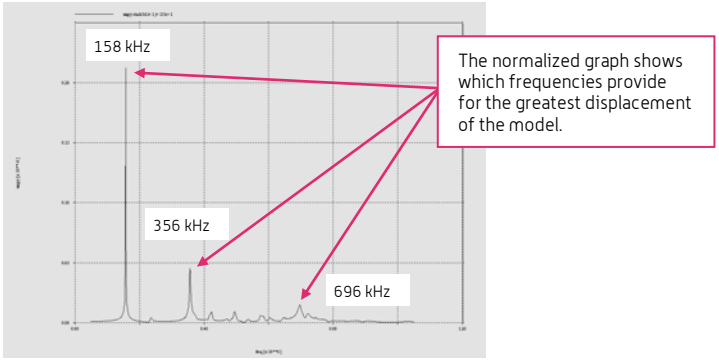


Figure 104—Output Function Normalized to Input Function

This is the end of the Review file for the 3D stack model. Once you have run the input and Review files for both the 2D and 3D stack models, you are ready to compare the responses. To do this, `run revinp.compare2d3d` in the command prompt. In following three figures, the blue line represents the 2D model and the red line represents the 3D model. The first graph, Fig. 105, compares the time histories. The voltage, charge, and displacement time histories are similar. The second graph to compare the two responses is Fig. 106.

The top and middle graphs show the electrical impedance magnitude and phase, and the bottom one shows the displacement in the frequency domain. The graphs are similar but not identical. The graph in Fig. 107, which compares the normalized graphs of the two stack models, also shows slight differences between the two responses.

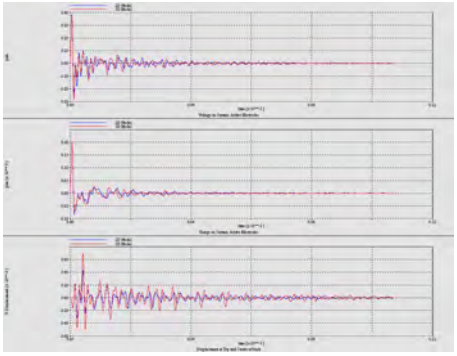


Figure 105—Comparison of Voltage, Charge, and Displacement Time Histories

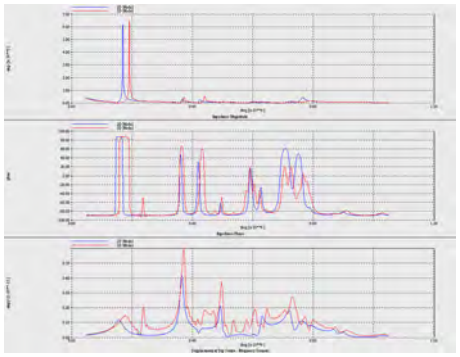


Fig. 106—Impedance Magnitude, Impedance Phase, and Displacement Spectra (both stack models)

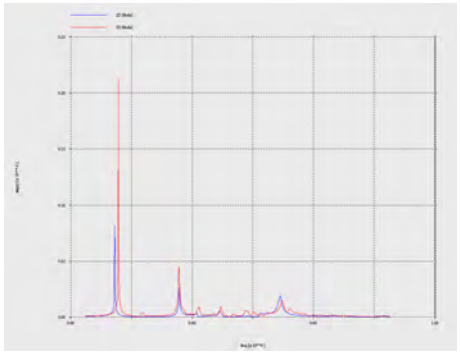


Figure 107—Output Functions Normalized to Input Functions (both stack models)

In this example, the 3D model is created by copying the text of the 2D model input file and adding a third dimension in the z direction; in all other respects, the two models are identical. The length of time required to run the two examples also differs. The runtime of the 2D model on a 1.1 GHz Celeron processor is 18 seconds. The runtime for the 3D model on the same processor is nearly 7 minutes and 16 seconds, about 24 times longer.

With such a simple model, a 7-minute runtime is hardly cause for concern. But with a larger and more complex model, the runtime for the 2D version might be closer to an hour and the runtime for the 3D version more than 24 hours. By creating and running the 2D model first, you can correct any errors without wasting an entire day for a single run. Incorrect meshing or drive frequency, for example, would require you to rerun the entire simulation, wasting several days just to fix relatively minor errors.

Discussion of Stack Example

Creating a 2D model first allows you to make changes and corrections without wasting a great deal of time. The responses, although not exactly the same as for the 3D simulation, are similar enough to confirm that the simulation is running correctly. Once the 2D model is perfected, you can simply add a third dimension. You then need to run the 24-hour simulation only once, rather than four or five times while refining it. This saves valuable time while working on a project.

Batch Example

The batch example creates a simple 2D piezoelectric bar with electrodes at the top and bottom. It demonstrates use of a control file, `review ctrlbatch`, to run both the input and Review files. The batch example also demonstrates use of the parameter sweep, which runs the simulation multiple times (as specified by the user) while changing various parameters for purposes of comparison. Here, the parameter sweep is set to run ten simulations of varying thicknesses of the piezoelectric bar.

In this example, the input file creates the model and saves the time history. The control file then reruns the input file with one changed parameter, the model thickness. It saves the time histories under a new name so as to not overwrite the time history from the previous thickness. This is done ten times for ten different thicknesses. Review takes the time histories and calculates the electrical impedance magnitude and phase for all ten runs. The upper graph in Fig. 108 shows the electrical impedance magnitudes and the lower graph shows the phase. Only four of the ten runs are shown, to prevent the graphs from becoming cluttered and hard to read. For clarity, the Review file assigns each line a different color and pattern. Up to six different curves can be plotted in a single window.

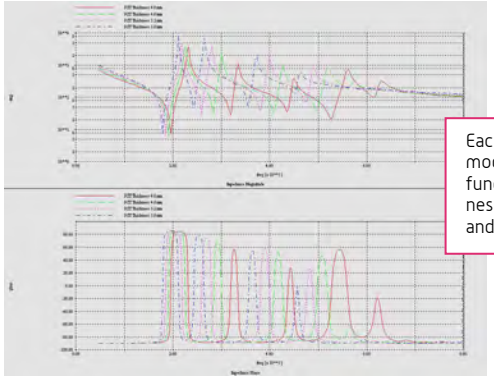


Figure 108—Batch Review Electrical Impedance Magnitude and Phase

The control file uses several commands to run this example, as well as the parameter sweep. The first command is “symb #submit”; it is followed by what is normally entered manually into the command prompt. This is the command that the control file uses to run each part of the example. The command “if noexist” follows the “symb pztthck” command, which defines the initial thickness of the PZT ceramic. The “if noexist” command is necessary because the parameter sweep sets the ceramic thickness before each run of the input file. Without “if noexist” the ceramic thickness would be overwritten by the input file; “if noexist” prevents this, as it is already defined in the control file. Another command used in

this example is “-s”; it sets variables from the command prompt. Here it sets the ceramic block thickness after the “symb #submit pzflex1j batch” command. It also can be entered manually in command prompt.

Discussion of Batch Example

The control file can change as many parameters as you wish. This shortens the time required to see what would happen to your responses if the input frequency, meshing, or anything else were slightly different. It is also safer, as the input file is never changed. If you manually change a number of parameters to see different results it can be difficult to return to the point where you started. A control file also can run multiple simulations without the need to manually start each one after the previous has finished. This allows you, for example, to run multiple simulations overnight.

Curved Ceramic Example

The curved ceramic example creates a rectangular piezoelectric block with a concave surface to focus pressure waves toward a focal point. The pressure waves travel through a region of water and “snapshots” show how the wave propagates towards the focal point. The example is run similarly to the first three.

First make sure that the command prompt leads to the folder containing the input and Review files, `flxinp.curved and revinp.curved`. The first image Fig. 109, shows the model. The red region is the PZT ceramic material, the green one is the backing that absorbs pressure emitted out of the back of the model, the purple regions are braces holding the ceramic in place, and the blue region is water. There are two differences between this example and the previous ones. First, the model is suspended in water, which has a much higher acoustic impedance than air and allows more energy to be coupled from the ceramic to the water, causing a damping effect. Second, and more important, the ceramic has a curved face. This curvature focuses the pressure wave toward a specific point. There are several ways to focus the pressure waves of the ceramic. The simplest are to use lenses or to curve the ceramic. Phased arrays are more versatile but more expensive.

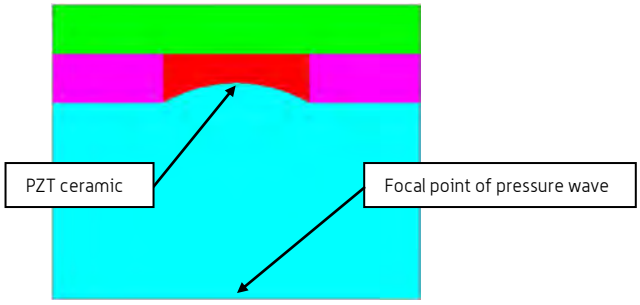


Figure 109—Curved Ceramic Model

Two important aspects of this model are the creation of the curved surface of the ceramic and, more important, the placement of a curved electrode on that surface.

In PZFlex, models are created using a Cartesian coordinate system, or grid. In previous examples, the models were created simply by specifying the coordinates of the model’s location. In this example, it is necessary to create a curved surface, which cannot be done with Cartesian coordinates. PZFlex does, however, allow you to overwrite previous commands with new ones written later in the input file. Therefore, first create the ceramic with a flat bottom surface and include all the extra pieces: sides, backing, and water. Then use “site cyln” (the cylinder command) to position a cylinder of water so that only a piece of it overlaps the rectangular ceramic of the model. Because PZFlex displays the last command written, the cylinder of water essentially erases the ceramic. You now have the curved ceram-

ic, ready for an electrode to be placed on it. Fig. 110 is a close-up view of the curved ceramic piece with the electrode.

The jagged black line, unlike the smooth curved line, represents the actual surface of the ceramic. The model was created in a Cartesian coordinate system, where curved lines do not exist. Thus the circular imprint of the cylinder consists of tiny stair steps of vertical and horizontal lines.

The upper electrode is identical to those in the previous models—a flat electrode along a y -coordinate/j-indices line. Use the "piez" command with the "node" subcommand to create the flat electrode. This approach is not valid for a curved surface. In fact, you will not create a curved electrode at all; you will create a stair-step electrode similar to that in Fig. 110. Because the size of the stair steps is much less than the wavelength of the pressure waves, this appears and functions like a curved electrode.

To create this stair-step electrode, use the "nod2" subcommand under the "piez" command. Tell PZFlex to create the electrode at the boundary between two materials, in this case, water and ceramic. It is helpful to create a small window around the boundary of the two materials, so that PZFlex does not waste time searching the entire model.

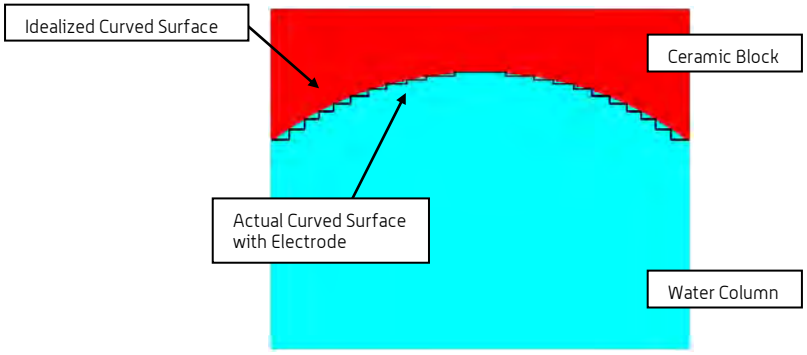


Figure 110—Curved Surface in Cartesian Grid

The next ten graphs in the input file are snapshots of the pressure wave traveling through the water. Four of these appear on the following pages.

Figs. 111—114 show various frames of the wave propagation. The main pressure wave focuses to a point near the bottom center of the graph. The pressure wave is emitted as a sinusoidal wave with maximum varying from a positive maximum, shown in red, to a negative maximum, shown in blue. In Fig. 111 the pressure wave is emerging from the ceramic. In Fig. 112 the wave has traveled farther through the water. In Fig. 113 the pressure wave consists of two large minimas with one large maximum between them. This is clear in Fig. 116, which shows the pressure wave. Fig. 113 also shows that the curved ceramic successfully forces the pressure wave toward the focal point. Fig. 114 shows the wave just before leaving the water region.

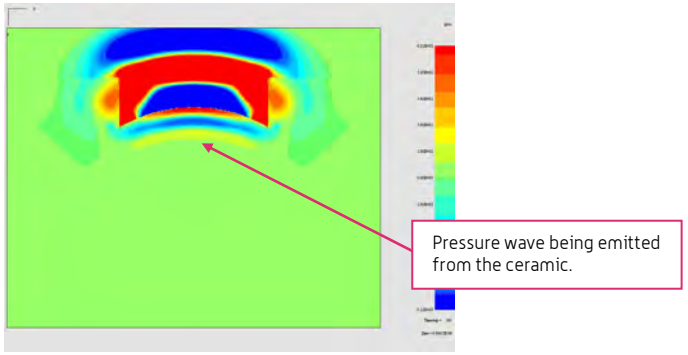


Figure 111—Pressure Waves Emitted by Curved Ceramic (1 of 10)

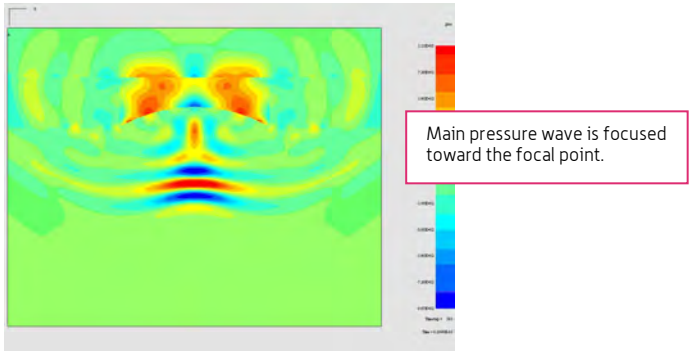


Figure 112—Pressure Waves Emitted by Curved Ceramic (3 of 10)

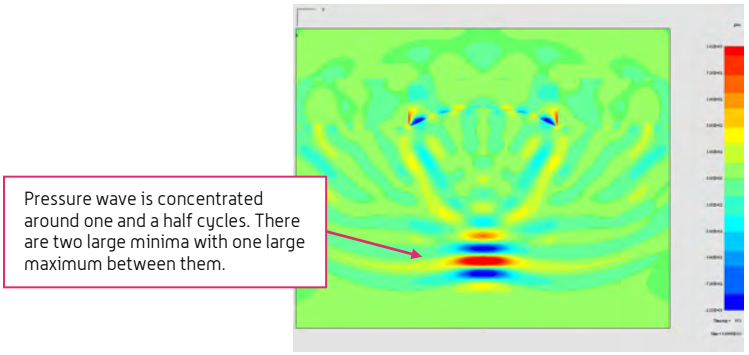


Figure 113—Pressure Waves Emitted by Curved Ceramic (5 of 10)

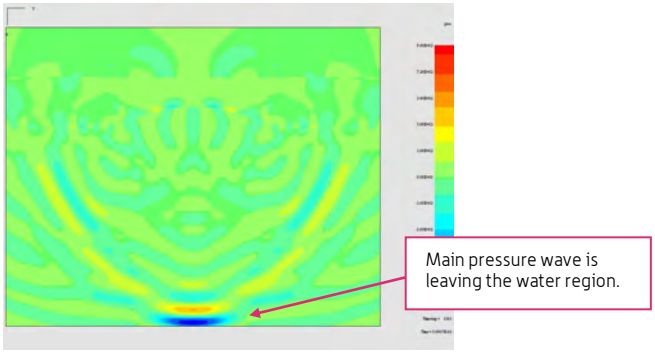


Figure 114—Pressure Waves Emitted by Curved Ceramic (7 of 10)

The input file is now finished. Next run the Review file, `revinp.curved`. Fig. 115 shows the electrical impedance magnitude and phase. The impedance magnitude graph has two resonance pairs. The first, which is easy to see, occurs around 250 kHz. The second increases only slightly from the resonance minimum to the resonance maximum and occurs around 2 MHz. On the impedance phase graph it is easy to see both resonance pairs. The first main peak occurs at the first resonance and the second, broader, main peak, at the second resonance. Because the variable thickness of the ceramic makes the second resonance less clear, the thickness mode is not defined.

Figure 115—Electrical Impedance Magnitude and Phase

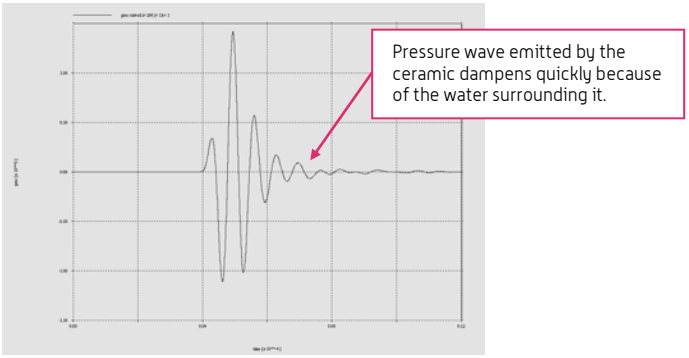


Figure 116—Pressure Wave Emitted by Curved Ceramic

Fig. 116 in the Review file is a graph of the pressure wave at the focal point, which dampens after only a few cycles because the acoustic impedance of the water matches that of the ceramic, allowing more energy to be transferred to the water. The two large minimums and one large maximum are represented by the thick red and blue parts of the pressure wave.

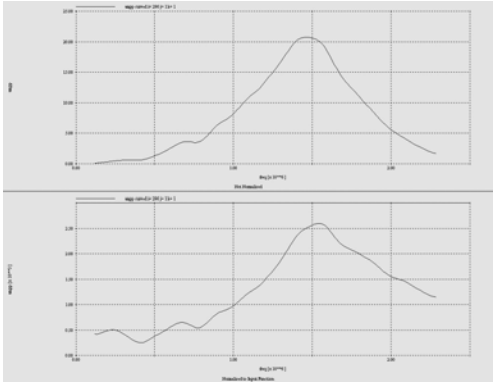


Figure 117—Frequency Distribution and Normalized Frequency Distribution

Finally, Fig. 117 compares the FFT of the displacement time history with the normalized displacement time history to the input function. The normalized graph shows which frequencies have maximum displacement when an ideal input function is applied. The ideal input function applies the exact same voltage at every frequency. The upper graph shows the actual FFT of the displacement for the input function applied to the ceramic. The input function is close to the ideal input function.

Discussion of Curved Ceramic Example

The curved ceramic example demonstrates the creation and use of curved surfaces. A Cartesian grid does not allow you to draw curved shapes or to apply curved electrodes. The curved surface of the ceramic block is created here by overlapping a cylinder of water and "cutting out" a portion of the ceramic, leaving a smooth curved surface. The curved electrode is created using a stair-step approach. The electrode is not actually curved, but consists of tiny vertical and horizontal electrodes at 90° angles to each other. This works because the wavelength of the pressure wave emitted is much larger than the length of any individual electrode along the curved surface. The curved PZT ceramic block of the model focuses pressure waves toward a specific point in the water.

Phased Array Example

The phased array example demonstrates the use of PZFlex in modeling complex piezoelectric transducers; it is intended for those who are already familiar with the more basic models.

Fig. 118 shows the physical structure of the model, with 16 individual piezoceramic elements in polymer, a backing, a matching layer, and a water load. Only 8 pillars are actually modeled, and symmetry is used to represent the other 8. Each pillar has its own electrode, allowing PZFlex to drive the elements individually. In this case, PZFlex drives the edge elements first, moving toward the center. This causes the emitted waves to focus at a point in the water. Because the wave is focused straight ahead, both the structure and load are symmetrical, allowing use of the symmetry option. Had the beam been steered to one side, the asymmetry of the load would have necessitated modeling the entire structure.

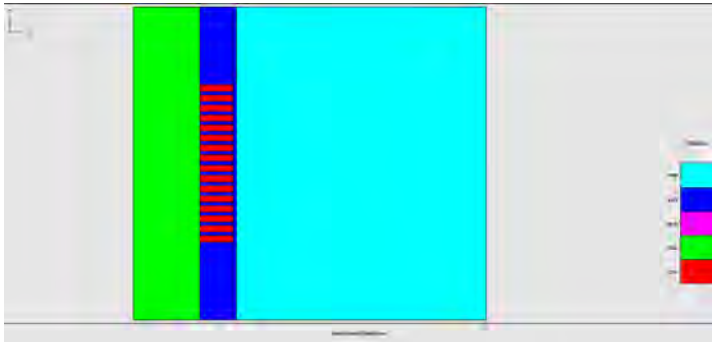


Figure 118—Phased Array Model

The first step in running the model is to define the load function. PZFlex contains a number of functions for representing commonly used drives such as sine waves and step waveform. Because this example involves the time domain, any arbitrary function can be used. It can be read into PZFlex from any external ASCII file, that is, in two-column format (time and amplitude columns). The external file can be created in any manner you choose, e.g., Microsoft Excel. Here you will use Review.

The file `makecurv.revinp` is used to generate a complex load function. For this example, use the "make" function. The first step is to create a time base (number of sample points, start time, and time step) for the generated drive signal. This time base needs to run long enough to contain all the signal of interest and to be longer in time than the full PZFlex simulation it will be used in. The time step chosen should allow a sampling rate high enough to represent adequately the frequencies of interest; we recommend at least 20 points in time over the shortest wave period. The time base need not match that of the simulation; PZFlex interpolates the data as needed.

Using the “make” function, generate the new curve for the desired drive function and display it in both time and frequency domains, as shown in Fig. 119. The “grph writ” function writes any line plot displayed in Review to a two-column ASCII file. You can open the resultant file, `function.dat`, in any text editor to view the format.

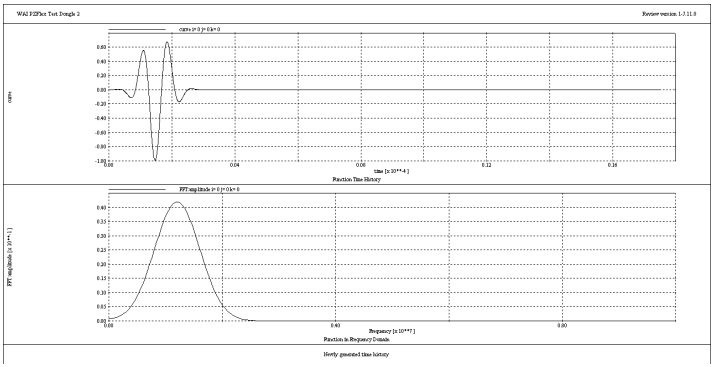


Figure 119—Drive Function

The next step is to build and run the model with `phased.flxinp`. As usual, define all important parameters as “symbol” variables at the top of the input file, then reference these variables in the remainder of the input file. The result is a parameterized input file that allows simple alteration of variables to produce desired changes in the model. Use of the “symb#get findfile” function ensures that the drive function you reference later, `function.dat`, actually exists.

As one of the variables you created, “numpzt,” refers to the number of pillars in the array, you are in the unusual position of not knowing the model width. Therefore, create logic “loops” in each important section using the “do loop” functionality of “symbol,” looping the same number of times as there are pillars. You must specify the -coordinates, indices, “geom” definition, “site” definition, and electrode placement (piez) for model construction.

The new “calc” subcommand “calc max” allows you to store the maximum (and minimum) value of any field as an elemental array throughout the simulation. In this example, it stores the maximum pressure, effectively creating an array that represents the beam profile of the device. You will view this later, in Review.

The new command “data hist” reads in the desired driving function created earlier. Assign it a name, `drv1`, for future reference. There is no limit to the number of drive functions that can be read in this manner; each, however, must have a unique name.

To define the electrical drive, you must create a new electrode for each independent element and for the ground electrode, for a total of 9 electrodes. Each requires a unique name for later reference. When using the “piez bc”command to apply the voltage to the

electrode (as in earlier models), use additional options on the “bc” line to time shift the drive and to scale the magnitude. When the time is shifted, the signal is applied to each element such that the transmit time from all array elements to a given location is the same. This ensures that the device focuses the ultrasound to that spot, firing the edge elements first and then the center elements. Also apodise the array drive, that is, apply a smaller drive signal to the edge elements than to the center ones. The value of this scaling varies as a shifted cosine fnction across the surface; use “symbol” logic to determine those values.

The model is run long enough to ensure that the ultrasound wave of interest has been generated and exited the model (or been absorbed by the boundary conditions). At a number of time points during the simulation, plot the state of the model in three subwindows, stressing different fields. Fig. 120 shows the output at one of those time points. The upper left subwindow shows the pressure wave focusing toward the chosen location. The lower left subwindow shows the pressure in the fluid immediately in front of the center of the transducer. On the right is a zoomed-in view of the displacement of the elements. Use of “rang” and “set wndo” in the “grph” command ensure consistency of the plot ranges of both the color bars and the time-trace window.

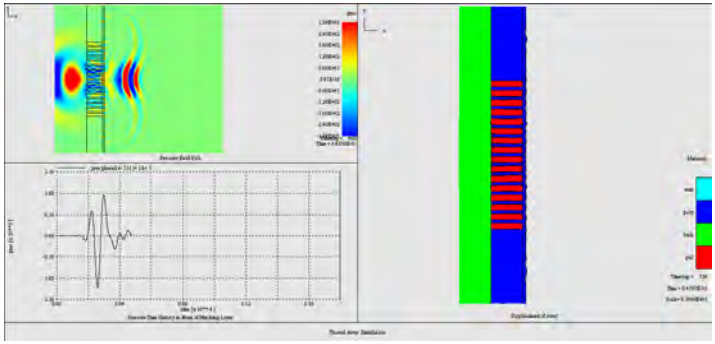


Figure 120—Pressure Wave Output from Phased Array

Upon completion, the beam shape calculated with “calc max” is written out to a “flxdato” file using the “data” command.

You now can view and manipulate both the time-trace results (`flxhst` files) and data results (`flxdato` files) in Review. The file `phased.revinp` displays the data.

The `flxdato` and `flxhst` files are read into Review with the “read” command. Both require folder names (e.g., `f1`). Review automatically detects the type of data and reads it in accordingly.

Once read in, a data array can be viewed the same way as any other array in PZFlex, with two additions: it must be referred to by folder name (e.g., `f1`) and by subfolder name (e.g., `f1/1`). For a single data field read in, the subfolder name is `1`. With multiple data snap-

shots, such as pressure snapshots for 100 timesteps as the wave propagates forward, this number can increase. The pressure beam can be plotted using the command “grph plot f1/1 prmx” to refer to the folder name, subfolder number, and array name.

Fig. 121 shows the first displayed results from `phased.revinp`: the plotted beam profile at the top, the beam along the axis of the device, and the beam across the front face of the device. For the last two, plot appropriate slices in i or j through the field, against the x or y position.

It is easy to see the shape of the beam, the intensities through the axis and at the peak, and the near-field variation of the field across the transducer face.

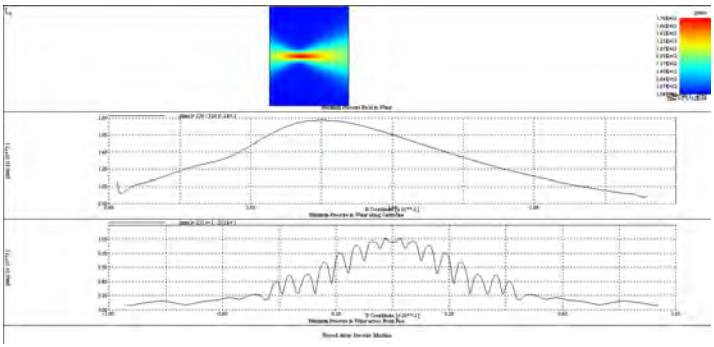


Figure 121—Beam Profile with Cross-Sectional Plot of Pressure on Axis and across Device Face

Now plot two different electrical impedances, one atop the other (one for a center pillar, one for the edge pillar). Despite the identical pillar shape, conditions at the edge of the device cause a slightly different impedance response in the element. The “grph pset lc” and “pset lq” commands preset the color and line quality (solid, dot-dashed, etc.) in the graphs (see Fig. 122).

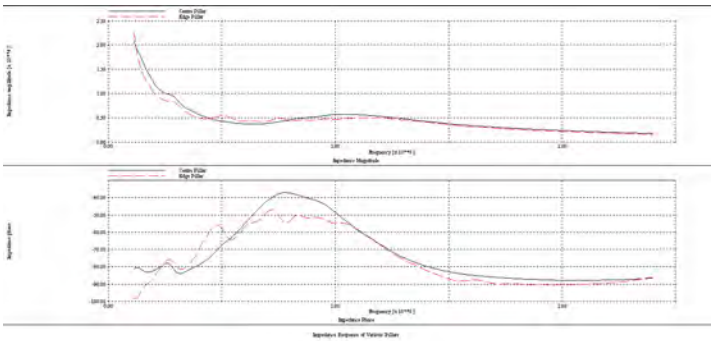


Figure 122—Electrical Impedance Profiles of Center and Edge Elements in Array

Fig. 123 shows the displacement in the direction of the thickness on the same center and edge pillars (time-domain response is on top, frequency-domain magnitude below). The time plot clearly shows the result of both the time shifting (edge pillar driven first) and amplitude scaling (edge pillar driven less, due to apodisation). Given the difference in magnitudes, it can be difficult to compare the displacement frequency responses. Fig. 124 shows the results when you compensate for the scaling and normalize the results. Here, the edge pillar has a slightly higher frequency response, which is difficult to see in the un-normalised plot.

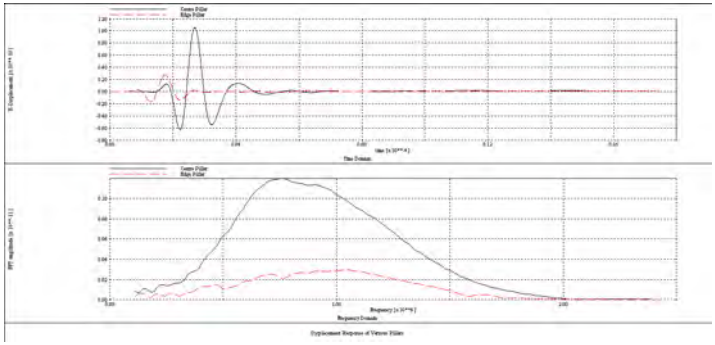


Figure 123—Displacement Response of Center and Edge Pillars (time domain at top, frequency domain at bottom)

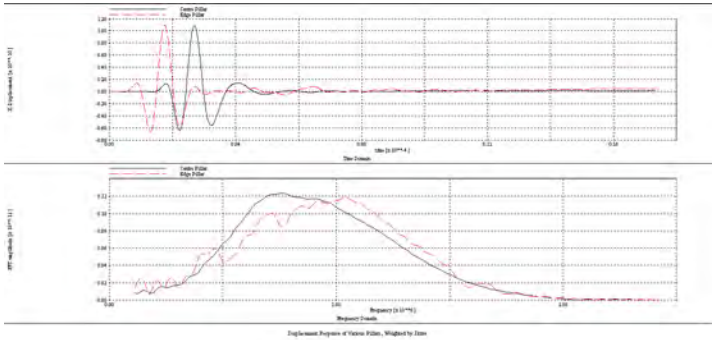


Figure 124—Normalised Displacement Response of Center and Edge Pillars (time domain at top, frequency domain at bottom)

Kirchoff Extrapolation Example

The Kirchoff extrapolation example demonstrates the extrapolation of pressure response at points distant from the pressure source. Kirchoff extrapolation is particularly useful at large distances. To create a finite-element model to display the pressures at a distance of 50 wavelengths or more, a model would have to be 50 wavelengths long. Such a huge model would take a great deal of memory to run and entail discretization problems. Instead, Kirchoff extrapolation creates a much smaller model that extends only 4 or 5 wavelengths past the pressure source. Kirchoff extrapolation assumes a uniform material from the pressure source to the point of interest. Thus the extrapolation is not be valid when there is more than one kind of material or even material in different physical states. When the material is water, for example, the entire body of water between the source and point of interest must be the same temperature.

In this model the source is a simple PZT ceramic disc suspended in water. The model is axis-symmetric through the center, i.e., all cross sections running through the center are the same. The model is a cylinder created by modeling half of a cross section and rotating it 360 degrees around the axis of symmetry. This is the cylinder or disc that you will use to create pressure waves. Modeling a 3D shape by using an axis of symmetry works only if the 2D shape is rotated 360 degrees. It does not work if you want only a quarter or half of a cylinder.

The PZT disc has electrodes on the top and bottom; when a driving function is applied, the ceramic emits pressure waves in all directions. The input file places a small imaginary rectangle around the ceramic disc; pressure magnitude and gradients are saved along this boundary pressure. Fig. 125 shows the ceramic disc and the rectangle used to calculate the initial pressures. Knowing the initial pressure magnitude and gradient allows you to extrapolate the pressure magnitude and gradient at a greater distance. The imaginary rectangle is in the near-field range, and the point at which the data are extrapolated is in the far-field range, 1 meter away, directly above the center of the model. The pressure wave in the near-field range increases in magnitude until it reaches its maximum value. The wave is then in the far-field range and can be described by the simple function 1/R, where R is the distance from the center of the disc to an equidistant arc that runs through the point where you wish to extrapolate the data.

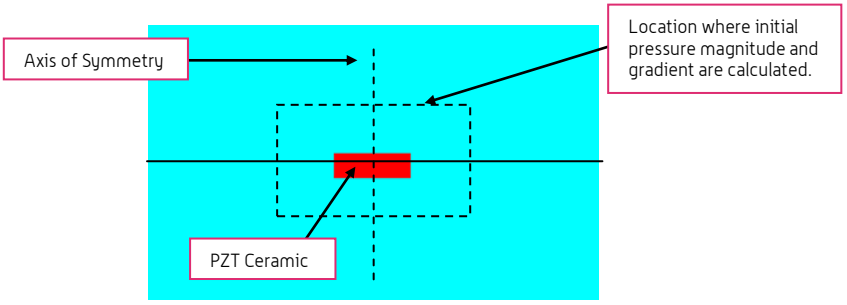


Figure 125—PZT Ceramic and Initial Pressure Rectangle

The input file for this example runs for a short time, creating three Review files. Although they can be combined into a single file, breaking them into three demonstrates the three types of response graphs that can be created using Kirchoff extrapolation.

The first Review file, revinp.kirkextr, creates two displays. Fig. 126, shows the two displacement time responses. The upper graph shows the extrapolated data at the point of interest, and the second shows the actual pressure time response at the point of interest. The two graphs are similar but not identical. The extrapolation does a good job of estimating the pressure. Fig. 127 shows the same two responses in a single graph for a better comparison. The red line is the Kirchoff extrapolated pressure and the blue line the finite-element data.

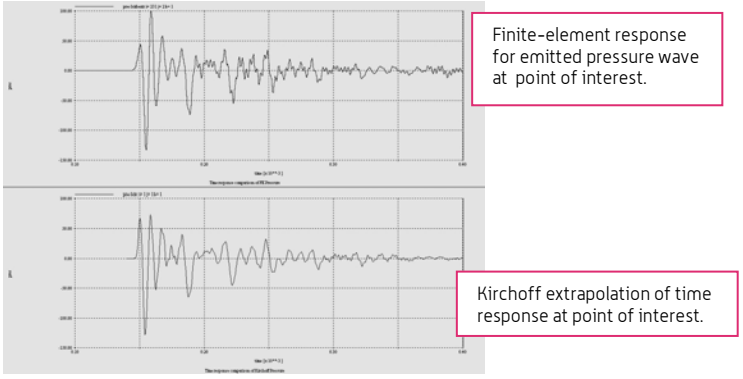


Figure 126—Kirchoff Extrapolation and Finite-Element Pressure Magnitudes

Figure 127—Comparison of Kirchoff Extrapolation and FE Data

The second Review file, `revinp.kirkbeam`, displays the beam profile of the PZT disc, Fig. 128. This graph ranges from -30° to +30°, the angle of the perpendicular line coming out of the middle of the top of the ceramic disc. The graph shows the magnitude of the pressure wave at each angle. Although the majority of the pressure wave is sent out the top of the model, pressure is emitted in all directions.

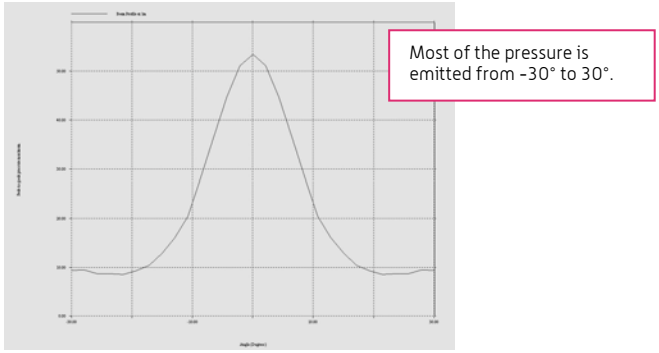


Figure 128—Beam Extrapolation of PZT Ceramic Pressure Wave

The final Review file, `revinp.tvrextr`, creates the TVR or transmitting voltage response. TVR graphs are similar to normalized response graphs but are converted to dBs. They are used mainly in naval applications. In the TVR in Fig. 129, the reference pressure is 1µPa. The graph has several peaks and valleys, which is not ideal for a TVR response.

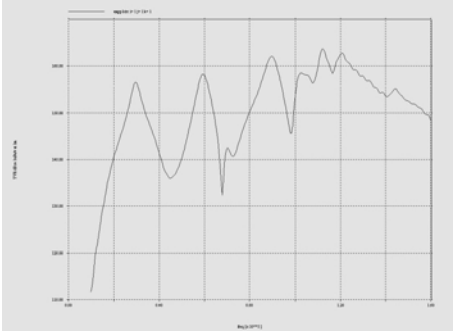


Figure 129—TVR for Kirchoff Extrapolation

Discussion of Kirchoff Extrapolation Example

Kirchoff extrapolation can be very helpful as the distances from the source increase. The first Review file compared the real and extrapolated time histories of the pressure wave at the point of interest, located 1 meter above the model. The second Review file created the beam profile for the ceramic disc, showing the magnitude of the pressure wave emitted in all directions from the disc. The final Review file created the TVR report, used mainly used in naval applications.

Kirchoff extrapolation is useful when finding pressure at points distant from the source. It does, however, have certain limitations. For the results to be valid, there must be uniform material from the source to the point of interest. To ensure that the pressure extrapolation is at the correct angle, it is useful to extrapolate at a short distance from the source and compare the results to the finite-element calculation at the same distance. This must be a distance inside the model. If the two results are the same, the angle is correct. Once the simulation has run and extrapolated the pressure at the point of distance, it is possible to find the pressure at any point along an arc running through the original point of interest and centered at the pressure source. This can be done without rerunning the entire simulation.

Sloped Example

The sloped example creates a trapezoidal PZT block that emits pressure waves into a column of water. The pressure fields in the water column are displayed for specific drive frequencies. The sloped structure is not created in the same way as the previous six examples. Instead, it is created using a skewed grid. A skewed grid is similar to a standard Cartesian grid but does not have perfect 90° intersections of the grid lines. Imagine taking a Cartesian grid and squeezing the grid lines together or stretching them apart to created diagonal grid lines. A skewed grid is created by positioning each node in the correct position to create the desired grid. Skewed grids allow for easier construction of sloped edges of a ceramic. Fig. 130 shows a skewed grid; the sloped edges of the ceramic run along the grid lines in the skewed grid.

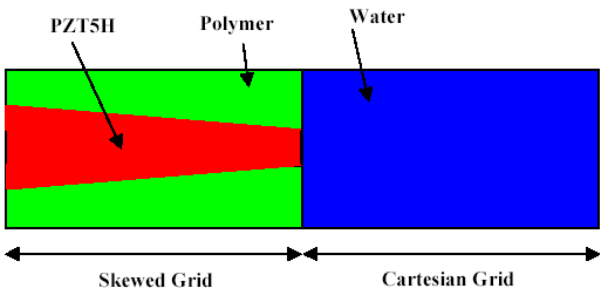


Figure 130—Sloped Ceramic Model

To create the skewed grid for the input file you must first run the Build file. There are no pictures associated with the Build file, i.e., no graphical output. The output is a `bldgrdo` file that is read into PZFlex. Next run the PZFlex input file. There is one picture for this file. The upper image of Fig. 131 shows the slope model in x, y, and z coordinates, as well as the location of each node. The sides of the ceramic taper in toward the bottom of the model. The lower image shows the same model but from PZFlex's perspective. PZFlex views all models in i, j, and k indices rather than x, y, and z coordinates. What matters to PZFlex is not what the model looks like or the exact locations of all the elements, but that the calculations are done for each element as accurately and quickly as possible. Using i, j, and k indices ensures that all the element are the same size and that all the grid lines are equally spaced and intersect orthogonally. Thus the model appears to be square, no matter what it looks like in a graph with x, y, and z coordinates. The i, j, and k indices allow PZFlex to run through the elements by systematically going through each i, j, and k point on the graph, i.e., 1,1,1 then 1,1,2 then 1,1,3 and so on. If all the grid lines are the same distance apart, it is simple to program PZFlex to know where each element is. If PZFlex used the x, y, and z coordinate system, some elements might be far apart and others close together. In that case, it would take much longer to complete the calculations.

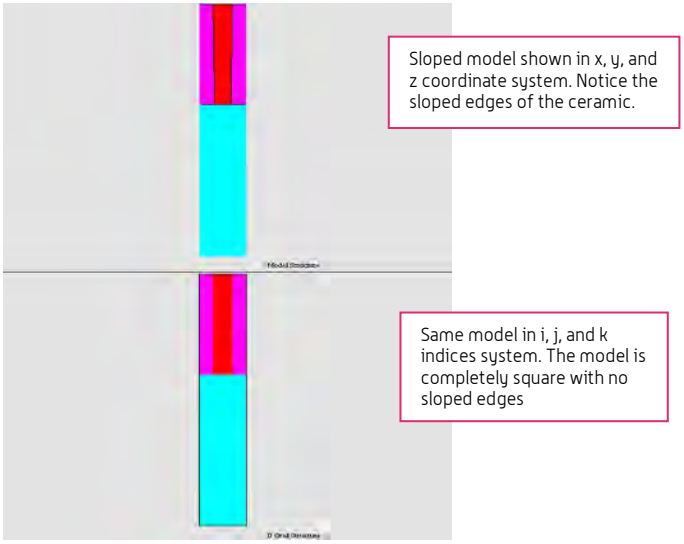


Figure 131—Model Structure and IJ Grid Structure

Once the Review file is running, Fig. 132, which shows the electrical impedance magnitude, appears. There are three resonances in this graph. In the frequency domain, the first resonance, centered around .9 MHz, is the largest. Fig. 133 shows the normalized pressure response at the center of the water column. It is evident that for the maximum pressure at the center of the water column, a drive frequency of .8MHz or 2.8 MHz should be applied.

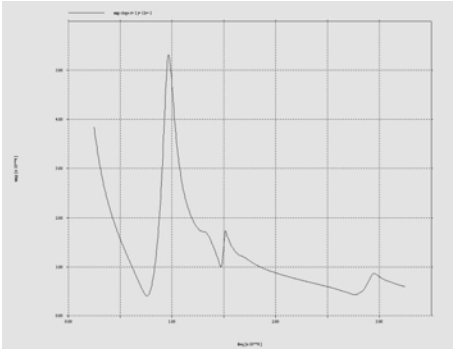


Figure 132—Electrical Impedance Magnitude

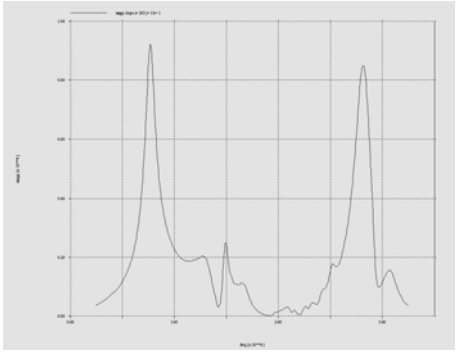


Figure 133—Normalized Pressure Response

Fig. 134. shows the pressure field in the water for specific drive frequencies. These images are similar to displacement shapes except that they graph pressure rather than displacement. In the top graph a drive function of .76 MHz is applied, producing a simple pattern of positive and negative pressure. In the middle graph, a frequency of 1.47 MHz produces more maximums and minimums, but they are not as large as those in the top graph. In the bottom graph, a frequency of 2.77 MHz produces a complex pattern.

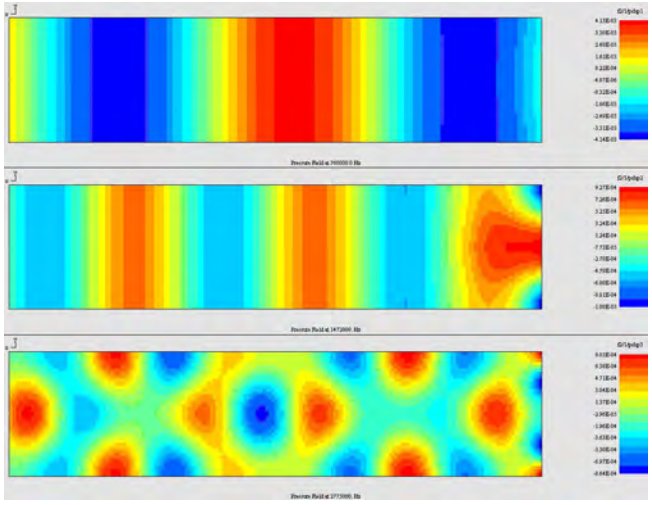


Figure 134—Pressure Field for Given Frequencies

Discussion of Sloped Example

The simple sloped example shows how to use the skewed region to create models with sloped edges or edges that do not run parallel to Cartesian grid lines. It also shows how to create pressure field plots for specific frequencies. As with the displacement models, the frequencies for the pressure fields are not chosen until after the Review file has been run one time. Fig. 132, the normalized pressure response graph, has three main peaks. They are centered at the frequencies that produce the maximum pressure in the water column. These are the frequencies chosen for the pressure field graphs.

Thermal Example

PZFlex is commonly used for the simulation of high-intensity focused ultrasound (HIFU). With HIFU, ultrasound is focused to a point either by mechanical focusing (a bowl-shaped device) or by electrical steering (a phased array). Mechanical attenuation in the propagating medium causes energy from the ultrasound wave to be lost as heat. The large pressure at the focal region, however, results in the largest temperature rise at that location. Selectively directing the focus to regions of unwanted tissue allows it to destroy solid tumors. PZFlex can model both the ultrasound wave propagation and thermal aspects of HIFU.

Finite-element (FE) models require a driving force, and thermal models are no different. In the case of a thermal model, the driving force is the energy deposition across the model, as calculated from the mechanical wave propagation. Although this energy deposition and temperature rise could be calculated simultaneously, the significant difference in time scale between thermal and mechanical effects makes such an approach inefficient. Because the mechanical wave propagation occurs on a scale of microseconds and the thermal effects on a scale of seconds (a factor of 1,000,000 difference), the mechanical and thermal models are decoupled. The usual sequence is to run the mechanical ultrasound propagation model to completion and then use the results in a second, thermal model that is run at a different time step. Thus you need two PZFlex input files for this example: one for the mechanical model and another for the thermal model.

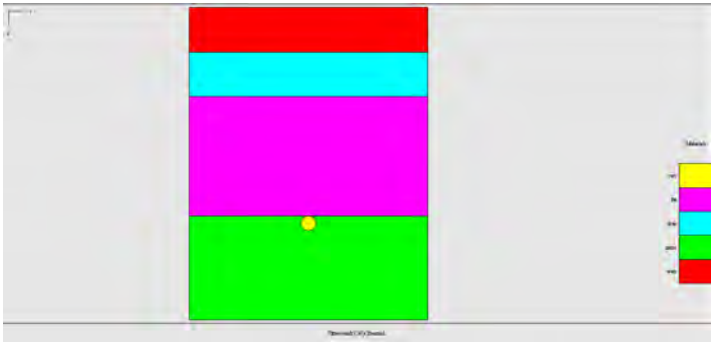


Figure 135—Model Structure

There are two ways to drive the mechanical model, pulsed and continuous wave (CW). With the pulsed model, a short broadband pulse of ultrasound is transmitted and the model is run until all the significant waves have exited the model (or been absorbed). Data are gathered for loss (energy deposition) over the entire simulation. With the CW model, the model is run with a continuous sinusoidal drive of a given frequency until the system reaches steady state, i.e., all the cycles are identical. Energy-loss data for the CW model are gathered over a single cycle only after the system has reached steady state. Examples of both pulsed and CW operations are provided.

Acoustic Model

Begin constructing the model the same way as any other PZFlex model. We encourage use of a dense elemental mesh for constructive interference between all waves and sharp pressure gradients near the focus.

Fig. 135 shows the structure of the model used in this example; it has multiple layers and a small round object to simulate the HIFU target. During the material definition stage, it is imperative that the attenuation parameters of the materials be accurate to ensure accurate calculation of the energy loss in each material. PZFlex has highly versatile damping models that allow not only for separation of the shear and longitudinal damping, but for frequency dependence of the damping. For accuracy, use the Material Damping Tool to investigate the different damping models and display the actual damping applied.

In addition to accurate attenuation parameters, you must also supply each material with thermal properties if they are to be used in the thermal model. Assign these with the "thrm" subcommand, being sure to include values for specific heat capacity and thermal conductivity. Although perfusion effects are an option with this subcommand, they are not used in this example.

In addition to normally calculated parameters, the "loss" array (energy deposition) must be stored, using the "calc loss" command. By calculating the PMAX array, you can store the maximum pressure in it at any time during the model, thus effectively storing the pressure beam profile.

The load in the initial mechanical simulation (i.e., the acoustic wave propagation model) uses a number of pressure surfaces, each driven at a slightly different time to focus the ultrasound to a point. This simulates either the output from a focused bowl or a phased array. Although all the pressure surfaces are driven with the same signal, each is shifted slightly in time so that all waves arrive simultaneously at the focus. The wave begins propagating from the edge of the source, moving toward the center. Use "symbol" logic loops to apply a pressure load ("plod") to each pressure surface. The size of the surfaces should be small relative to the signal wavelength so that the pressure source more closely represents a point source. Using the surface distance from focus, you can calculate the time of flight to the focus and use that as the delay time in the "plod pdef" command.

The "wndo" command sets a computational window to the size specified for the first model time step. PZFlex then calculates wave effects only in that specified region. As the model "steps forward" one time step, the window expands one element in each direction (i, j, and k), constantly traveling ahead of the wave. Although this can result in substantial speedups in the early stages of a model, by the time the window expands to the full model size, the advantage is lost. Be sure that "wndo" covers all acoustic and piezoelectric sources.

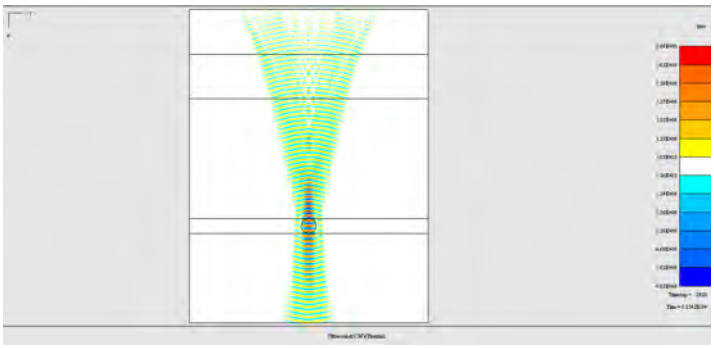


Figure 136—Steady State Field under CW Excitation

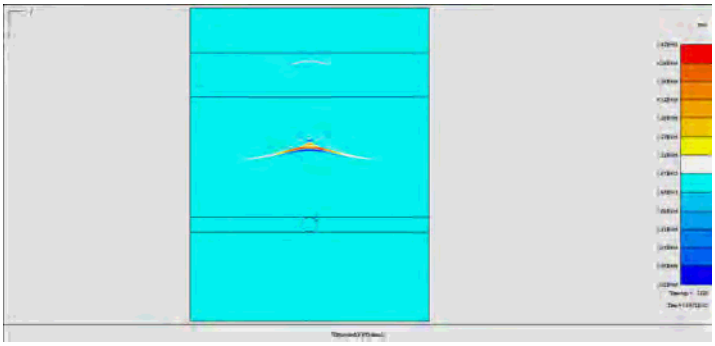


Figure 137—Pulse Wave Propagating toward Target

The simulation is then run for a period of time. Note that for a CW model this is until steady state (approximately 75 cycles, in this case) and for a pulsed model until the wave exits the model. Fig. 136 shows the model when it has reached steady state under CW excitation. Fig. 137 shows the pulse propagating toward the target. Once the pulse has passed, the pulsed model finishes, saving the “loss” array for use as input function for the thermal model. The CW model requires an additional step. Using the “set loss 0” command, reset the “loss” array, deleting all energy loss calculated up to that point. Then run the model for a single cycle (the “exec cycl” command uses the frequency in “func” to determine the cycle period). The energy deposited over a single cycle is calculated and stored in “loss”. The CW model then ends, saving these data.

Fig. 138 shows the “loss” array for the CW model, and Fig. 139 shows the “loss” array for the pulsed model. The distribution is similar but not identical. It depends not only on the pressure wave magnitude at that location, but on the frequency content of the passing wave and the attenuation parameters of the material.

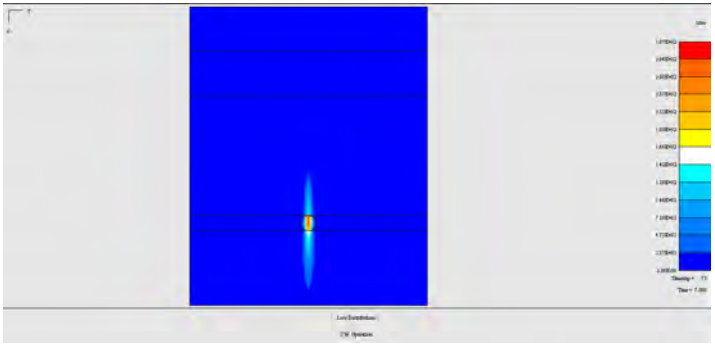


Figure 138—Loss Distribution with CW Excitation

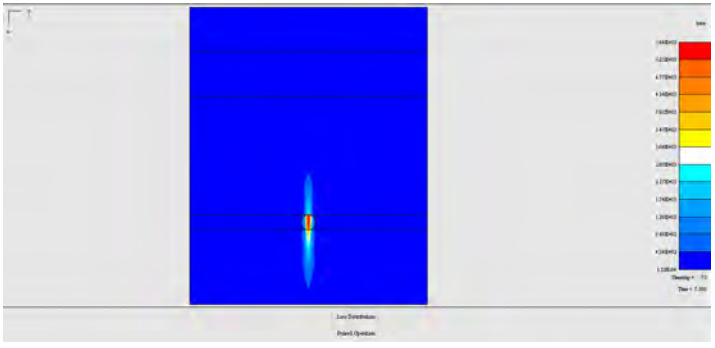


Figure 139—Loss Distribution with Pulsed Excitation

Thermal Model

Use the acoustic model as a base to begin building the thermal model; the files are similar up to the point of the "site" commands. Take the `focuscw.flxinp` (or `focuspulse.flxinp`) files and discard all commands after "site" has been completed. The material properties for the thermal model are the same as those for the acoustic model.

The only alteration is in the mesh density. As thermal gradients are typically much smaller than acoustic gradients, the mesh can be less dense for the thermal model. You need only enough elements to capture the temperature variation across the model. This may result in uneven mesh densities, such as around the focus; for this model, however, use an approximately even mesh. The low mesh density is useful, as thermal equations are solved implicitly and large thermal models can be significantly slower than explicit solvers of the same mesh density.

The first significant difference between acoustic and thermal models is in the "calc" command. Because this is a thermal model, temperature ("tmpr") rather than stresses, strain, or pressures is calculated. As you want to see the hottest points throughout the simulation, use the "calc max"option to store the highest temperatures.

You must tell PZFlex that this is a thermal rather than acoustic calculation. Using the "heat" command, tell PZFlex to use the conjugate gradient solver ("slvr cgds"), which is ideal for large thermal models. Also tell PZFlex that, due to the difference in time step ("cupl off"), the thermal and mechanical fields are not coupled.

Having calculated a loss map to use as a load, you now need to specify how the loss varies with time. Most HIFU systems remain on for a period of time, then go off for a period of time—the periods defined by the duty cycle. As no such drive function is available under FUNC, you must manually define one with the "data hist" command. Specify points on the curve by two values, time and amplitude. You need define only those points at which change occurs or is about to occur; PZFlex interpolates linearly between the specified points. Assign a name to this drive, in this case `drv1`, for future reference.

Boundary conditions ("boun") are defined differently in thermal models than in acoustical ones. By default, any unspecified boundary in a thermal model is symmetrical; that is, you assume that an identical structure and load exist on the other side of the surface plane. Any other surface, either in contact with a heat source or acting as a radiator to infinity, must be specified.

Use the "boun" commands to import the loss from PZFlex for use as a drive. With the "defn" subcommand, tell PZFlex that the energy distribution is to be considered a power and how it will vary with time. Because you are using an energy load that is considered power, you must provide a scaling factor to convert from energy to power. In the case of the CW simulation, the scaling factor is the same as the driving frequency. For pulsed simulations, it is the pulse repetition frequency. The "mshp" subcommand then reads in the specified array ("loss") from the named data file.

Unlike with most acoustic simulations in PZFlex, you must specify a time step with the "time" command. As this is a thermal simulation, there are no wavespeeds with which to set the appropriate time step. Using even the true material velocities would result in a time step suited to a microseconds-long acoustical simulation rather than a seconds-long thermal one. Fortunately, the implicit thermal solution is unconditionally stable; that is, the model will be stable and run to completion no matter what time step is specified. You must, however, specify a time step that is small enough to capture the temperature variation over time anywhere in the model.

Finally, unlike with acoustic models, with thermal models the base settings must be determined—in this case, the starting temperature. Use the "set tmpr" commands to set the temperature to 37°C, average bodily temperature. This must be done after the "prcs" step.

A thermal model is run exactly the same as an acoustic model. After running time steps with "exec" you can view the results ("grph plot tmpr"), run more time steps, and so on. The temperature fields can be read and displayed the same as any other array in PZFlex. Fig. 140 shows the temperature field at the end of the CW simulation; it is similar, but not identical, to the loss distribution. Factors such as the thermal properties of the materials strongly affect temperature distribution. Again, temperatures in the pulsed run are similar (Fig. 141), but not identical, to those in the CW run. Note the lower peak temperature.

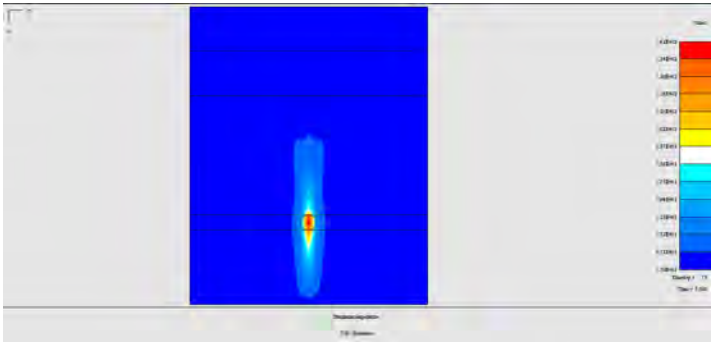


Figure 140—Temperature Field at End of CW Thermal Simulation

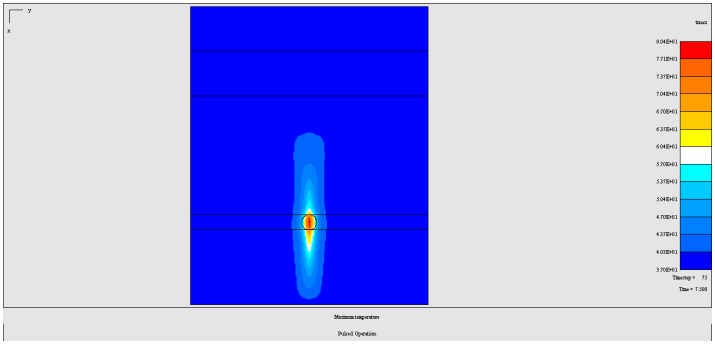


Figure 141—Temperature Field at End of Pulsed Thermal Simulation

Finally, examine both the maximum temperatures in the model and the temperature against time at the focus in Fig. 141. Note the rise during the drive period, with rapid cooling during the off phase. The temperature does not manage, however, to recover to the base 37°C by the time of the second pulse, resulting in a higher temperature peak.

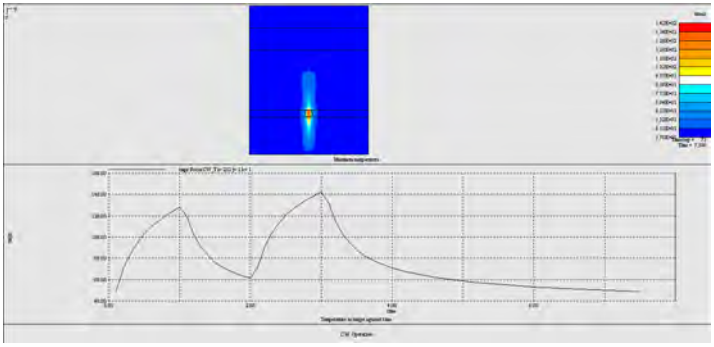
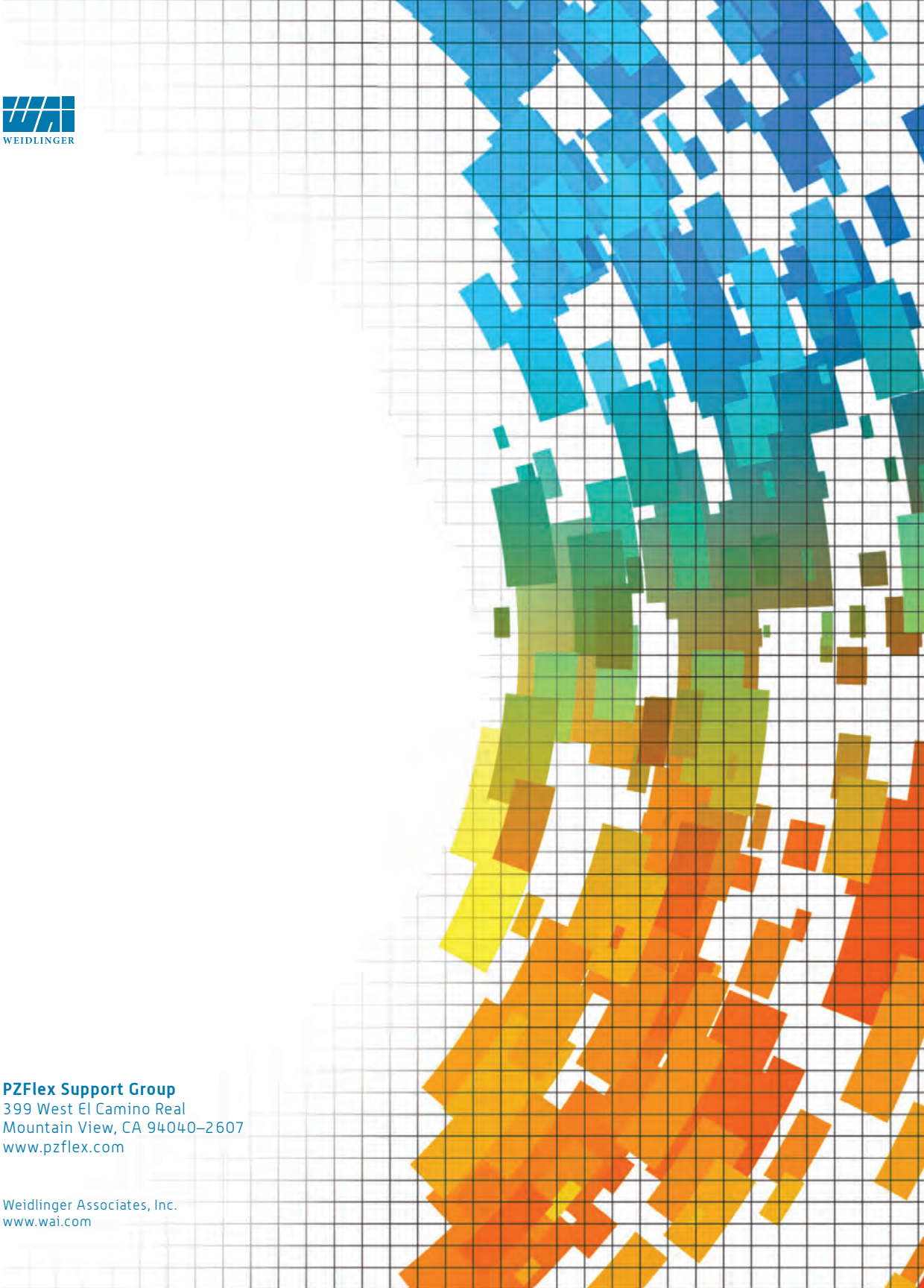


Figure 142—Maximum Temperature at Any Time during Simulation (upper) and Temperature at Focus against Time (lower)

Discussion of Thermal Example

The thermal example shows how to determine heat losses created in a piezoelectric model by splitting the mechanical and thermal simulations into two sections. First you model the acoustic wave interactions in the model and then you use the resulting loss data to model the thermal losses in the device. The decoupling of the two models allows for efficient modeling, as they occur on vastly different time scales. The example also shows how to calculate the “pmax” or pressure maximum developed in the model.

INDEX



PZFlex Support Group
399 West El Camino Real
Mountain View, CA 94040–2607
www.pzflex.com

Weidlinger Associates, Inc.
www.wai.com