PZFlex Matlab Toolbox

This Matlab Toolbox has been developed to allow users to create and run PZFlex models directly from Matlab. This helps users who are experienced in using Matlab to quickly and easily create and run simulations in PZFlex.

The toolbox contains a series of Matlab fuctions for the most commonly used PZFlex commands. When called in a Matlab script these functions print the correct lines of code to a PZFlex input file or review file.

**Toolbox contents**

* The ‘Matlab Functions’ folder contains all functions available for use when creating a PZFlex input or review file.
* The ‘Example Script Files’ folder contains example Matlab scripts which have been created to demonstrate how to use the Matlab Toolbox to generate and run PZFlex models.
* The ‘Complete Function List’ word document has been created to provide a useful resource to view which PZFlex PCOM are available for use. It provides details of which secondary commands (SCOM) are available and their required input parameters.

**The functions**

The Matlab functions contained in the toolbox have been named to match the PZFlex primary commands (PCOM) as closely as possible. The functions used for creating a review file contain ‘Review’ at the end of the function name, e.g ‘extrReview’.

Details of the function and their input parameters are commented at the start of each function and can be seen by typing ‘help (insertfunctionname)’ in the Matlab command window. The ‘Complete Fuction List’ word document is very useful to see what SCOMs are available as well as providing details of the input parameters.

Where the PZFlex PCOM requires secondary commands (SCOM), the first input into the function is always the SCOM as a string. The number and type of remaining inputs to the function then depend on which PCOM and SCOM is being called.

**Before use**

It is important that the Matlab Toolbox is included in the Matlab search path before use. This can be done by copying the folder into the ‘MATLAB’ folder contained in ‘My Documents’ or by manually adding the folder to the Matlab search path.

**Using the toolbox**

**Creating a PZFlex input file**

When creating a Matlab script to run PZFlex it is essential that a PZFlex input or review file is generated and given the correct name, as follows:

FID = fopen('pzflex.flxinp','w+');

fclose(FID);

or

FID = fopen('pzflex.revinp','w+');

fclose(FID);

If the PZFlex input file is not created or named correctly, when the functions are called they will not print to the file. It is possible to then rename the input file to a name of your choosing at the **end** before it is ran in PZFlex. This can be done using the movefile function in Matlab:

movefile('pzflex.flxinp', 'newname.flxinp');

**Model variables**

The model variables can be created in Matlab and can then be used as inputs to the functions.

**Writing to PZFlex input file**

With the PZFlex input file created, the required PZFlex PCOMs can be written to the input file by using the appropriate function.

The function can be called multiple times to add numerous SCOMs to the input file as long as another function is not called in between. With the exception of ‘grph’, the PCOMs will only be written to the PZFlex file **once**.

**Running the PZFlex input file**

At the end of the Matlab script, the PZFlex input file can be ran using the ‘pzflex’ function.

**Running multiple simulations**

It is possible to use the Matlab Toolbox to run multiple PZFlex models like it is done using batch in PZFlex. It can be done by creating a loop in the Matlab script to write and run numerous input files. When used it is important that the input file is renamed, as described previously, before the model is run to prevent data being overwritten.

**PZFlex Help**

For general help with PZFlex simply entering ‘pzflexHelp’ in the Matlab command will open the PZFlex help documentation.

|  |  |  |
| --- | --- | --- |
| **COMPARRISON OF PZFLEX AND MATLAB COMMAND INPUTS** | | |
|  | **PZFlex** | **Matlab** |
| Read general material file | symb #read general.mat | readOption(‘general.mat’); |
| Set up grid and nodal coordinates | symb x1 = 0  symb x2 = 10  symb y1 = 0  symb y2 = 20  symb i1 = 1  symb i2 = $i1 + nint ( ( $x2 - $x1 ) / $box )  symb indgrd = $i2  symb j1 = 1  symb j2 = $j1 + nint ( ( $y2 - $y1 ) / $box )  symb jndgrd = $j2  grid  $indgrd $jndgrd  end  geom  xcrd $x1 $x2 $i1 $i2  ycrd $y1 $y2 $j1 $j2  end | x = [0 10];  y = [0 20];  i = [1 1+round((x(2)-x(1))/box)];  j = [1 1+round((y(2)-y(1))/box)];  grd(i(end), j(end));  geom(‘xcrd’, x(1), x(2), i(1), i(2));  geom(‘ycrd’, y(1), y(2), j(1), j(2)); |
| Define nodal coordinates using ‘keypnt’ | geom  keypnt 2 2  end | geom(‘keypnt’, 100, 100); |
| Set material properties for region | site  regn void  end | site(‘regn’, ‘void’); |
| Define x, y and z boundary conditions | boun  side xmin free  side xmax free  side ymin absr  side ymax absr  side zmin symm  side zmax symm  end | boun(‘side’, ‘free’, ‘free’, ‘absr’, ‘absr’, ‘symm’, ‘symm’); |
| Define sinusoidal drive function | func  sine 1e6 1  end | func(‘sine’, 1e6, 1); |
| Define piezoelectric window and apply and connect a ground electrode and an active electrode with the func drive function. | piez  wndo $i1 $i2 $j1 $j2  defn uppr  node $i2 $i2 $j1 $j2  defn lowr  node $i1 $i1 $j1 $j2  bc lowr grnd  bc uppr volt func  end | piez(‘wndo’, i(1), i(2), j(1), j(2));  piez(‘defn’, ‘uppr’);  piez(‘node’, i(2), i(2), j(1), j(2));  piez(‘defn’, ‘lowr’);  piez(‘node’, i(1), i(1), j(1), j(2));  piez(‘bc’, ‘lowr’, ‘grnd’);  piez(‘bc’, ‘upper’, ‘volt’,’func’); |
| Define pressure source to be applied in positive x direction on the left hand side of the model with func drive function | plod  pdef pld1 func  vctr vct1 1 0 0  sdef pld1 vct1 $i1 $i2 $j1 $j2  end | plod(‘pdef’, ‘pld1’, ‘func’);  plod(‘vctr’, ‘vct1’, 1, 0, 0);  plod(‘sdef’, ‘pld1’, ‘vct1’, i(1), i(2), j(1), j(2)); |
| Plot pressure and pressure in a sub region in two viewing windows | grph  nvew 2 1  plot pres  blok B1 1 4 1 10  plot pres blok b1  end | grph(‘nvew’, 2, 1);  grph(‘plot’, ‘pres’);  grph(‘blok’, ‘B1’, 1, 4, 1, 10);  grph(‘plot’, ‘pres’, ‘B1’); |