FDwave 1.0

Simulate seismic wave in earth and materials

Author: Ajay Malkoti

21 Nov 2019

Contents

Chapter 1

Introduction

FDwave is a a set of MATLAB functions which can simulate the wave propagation in a different rheologies (acoustic/elastic/viscoelastic/anisotropic). In the present version FDwave1.0 only elastic rheology is included. This code is intended for the use by students and researchers.

Code characteristics

- Based upon finite difference time domain modeling over- staggered grid in 2-dimension.
- Vectorized code for faster execution.
- Temporal and spatial accuracy of $(\Delta t^2, \Delta x^4)$
- Free surface and damping absorbing boundary condition can be applied.

Utilities

- Various rheologies available (e.g. acoustic, elastic, viscoelastic, anisotropic).
- Readily define the layered model, create used defined geometry, or read model from SEGY file.
- Various source wavelet are available
- Various geometries for source and receiver layout are available, e.g. straight line along surface, VSP, user defined.

Contents in the package   The package contains two directories

- FDwave: This directory contains all the programs & functions related to seismic modelling.
- scripts: This directory contains all the file containing commands used for the modelling.

Installation :
This package has been tested on windows environment and currently being under test on Linux/unix environment. This package is provided in "ready to use" form. The MATLAB-scripts-files from the script-folder can be run directly. Which utilize the `FDwave_initialization` function and its details are provided in the next chapter.

Author   : Ajay Malkoti (ajmalkoti@gmail.com)

Contributors : N Vedanti, RK Tiwari

Disclaimer :
The code comes without any guarantee and author will not be responsible for any kind of loss.

References :

This work has resulted in following articles. Kindly consider citing these if this code was useful in your work.

Malkoti, A., Vedanti, N., Tiwari, R. K., 2018. An algorithm for fast elastic wave simulation using a vectorized finite difference operator. Computers & Geosciences 116, 23-31

Malkoti, A., Vedanti, N., Tiwari, R. K., 2018. Viscoelastic modeling with a vectorized finite difference operator.

Chapter 2

Setup and running code

The program can be run either in two modes

- GUI mode, with limited functionality and

- COMMAND mode, which offers more options.

A quick outline/steps used in modelling is given below:

1. Initialization
2. Model building and plot it [1]
3. Source selection and plot it [1].
4. Analyse the parameters for various conditions
5. Derive the additional parameters.
6. Select the parameters for boundaries and plot it [1].
7. Generate the source and the receiver geometry and plot it [1].
8. Run the FD calculation for given shot and plot it [1].

Initialization and setting up the path :
For initialization we require two pats:

1. It is necessary to provide the path to directory were the code files can be located. It can be done as following:

```
1  code_path='F:\My_M_Codes\FDwave'; % path
2  addpath(code_path);                % Add the code folder to the current command space
```

2. Path to the working directory where data should be provided. In case when it is not provided then it is the automatically set to the current working directory. Following command shows how to setup a folder as the working directory for the first time.

```
1  code_path='F:\My_NVPAIR\FDwave';          % path
2  initialize('cp',code_path,'wfp',wf_path); % initialization of code
```

You may want to start with a fresh, clean installation by removing earlier output files/folders of previous simulations, run following.

```
1  code_path='F:\My_NVPAIR\FDwave';                                % path
2  initialize('cp',code_path,'wfp',wf_path,'clean','y'); % initialization (clean) of code
```

---

[1] Plotting is optional

Sample Program :

A sample program can be written as following:

```
1  -----------------------------------------------------------------------------
2  % Add the code folder to the current command space
3  code_path='F:\My_NVPAIR\FDwave';
4  addpath(code_path);
5
6  % Do necessary steps for initialization
7  wf_path='F:\My_Test';
8  initialize('cp',code_path,'wfp',wf_path);
9
10 % Initialize the model
11 model_n_layers('wave_type','Elastic','dx',10,'dz',10,'Thickness',[2000 2000],...
12         'HV_ratio',1,'Vp',[1700 2000],'Vs',[1500 1800],'Rho',[1600 1800],'PlotON','y');
13
14 % Generate the source wavelet
15 source_ricker('wfp',wf_path,'T',3,'dt',.001,'f0',15,'plotON','y');
16
17 % Analyse the parameters
18 analyse_elastic('wfp',wf_path)
19
20 % Derive additional parameters
21 model_derived_elastic_g1('wfp',wf_path)
22
23 % Select the Absorbing boundaries
24 bc_select('wfp',wf_path,'BCname','ABL','BCtype','topFS','nAB',50,'PlotON','y') ;
25
26 % Place a single source
27 geometry_src_single('wfp',wf_path,'PlotON','y')
28
29 % Place an array of receivers
30 geometry_rec_st_line_surf('wfp',wf_path);
31
32 % Run the simulations
33 calculation_elastic_g1('plotON','y');
34 -----------------------------------------------------------------------------
```

Each of the above command will be described later in respective chapters

Chapter 3

Model building

This package comes with many pre defined models. There is also a way to customize a model according to your preferences. For each model there are different model parameters and all of them are described below one by one.

3.1 N-Layered model

Complete description of command :

```
1   --------------------------------------------------------------------------------
2    MODEL_N_LAYERS
3     This function can create a model consisting of 'n' horizontal layers.
4     Complete Syntax:
5         model_N_layers('WFP',path,'WAVE_TYPE',options,'DX',value,'DZ',value,'THICKNESS',value,...
6             'HV_RATIO',value,'VP',value,'VS',value,'RHO',value,'QP',value,'QS',value,'PlotON',
7                 option )
8
9           Description of parameters:
10          WFP          :  Path to working folder
11          WAVE_TYPE    :  'acoustic1', 'acoustic2', 'elastic', 'viscoelastic'
12          DX, DZ       :  Grid size in horizontal and vertical direction in meters
13          THICKNESS    :  Thickness of each layer in form of vector
14          HV_RATIO     :  Total size of the model in respect to
15          VP, VS       :  Velocity of P and S wave in form of vector
16          RHO          :  Density of medium in form of vector
17          QP, QS       :  Attenutation of P and S value in form of vector
18          PlotON       :  'y'/'n' for plotting
19    Note:
20          acoustic1 - requires VP
21          acoustic2 - requires VP, RHO
22          Elastic  - requires VP, VS, RHO
23          viscoelastic - require VP, VS, RHO, QP, QS
24          The vector is in the form of e.g. [600,700,800,1000]
25    Example:
26        model_n_layers('WFP',wf_path,'WAVE_TYPE','Elastic','DX',4,'DZ',4,'THICKNESS'
                ,[250,500,300,450,600],'HV_RATIO',1,'Vp',2000:250:3000,'VS',1700:250:2700,'RHO'
                ,2300:100:2700,'PlotON','y')
27   --------------------------------------------------------------------------------
```

Demonstration/Example :
To create a N-stacked layer model we issue following command at MATLAB command prompt

```
1  >> model_n_layers('WFP',wf_path,'Wave_Type','Elastic','DX',4,'DZ',4,'Thickness'
       ,[250,500,300,450,600],'HV_Ratio',1,'Vp',2000:250:3000,'VS',1700:250:2700,'RHO'
       ,2300:100:2700,'PlotON','y')
```

Ouptut :
It causes some info to appear at the command screen, as shown below:

```
1  ------------------------------------------------------------------------
2  FUNC: Model Building
3     Model Selected: N-Layers
4          Provided parameters
5     Type of wave      : Elastic
6     Grid spacing along x : 4
7     Grid spacing along x : 4
8     X/Y dimension Ratio : 1      (Default)
9     Thickness of layer(s): 250 500 300 450 600
10    Vp of layer(s)     : 2000 2250 2500 2750 3000
11    Vs of layer(s)     : 1700 1950 2200 2450 2700
12    Density of layer(s) : 2300 2400 2500 2600 2700
13    dh = 4
14    dv = 4
15    nh = 527
16    nv = 527
17    Model saved inF:\My_Test\Data_IP\model
18 ------------------------------------------------------------------------
```

Following command can also be used to plot the model. However the plot will be same as before.

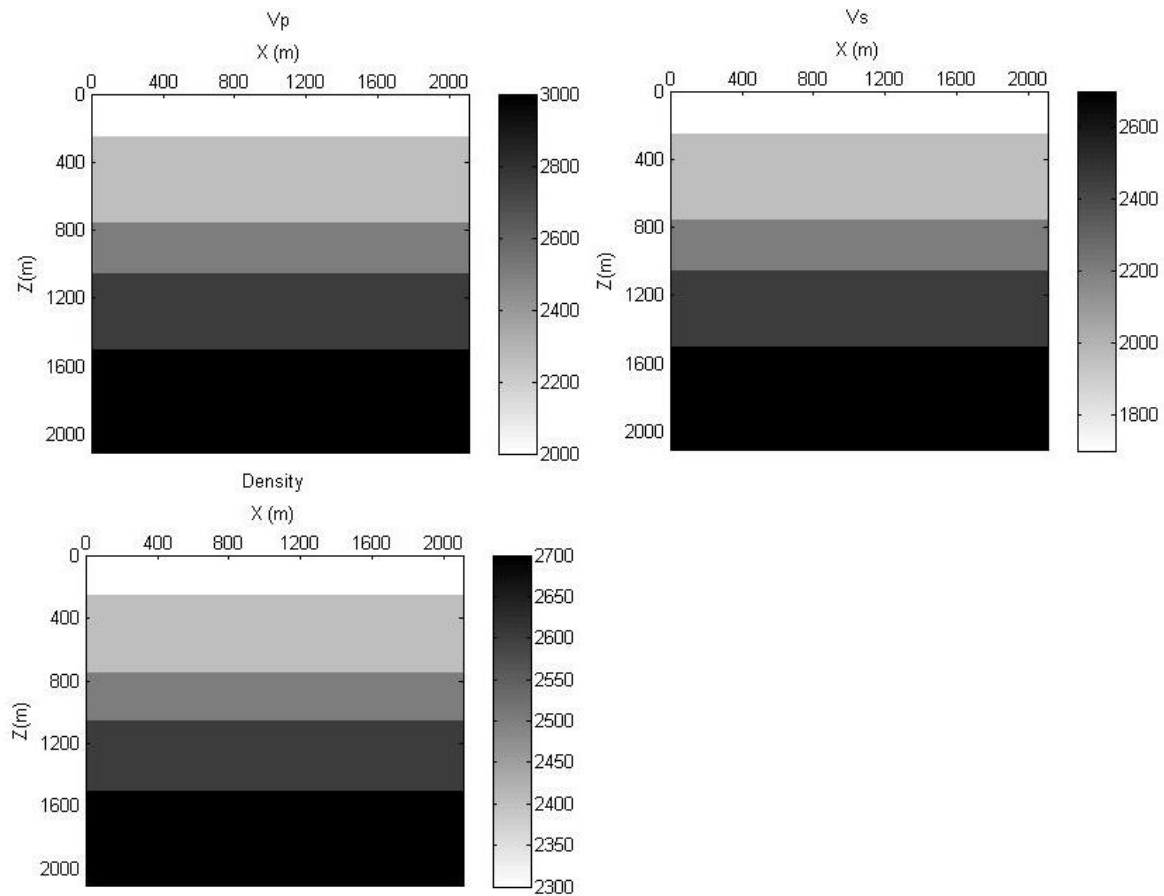```
>> model_plot()
```



Figure 3.1: A N-Layerd model generated using `N-Layered model` command

## 3.2 Marmousi model

This is one of the very famous model which was generated by Institut Français du Pétrole (IFP) in 1988. To use this model you many need to unzip the files first (provided in model folder). The model has a dimension of the model is 13601x2801 in grid point. Separation between the grid points is 1.5m.

**Complete description of command** :

```
1  --------------------------------------------------------------------------------
2  MODEL_READ_SEGY
3    This function can load the segy file of marmousi model which is
4    originally a fine scaled model with dx=dz=1.5m.
5    This func can CROP as well as INTERPOLATE the model to form a coarse/finer model
6  Complete Syntax:
7        model_read_segy('WFP',path,'M_NAME',name,'WAVE_TYPE',options,'CROP_MODEL',option, 'X1',
8              value, 'Z1',value, 'X2',value, 'Z2',value,'INTERPOLATE',option, 'DX_NEW',value,'
9              DZ_NEW',value,PAD,option,PADN,value,PADSIDE,option,'PlotON',option )
8  Description of parameters:
9        WFP         :  Path of the folder where model should be saved.
10       M_NAME      :  Name of the model (string)
11       WAVE_TYPE   :  'acoustic1', 'acoustic2', 'elastic', 'viscoelastic'
12
13       CROP_MODEL  :  'y/n'  ( If selected yes then following parameters must be provided:
14                      X1, Z1 represent the left upper corner of selected model
15                      X2, Z2 represent the right lower corner of selected model
16
17       INTERPOLATE :  'y'/'n'  ( If selected yes then following parameters must be provided:
18                      DX_NEW, DZ_NEW: new grid spacing(in meters) along x and z direction
19
20       PAD         :  'y'/'n' ( If selected yes then following parameters must be provided:
21                      PADN   : No of layers to be placed outside extra
22                      PADSIDE : 'tblr'/'blr'
23       PlotON      :  'y'/'n'
24  Example:
25        model_read_segy('WAVE_TYPE','elastic',...
26         'CROP_MODEL','y', 'X1',7000, 'Z1',600, 'X2',15000, 'Z2',4200,...
27         'INTERPOLATE','Y', 'DX_NEW',12.5,'DZ_NEW',12.5)
28
29  Note: Initialization must have been performed.
30        acoustic1 - requires VP
31        acoustic2 - requires VP, RHO
32        elastic  - requires VP, VS, RHO
33        viscoelastic - require VP, VS, RHO, QP, QS
34        The vector is in the form of e.g. [600,700,800,1000]
35  --------------------------------------------------------------------------------
```

Demonstration/Example :
To create marmousi model for a **Elastic wave** with **grid spacing** of 4m, while using only portion of $7000m < x < 12000m$ and $700m < z < 4200m$ issue following command:

```
1  >> model_read_segy('WFP',wf_path,'M_NAME','marmousi','WAVE_TYPE','Elastic','CROP_MODEL','y','X1'
2      ,7000,'Z1',700,'X2',12000,'Z2',4200,'INTERPOLATE','y','DX_NEW',4,'DZ_NEW',4,'PlotON','y')
```

Output of command :
It causes some information to apper on the command screen as shown below:

```
1  --------------------------------------------------------------------------------
2  >>FUNC: Model Building
3     Model Selected: MARMOUSI
4     Note: This function require the SegyMAT package installed
5           Provided parameters
```

```
 6      Type of wave  : Elastic
 7      Cropped       : Yes
 8      Interpolated  : Yes
 9            Final model properties
10      dh = 4
11      dv = 4
12      nh = 1250
13      nv = 875
14      Model saved in "F:\My_Test\Data_IP\model
15   --------------------------------------------------------------------------------
```
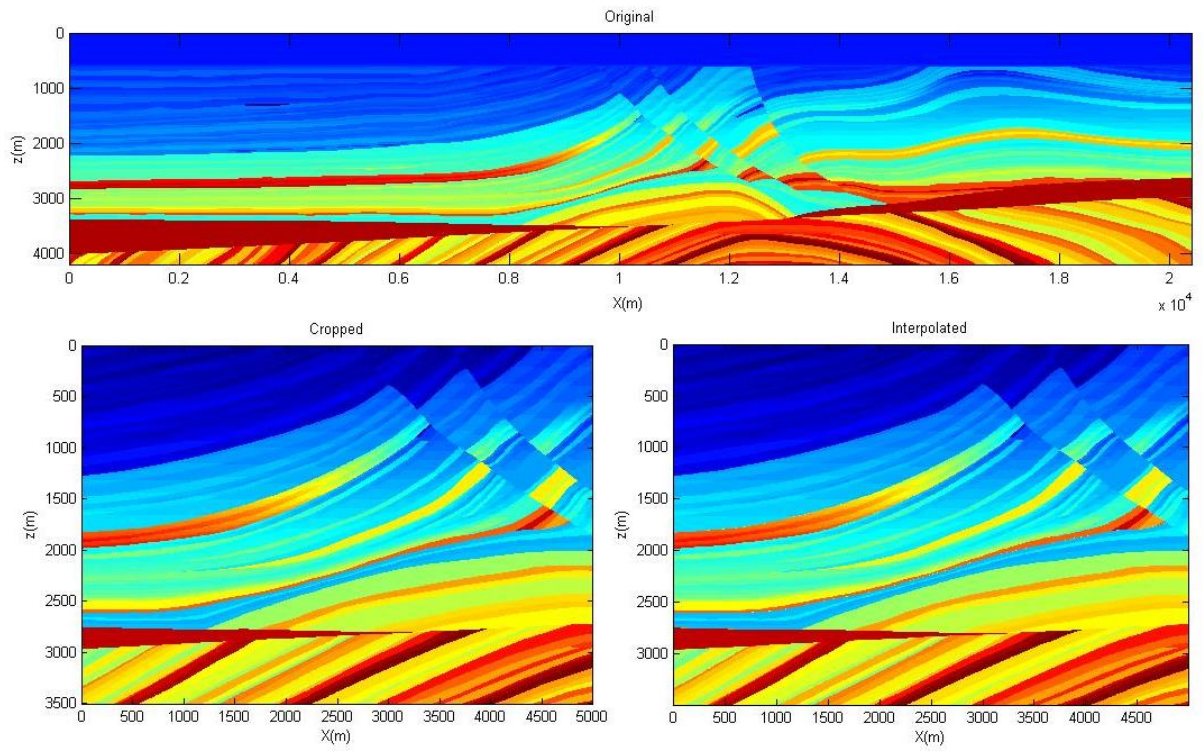

Figure 3.2: Above figure shows the original, cropped and interpolated stages/versions of Marmousi model which are generated using `model_read_segy` function.
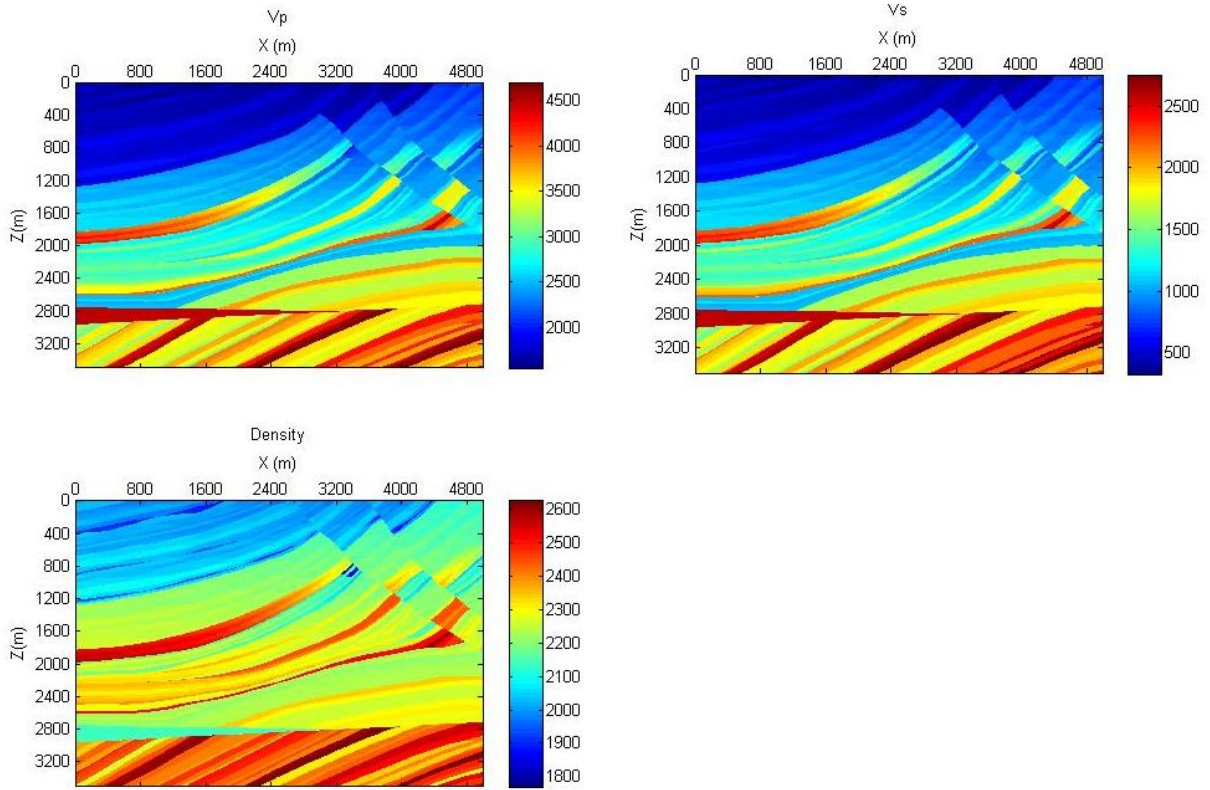
Figure 3.3: A cropped and interpolated Marmousi model generated using `model_read_segy` function. The positions are modifies for better visibility and understanding.

3.3 Build a NEW MODEL yourself

This program helps you to build a model in a layer-by-layer (or object-by-object) manner. It uses coordinates to define a layer/objects. Model building is achieved primarily through two steps.

1. Initialize the model: It creats the initial background model and saves it. The command name to initialize the model is **model_build_init**. The complete syntax for the model initialization command is

```
1  model_build_start('WFP',path,'WAVE_TYPE','elastic','NX',value,'NZ',value,'DX',value,'DZ',
       value,'VP',value,'VS',value,'RHO',value,'QP',value,'QS',value, 'PlotON',option)
```

2. Create layer(s): A single layer properties of the existing model are changed using the object/shape building function **model_build_shape_arbitrary**. In fact, if you want to change 'n' layers value then you may require to call this function n times. The syntax of the command is following

```
1  model_build_shape('WFP',path,'COORD',struc,'VP',value,'VS',value,'RHO',value,'QP',value,'QS
       ',value,'PlotON',option)
```

3.3.1 Command: MODEL_BUILD_INIT

Complete description of command (model_build_init) :

```
1  -------------------------------------------------------------------------------
2  MODEL_BUILD_INIT
3    This function can be used to create a single layer of arbitrary shape.
4    The shape is defined by a set of points connected linearly.
5    It doesn't create a new model but modifies the existing and then saves
6    new one.
```

```
 7    Complete Syntax:
 8        model_build_start('WFP',path,'WAVE_TYPE','elastic','NX',value,'NZ',value,'DX',value,'DZ',
                value,'VP',value,'VS',value,'RHO',value,'QP',value,'QS',value, 'PlotON',option)
 9
10   Description of parametes
11        WFP         :  Path to working directory
12        WAVE_TYPE   : 'acoustic1', 'acoustic2', 'elastic', 'viscoelastic'
13        NX,NZ       : No of grid points along x and z directions
14        DX,DZ       : Grid spacing along x and z
15        VP, VS      : Velocities of P and S wave
16        RHO         : Density of the material
17        QP, QS      : Attenuation values of the material.
18        PlotON      : 'y'/'n'
19   Note:
20        acoustic1 requires Vp
21        acoustic2 requires Vp, Rho
22        elastic requires Vp, Vs, Rho
23        viscoelastic require Vp, Vs, Rho, Qp, Qs
24        The structure for coordinats can be defined as {[x1,z1],[x2,z2],...}
25   Example:
26        model_build_start('WFP',pwd,'WAVE_TYPE','elastic','NX',200,'NZ',200,'DX',5,'DZ',5,'VP'
                ,2200,'VS',1800,'RHO',1800,'PlotON','y')
27   --------------------------------------------------------------------------------
```

Demonstration/Example :
Following command makes/initialize an homogeneous Elastic model of dimension (nx,nz), of specified grid spacing (dx,dz) and assigns the respective values for material parameters (vp, vs, etc.)

```
 1  >> model_build_init('WFP',wf_path,'Wave_Type','elastic','NX',500,'NZ',400,'DX',5,'DZ',5,'Vp'
        ,2200, 'VS',1800,'RHO',1600,'PlotON','y')
```

Output of command :
Above first command makes some information, about the created model, to appear at command window

```
 1  --------------------------------------------------------------------------------
 2  FUNC: Model Building
 3      Model Selected     : Build
 4          Provided parameters
 5      Type of wave       : Elastic
 6      No of grids along x : 500
 7      No of grids along x : 400
 8      Grid spacing x (m) : 5
 9      Grid spacing z (m) : 5
10      Vp (m/s)           : 2200
11      Vs (m/s)           : 1800
12      Density (g/cc)     : 1600
13          Final model properties
14      dh = 5
15      dv = 5
16      nh = 500
17      nv = 400
18      Model saved inF:\My_Test\Data_IP\model
19  --------------------------------------------------------------------------------
```
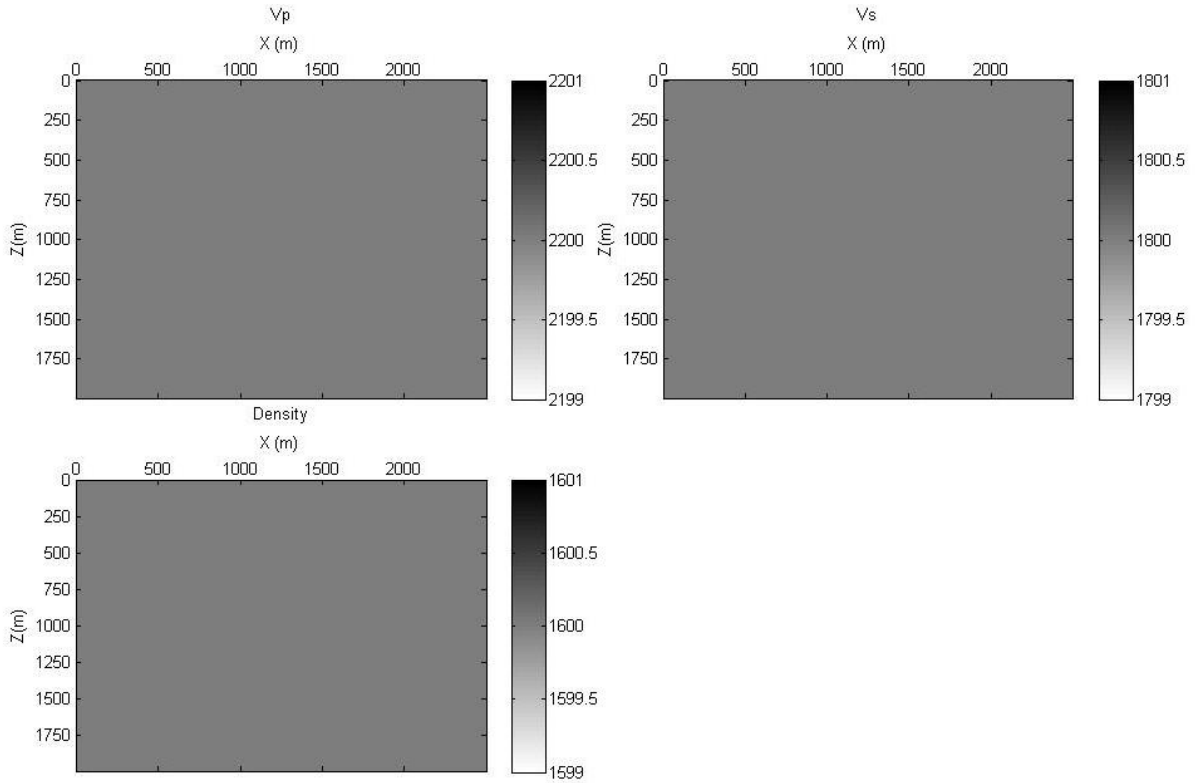
Figure 3.4: A model generated/initialized using `model_build_init` funtion.

3.3.2 Command MODEL_BUILD_SHAPE_ARBITRARY

Complete description of command (model_build_shape_arbitrary) :

```
1  ---------------------------------------------------------------------------
2  MODEL_BUILD_SHAPE_ARBITRARY
3    This function can be used to create a shape/object of arbitrary shape.
4    The shape is defined by a set of points connected linearly.
5    It doesn't create a new model but modifies the existing and then saves
6    new one.
7    Complete Syntax:
8        model_build_shape('WFP',path,'COORD',struc,'VP',value,'VS',value,'RHO',value,'QP',value,'
             QS',value,'PlotON',option)
9    Description of the parameters:
10       WFP          : Path to working directory
11       COORD        : A structure of coordinates of shape vertices
12       VP,VS        : P and S Velocities of the structure/object
13       RHO          : Density of the structure/object
14       QP, QS       : P and S Attenuations of the structure/object
15       PlotON       : 'y'/'n'
16   Note:
17       acoustic1 requires Vp
18       acoustic2 requires Vp, Rho
19       elastic requires Vp, Vs, Rho
20       viscoelastic require Vp, Vs, Rho, Qp, Qs
21       The structure for coordinats can be defined as {[x1,z1],[x2,z2],...}
22   Example:
23      model_build_shape_arbitrary('coordinates',CVec,'Vp',2800,'Vs',2200,'Rho',1800,'PlotON','y')
           ;
24  ---------------------------------------------------------------------------
```

Demonstration/Example :

Following are commands to create a corner model (to study diffractions). The rectangle shape/object is superimposed over the existing model.

```
1  >> CVec={[250,200],[500,200],[500,400],[250,400],[250,200]};
2  >> model_build_shape_arbitrary('WFP',wf_path,'Coord',CVec,'Vp',2800,'VS',2200,'RHO',1800,'PlotON
       ','y');
3  >> model_plot('WFP',wf_path)
```

Output of command There are two outputs-

- Intermediate output:
  It shows the current layer only (plotted by plotON option in command). such output for above command is shown below. The rectangle shown is created by above command only.
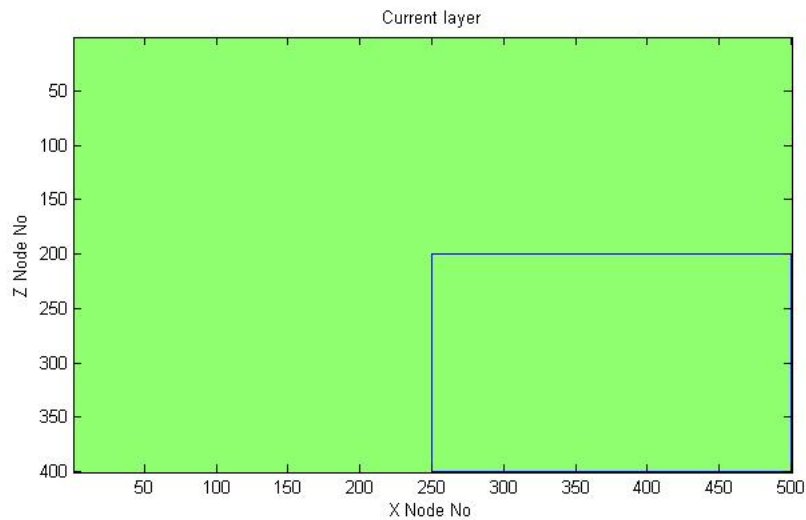

Figure 3.5: A homogeneous model is modified using `model_build_shape_arbitrary` funtion to include a wedge type structure.

- Final output: It shows the final model (plotted by model_plot command)

  It shows the final modified model achieve after initialization, previous and current operation of this command.
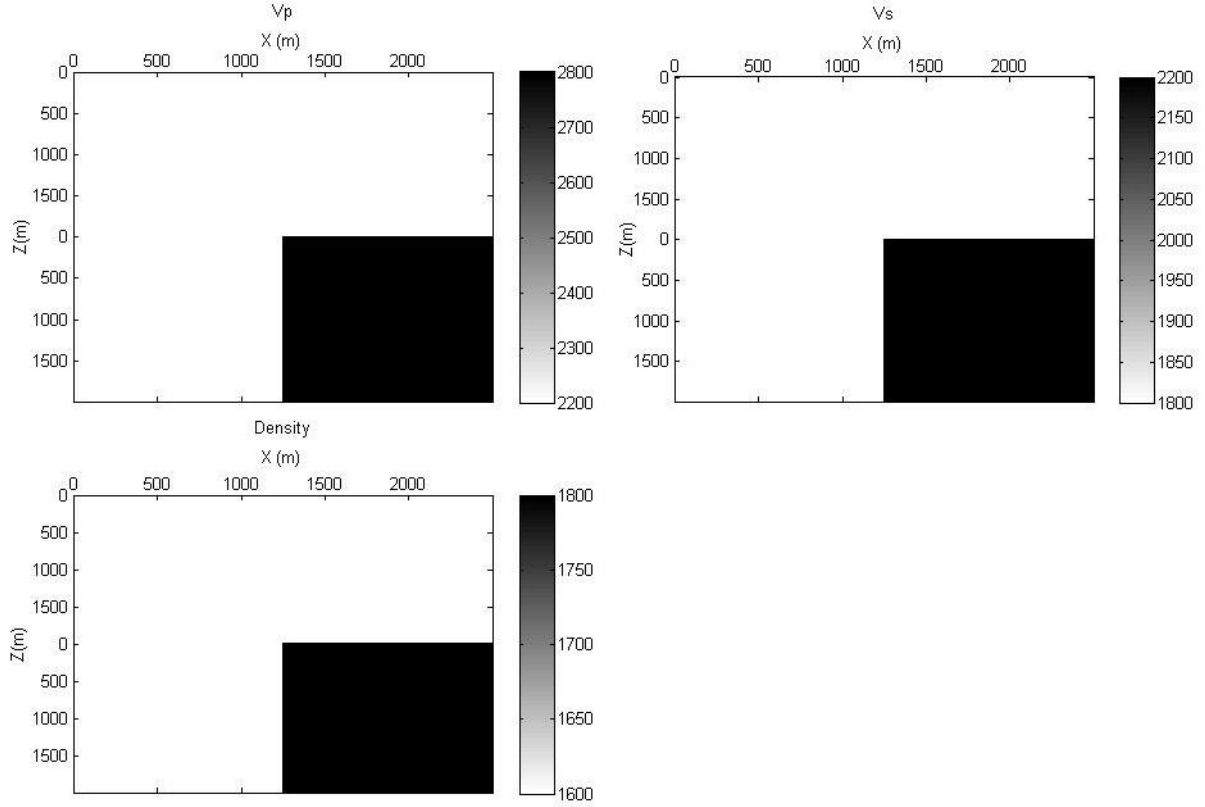
13

Figure 3.6: The final model produced after modifying the homogeneous model using `model_build_shape_arbitrary` funtion to include a wedge type structure.

3.4 Two working examples

Example 1 : Building a simple dipping layer model consists of 2 layers.

```
1  --------------------------------------------------------------------------------
2  % A dipping-layer model
3
4  % Initialize a model with a homogeneous background.
5  model_build_init('WFP',wf_path,'Wave_Type','Elastic','NX',500,'NZ',400,'DX',5,'DZ',5,'vp',2200,
      'VS',1800,'RHO',1600,'plotON','y')
6
7  % Create first dipping layer (at shallow depth).
8  CVec={[1,100],[500,200],[500,400],[1,400],[1,200]};
9  model_build_shape_arbitrary('WFP',wf_path,'coord',CVec,'VP',2500,'VS',2000,'RHO',1800,'plotON','
      y');
10
11  % Create second dipping layer (at greater depth).
12  CVec={[1,200],[500,400],[500,400],[1,400],[1,200]};
13  model_build_shape_arbitrary('WFP',wf_path,'coord',CVec,'VP',2800,'VS',2200,'RHO',2000,'plotON','
      y');
14
15  % Plot the final model.
16  model_plot('WFP',wf_path)
17  --------------------------------------------------------------------------------
```
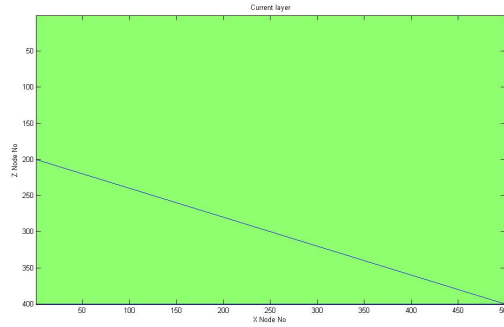
The starting/initialized homogeneous model is same as fig **??** shown earlier hence not shown again. The intermediate and final output of above command are following respectively.
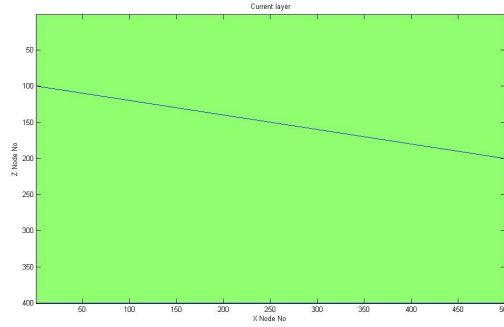
(a) Dipping Layer with gentle slope)

(b) Dipping Layer with steep slope)

Figure 3.7: Intermediate results for the 2 dipping layers model generated using recursive use of `model_build_shape_arbitrary` function.
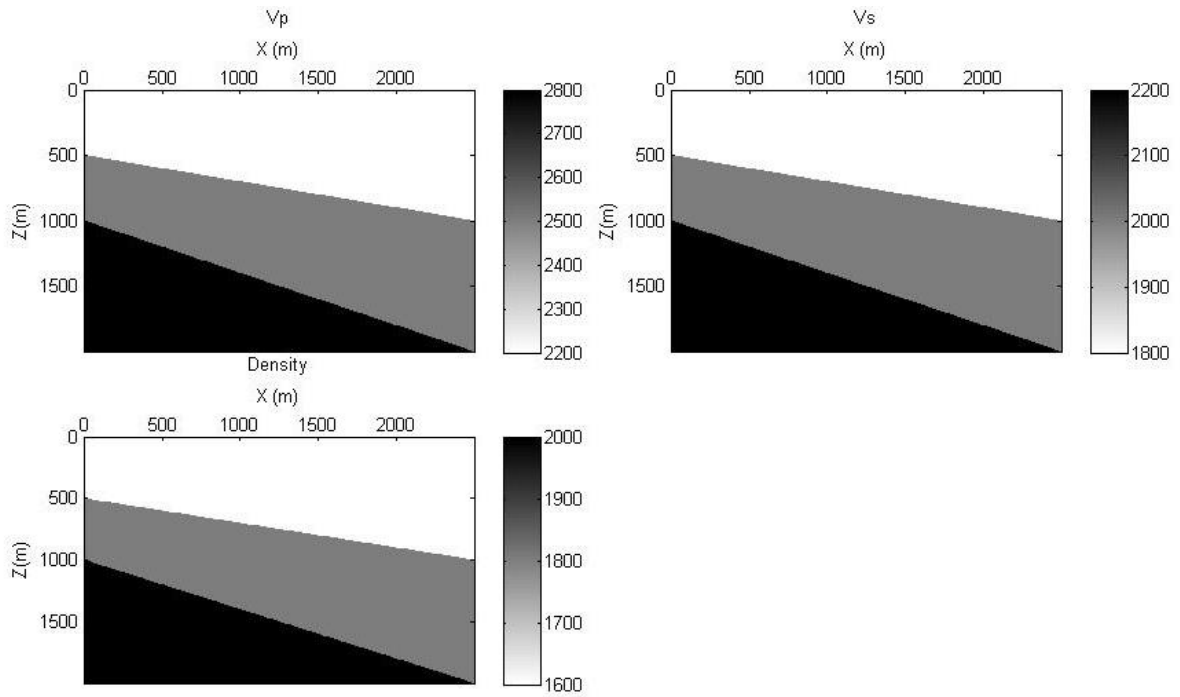

Figure 3.8: Final output for a 2-layers model, created using the `model_build_shape_arbitrary` function

Example 2: A wedge model with dipping layers

15

```
1   ------------------------------------------------------------------------------
2   % Building a complex model (wedge with dipping layers)
3
4   % Initialize a model with a homogeneous background
5   model_build_init('WFP',wf_path,'Wave_Type','Elastic','NX',500,'NZ',400,'DX',5,'DZ',5,'VP',2200,
        'VS',1800,'RHO',1600,'plotON','y')
6   model_plot2
7
8   % insert: a wedge
9   CVec={[200,200],[500,200],[500,300],[300,300],[200,200]};
10  model_build_shape_arbitrary('WFP',wf_path,'coord',CVec,'VP',2800,'VS',2200,'RHO',1800,'plotON','
        y');
11  model_plot('WFP',wf_path)
12
13  % insert: a dipping layer
14  CVec={[1,250],[500,300],[500,400],[250,400],[1,350],[1,250]};
15  model_build_shape_arbitrary('WFP',wf_path,'coord',CVec,'VP',3000,'VS',2400,'RHO',2000,'plotON','
        y');
16  model_plot('WFP',wf_path)
17
18  % insert: a triangle to fill rest of part,
19  CVec={[1,350],[250,400],[1,400],[1,350]};
20  model_build_shape_arbitrary('WFP',wf_path,'coord',CVec,'VP',3100,'VS',2500,'RHO',2100,'plotON','
        y');
21  model_plot('WFP',wf_path)
22  ------------------------------------------------------------------------------
```

Here we will be showing only the figures generated at intermediate steps and the final mode.

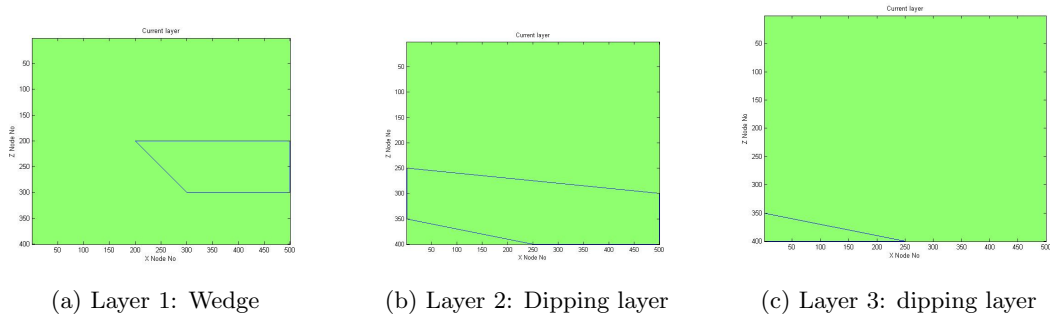| (a) Layer 1: Wedge | (b) Layer 2: Dipping layer | (c) Layer 3: dipping layer |

Figure 3.9: Intermediate results for the complex model generated using recursive use of model_build_shape_arbitrary function.
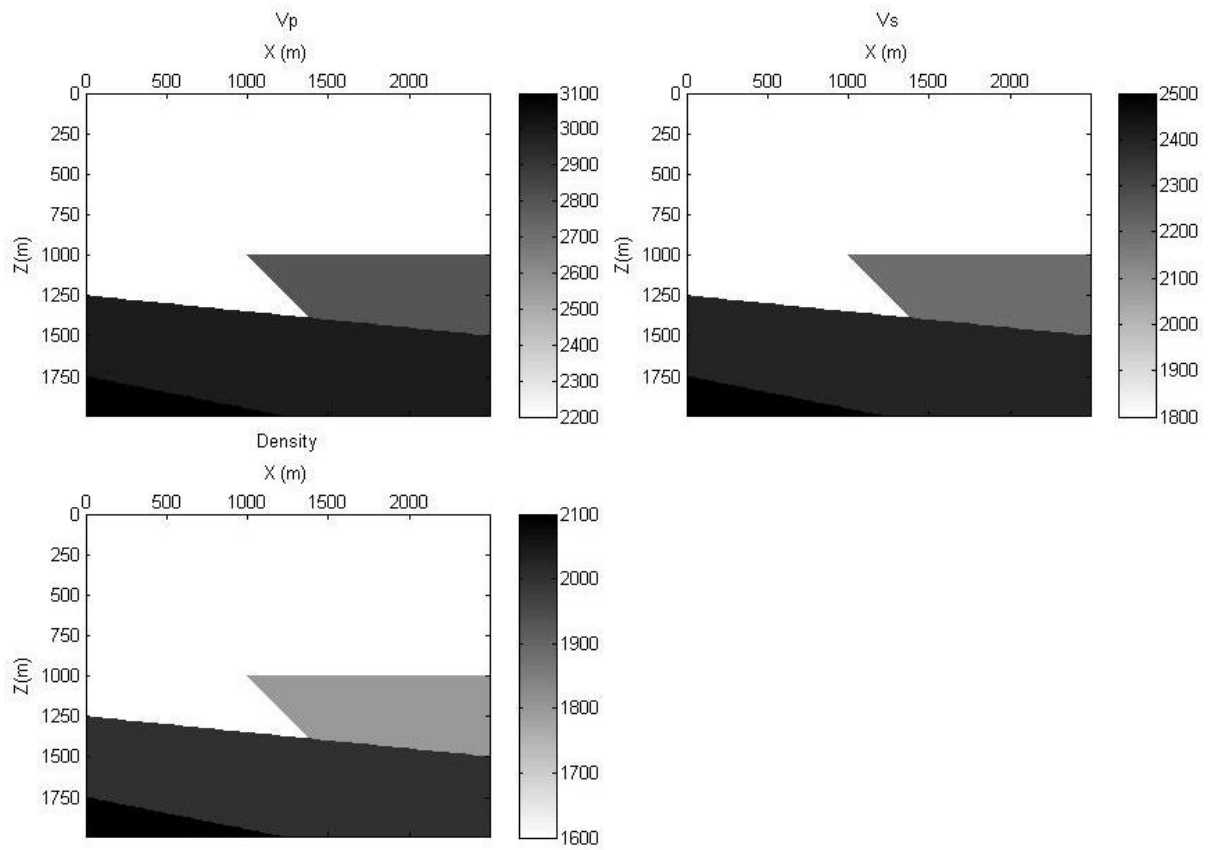
16

Figure 3.10: Final output for the complex-model, created using the `model_build_shape_arbitrary` function

Chapter 4

Source

FD wave provides various types of the sources as listed below

4.1 Sine source

Complete description of the command :

```
1   --------------------------------------------------------------------------------
2    SOURCE_SINE
3     This function generates the Sine wavelet signature.
4     Complete Syntax:
5         source_sine('WFP',path,'T',value, 'DT',value, 'F0',value, 'T0',value, 'SRC_SCALE',value,'
             PlotON',option)
6     Description of parameters:
7         WFP        :  Path to working directory
8         T          :  Total time duration for simulation.
9         DT         :  Time step
10        F0         :  Central frequency of source
11        T0         :  Zero offset time aka Lag time (optional)
12        SRC_SCALE  :  Scaling of amplitude (optional)
13        PlotON     :  'y'/'n'
14     Example:
15        source_sine('WFP',pwd,'T',2,'DT',.0004,'F0',10)
16        source_sine('T',2,'DT',.0004,'F0',10,'T0',.03,'SRC_SCALE',1,'PlotON','y');
17   --------------------------------------------------------------------------------
```

Demonstration/Example :

```
1   >> nvpair_source_sine('WFP',wf_path,'T',1,'DT',.001,'F0',20,'PlotON','y');
```

Output of the command :

```
1   --------------------------------------------------------------------------------
2    FUNC: Source parameters
3        Source type             :   Sine Wavelet
4        Source Frequency,    f0 = 20
5        Time step size,      dt = 0.001
6        Total time duration,  T = 1
7        Time shift,          t0 = 0       (Default)
8        Source amplitude scaling = 1      (Default)
9   --------------------------------------------------------------------------------
```
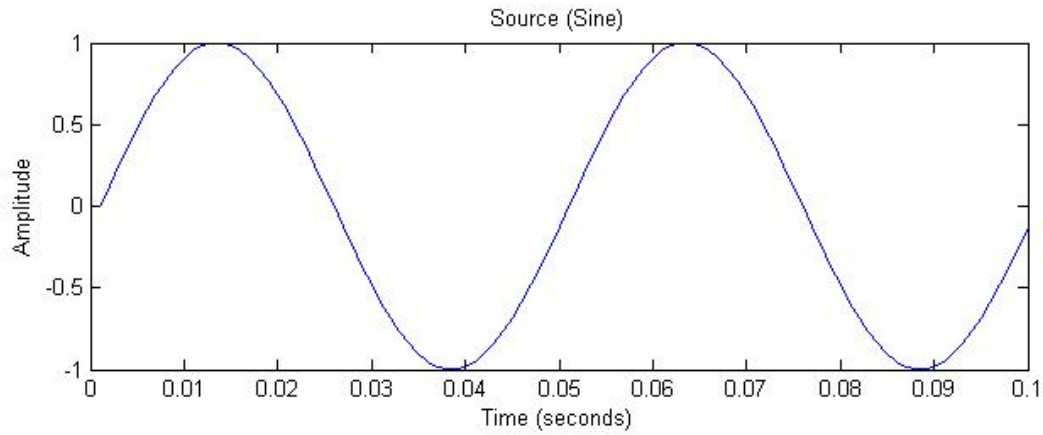
Figure 4.1: A sine function source signature generated using the command `nvpair_source_sine`

4.2 Gaussian Family wavelets

:

The Gaussian family wavelets include the

1. Gaussian wavelet

2. First derivative of Gaussian wavelet a.k.a. Maxican-Hat wavelet

3. Second derivative of Gaussian wavelet a.k.a. Ricker wavelet

The arguments for each of above are almost same (assuming alpha f0).

4.2.1 Gaussian wavelet

Complete description of the command :

```
1  -------------------------------------------------------------------------------
2    SOURCE_GAUSSIAN
3    This function generates the gaussian wavelet signature.
4    Complete Syntax:
5        source_gaussian('WFP',path,'T',value,'DT',value,'F0',value,'T0',value,'SRC_SCALE',value,'
              PlotON',option)
6    Description of parameters:
7        WFP         :  Path to working directory
8        T           : Total time duration for simulation.
9        DT          : Time step
10       ALPHA       : Related to central frequency of source
11       T0          : Zero offset time a.k.a. Lag time (optional)
12       SRC_SCLAE   : Scaling of amplitude (optional)
13       PlotON      : 'y' for plotting
14                     'n' for not plotting (default)
15   Example:
16       source_gaussian('WFP',pwd,'T',2,'DT',.0004,'F0',10)
17       source_gaussian('T',2,'DT',.0004,'F0',10,'T0',.03,'SRC_SCLAE',1,'PlotON','y');
18   -------------------------------------------------------------------------------
```

Demonstration/Example :

```
1  >> nvpair_source_gaussian('WFP',wf_path,'T',1,'DT',.001,'ALPHA',20,'PlotON','y');
```

19

Output of the command :

```
1  --------------------------------------------------------------------------------
2    FUNC: Source parameters
3         Source type             :    Gaussian Wavelet
4         Source parameter,   alpha = 20
5         Time step size,        dt = 0.001
6         Total time duration,   T = 1
7         Time shift,            t0 = 0.3        (Default)
8         Source amplitude scaling = 1        (Default)
9  --------------------------------------------------------------------------------
```
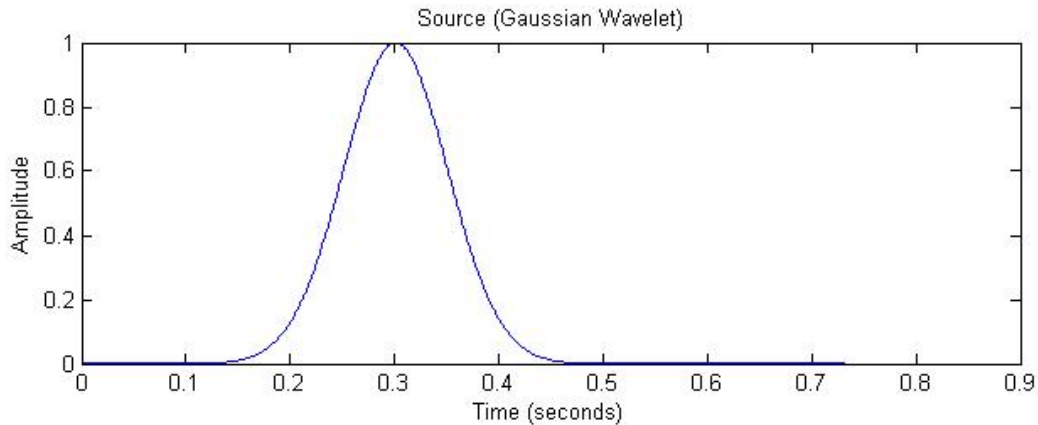

Figure 4.2: A sine function source signature generated using the command `sources_gaussian`

4.2.2 Maxican-Hat wavelet

Complete description of the command :

```
1  --------------------------------------------------------------------------------
2    SOURCE_GAUSSIAN1
3    This function generates the gaussian1 wavelet signature.
4    Complete Syntax:
5         source_gaussian1('WFP',path,'T',value, 'DT',value, 'F0',value, 'T0',value, 'SRC_SCALE',
               value,'PlotON',option)
6    Description of parameters:
7         WFP         :  Path to working directory
8         T           : Total time duration for simulation.
9         DT          : Time step
10        ALPHA       : Related to central frequency of source
11        T0          : Zero offset time aka Lag time (optional)
12        SRC_SCALE   : Scaling of amplitude (optional)
13        PlotON      : 'y' for plotting
14                      'n' for not plotting (default)
15   Example:
16        source_gaussian1('WFP',pwd,'T',2,'DT',.0004,'T0',10)
17        source_gaussian1('T',2,'DT',.0004,'F0',10,'T0',.03,'SRC_SCALE',1,'PlotON','y');
18  --------------------------------------------------------------------------------
```

Demonstration/Example :

```
1  >> nvpair_source_gaussian1('WFP',wf_path,'T',1,'DT',.001,'ALPHA',20,'PlotON','y');
```

Output of the command :

```
1 -------------------------------------------------------------------------------
2 FUNC: Source parameters
3       Source type              :   gaussian1 Wavelet
4       Source parameter,  alpha = 20
5       Time step size,      dt = 0.001
6       Total time duration,   T = 1
7       Time shift,           t0 = 0.3      (Default)
8       Source amplitude scaling = 1       (Default)
9 -------------------------------------------------------------------------------
```
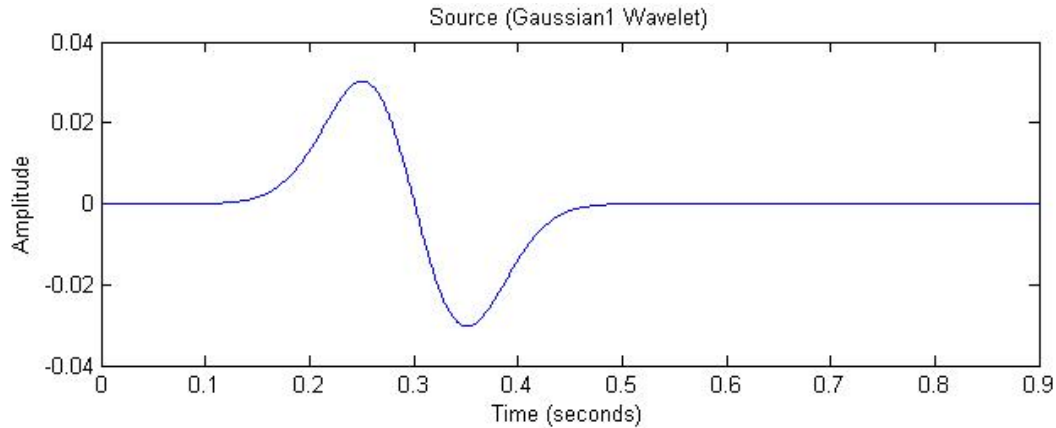

Figure 4.3: A sine function source signature generated using the command `sources_gaussian1`

4.2.3 Ricker wavelet

Complete description of the command :

```
1  -------------------------------------------------------------------------------
2   SOURCE_RICKER
3   This function generates the ricker wavelet signature.
4   Complete Syntax:
5       source_ricker('WFP',path,'T',value,'DT',value,'F0',value,'T0',value,'SRC_SCALE',value,'
           PlotON','y');
6   Description of parameters:
7       WFP        :  Path to working directory
8       T          : Total time duration for simulation.
9       DT         : Time step
10      F0         : Central frequency of source
11      T0         : Zero offset time aka Lag time (optional)
12      SRC_SCALE  : Scaling of amplitude (optional)
13      PlotON     : 'y'/'n'
14  Example:
15      source_ricker('WFP',pwd,'T',2,'DT',.0004,'F0',10)
16      source_ricker('T',2,'DT',.0004,'F0',10,'T0',.03,'SRC_SCALE',1,'PlotON','y');
17  -------------------------------------------------------------------------------
```

Demonstration/Example :

```
1 >> nvpair_source_ricker('WFP',wf_path,'T',2,'DT',.0003,'F0',15,'T0',0.07,'PlotON','y');
```

Output of the command :

```
1 -------------------------------------------------------------------------------
2    FUNC: Source parameters
3       Source type              :   Ricker Wavelet
4       Source Frequency,    f0 = 15
```

21

```
5        Time step size,        dt = 0.0003
6        Total time duration,   T = 2
7        Time shift,            t0 = 0.07
8        Source amplitude scaling = 1        (Default)
9   --------------------------------------------------------------------------------
```
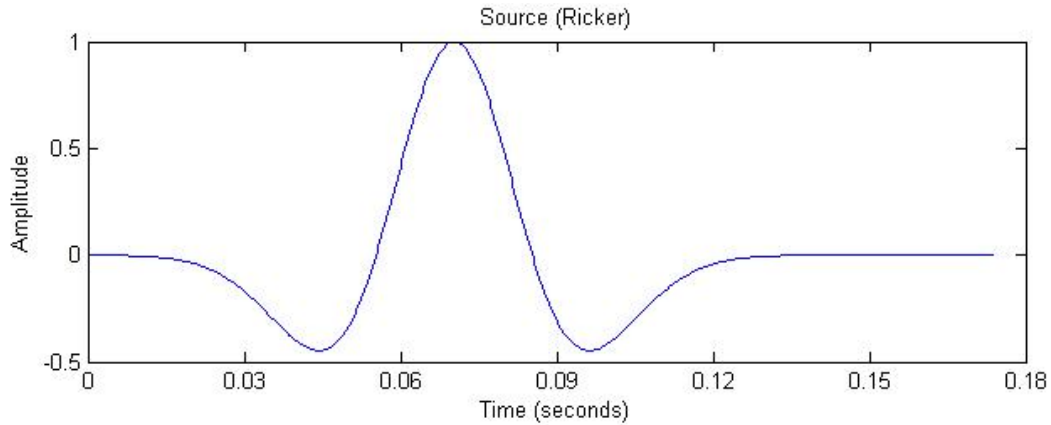

Figure 4.4: A sine function source signature generated using the command `sources_ricker`

4.3   Plotting function

The plotting function is very useful if you want to modify the plotting pattern. It takes the stored wavelet and plot it according to given parameters.

```
--------------------------------------------------------------------------------
SOURCE_PLOT
  This function generates the Sine wavelet signature.
  Complete Syntax:
        source_plot('WFP',path,'Wave_Type',option,'FigNO',value,'I1',value,'I2',value,'I3',value)
  Description of parameters:
        WFP          : Path to working directory
        Wave_Type    : Type of wave  e.g. Elastic
        FigNO        : Figure number
        I1,I2,I3     : subfigure index
  Example:
        source_plot('WFP',pwd,'FigNO',1,'I1',1,'I2',1,'I3',1)
--------------------------------------------------------------------------------
```

Chapter 5

Set-up Boundaries

There are many types of boundaries used in seismic modelling depending upon the type of wave used. For now we will use non-reflecting boundary condition [1].

In this method a damping layer consisting of some nodes is wrapped around the model so that the reflection from edges can be suppressed.

To set up boundaries it is necessary to define the model first. We will be using a N-Layer model which can be created using following command.

```
1 >> nvpair_model_n_layers('WFP',wf_path,'WAVE_TYPE','Elastic','DX',4,'DZ',4,'THICKNESS'
     ,[250,500,300,450,600],'HV_RATIO',1,'Vp',2000:250:3000,'VS',1700:250:2700,'RHO'
     ,2300:100:2700,'PlotON','y')
```

Description of command :

```
1  ------------------------------------------------------------------------------
2    SELECT_BC
3    It generate the necessary data for different BC
4    Complete Syntax:
5        select_bc('WFP',path,'BCNAME',value,'BCTYPE',value,'NAB',value,'PlotON',option)
6    Description of parameters:
7        WFP   : Path to working directory
8        BCNAME = 'ABC1' for Englist & Clayton EnglistClayton (acoustic wave eqn) (future
               implementation)
9                 'ABC2'  for Raynold EnglistClayton (acoustic wave eqn)(future implementation)
10                'ABL'   for Damping EnglistClayton, after Cerjan
11       BCTYPE = 'topFS'  for free surface at top
12              = 'topABC' for absorbing type boundary at surface
13       NAB   = no of layers to be used (for damping BC only)
14       PlotON = 'y', To plot the boundaries
15   Example:
16       select_bc('WFP',pwd,'BCNAME','ABL','BCTYPE','topABC','NAB',40)
17  ------------------------------------------------------------------------------
```

Note: Only damping type of boundaries will not work with elastic wave equation. Other boundaries are retained for future implementations.

Demonstration/Example: Top Free, others as absorbing surfaces :

For the free surface following command can be used

```
1 >> nvpair_bc_select('WFP',wf_path,'BCTYPE','topFS','NAB',50,'PlotON','y') ;
```

Output of the command :

```
1  ------------------------------------------------------------------------------
2      FUNC: Select the boundary types
3          BC Method  :  Damping Layer (Cerjan)
```

---

[1]Cerjan C. Kosloff D. Kosloff R. Reshef M., 1985. A non-reflecting boundary condition for discrete acoustic and elastic wave equation, Geophysics , 50, 705–708.

```
4        No of Layers : 50
5        BC type     : topFS
6        BC name     : ABL
7   --------------------------------------------------------------------------------
```
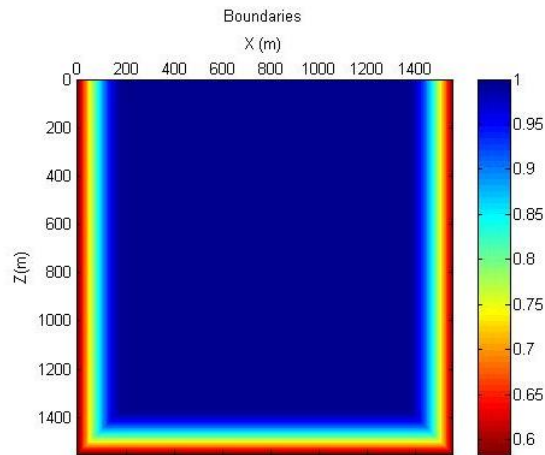

Figure 5.1: Absorbing boundaries generated using the `select_bc` function.

Demonstration/Example: All side absorbing layer :
To place absorbing boundaries all around the model following command can be used.

```
1   >> nvpair_bc_select('WFP',wf_path,'BCTYPE','topABC','NAB',50,'PlotON','y') ;
```

Output of the command :

```
1   --------------------------------------------------------------------------------
2      FUNC: Select the boundary types
3        BC Method   :  Damping Layer (Cerjan)
4        No of Layers : 50
5        BC type     : topABC
6        BC name     : ABL
7   --------------------------------------------------------------------------------
```
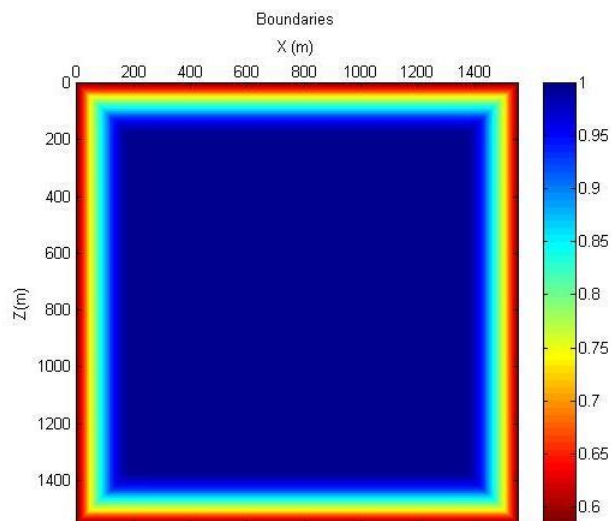

Figure 5.2: Absorbing boundaries generated using the `select_bc` function.

Chapter 6

Setup acquisition geometry

6.1 Introduction

Source and receivers can be laid out in various configurations. In FDwave there are primarily three ways to create geometry.

1. Along surface (conventional reflection/refraction surveys)

2. Along vertical (similar to VSP)

3. User defined position

In following sections we have explained these geometries one by one for source and receivers respectively. This modules assume model and boundaries has been created. Hence we will be assuming following model for creating different geometries.

```
1  >> % Create Model
2  >> nvpair_model_n_layers('wave_type','Elastic','DX',4,'DZ',4,'Thickness',[200,300,150,400,500],
       'HV_ratio',1,'VP',2000:250:3000,'VS',1700:250:2700,'RHO',2300:100:2700);
3
4  >> % Create boundaries
5  >> nvpair_bc_select('BCname','ABL','BCtype','topFS','nAB',50,'plotON','y' ) ;
```

Note 1: Boundary creation is not necessary but important for plotting part. It helps you to avoid you to place any source or receiver in absorbing layer.

Note 2: Later in this chapter, we will explain three plotting functions which are handy for visualizing the position of source and receiver with/without model and will be used frequently.

6.2 Source geometry

6.2.1 Single source

```
1   GEOMETRY_SRC_SINGLE
2   It places a single source at the given location
3   Check is applied so that no reciever is placed in absorbing boundary.
4   Complete Syntax:
5       geometry_src_Single('WFP',path,'SX',value, 'SZ',value,'DX',value,'DZ',value,'NX',value,
           'NZ',value,'BCTYPE',value, 'NAB',value)
6   Description of parameters:
7       WFP   :  Path to working directory
8       SX    :  Node Location of source (x-position)
9       SZ    :  Node Location of source (z-position)
10      DX    :  Grid size in x direction
11      DZ    :  Grid size in y direction
12      NX    :  No of nodes in model along X axis
13      NZ    :  No of nodes in model along Z axis
14      BCTYPE: Type of boundary used
15      NAB   :  Number of grid nodes used for boundaries.
```

```
16          PlotON: 'y'     to plot the source positions
17    Note:
18    1) Do not place source very close to surface.
19    2) Give all the position after adding/subtracting the ABC layer thickness
20       according to side where they are added.
21       e.g. if you want to place source at some depth then grid location for
22          topABC is then, Sz = nAB + dept/dz
23    3) All parameters are mandatory.
24       In case any parameter is not provided the program will try to use
25       the values stored in input folder.
26       If it cannot find any stored value then it shows error.
27    Example:
28       geometry_src_Single('WFP',pwd,'Sx',2000,'Sz',300,'dx',5,'dz',5,'BCtype','topABC','nAB'
              ,50,'PlotON','Y');
29       geometry_src_Single('Sx',2000,'Sz',300)
30       geometry_src_Single()
```

Demonstration/Example :

```
1  >> nvpair_geometry_src_single('WFP',wf_path,'SX',200,'SZ',200,'PlotON','y');
```

Output of command :

```
1  FUNC: Source geometry (Single at surface)
2       Source location along x(m)    :  800
3       Source location along z(m)    :  800
4       Source node location along x : 200
5       Source node location along z : 200
6       Note: the program take care of absorbing boundary nodes (if present).
```
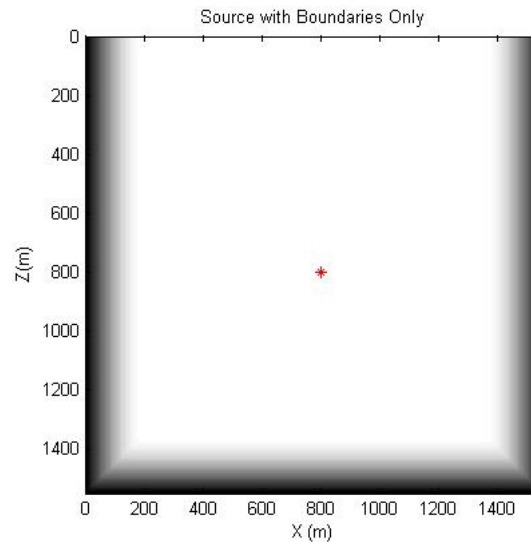

Figure 6.1: Source placed at center with the help of function `nvpair_geometry_src_single`

6.2.2   Array of sources at surface

Description of command :

```
1  -----------------------------------------------------------------------------------------
2  GEOMETRY_SRC_ST_LINE_SURF
```

```
 3    It places sources along the surface at a given depth.
 4    Complete Syntax:
 5        geometry_src_StLine_Surf('WFP',path,'Depth',value,'FIRST',value,'LAST',value,...
 6        'DIFF',value,'DX',value,'DZ',value,'BCTYPE',value,'NAB',value,'PlotON',option)
 7    Description of parameters:
 8        WFP   : Path to working directory
 9        DEPTH : Depth in terms of nodes
10        FIRST : Location of first Source along x in terms of nodes
11        LAST  : Location of last Source along x in terms of nodes
12        DIFF  : Difference in consecutive Source along x in terms of nodes
13        DX    : Grid size in x direction
14        DZ    : Grid size in y direction
15        BCTYPE: Type of boundary used
16        NAB   : Number of grid nodes used for boundaries.
17        PlotON: 'y'     to plot the source positions
18    Note:
19    1) Give all the position after adding/subtracting the ABC layer thickness according to side
          where they are added.
20    2) All parameters are mandatory.
21        In case any parameter is not provided the program will try to use the values stored in
            input folder.
22        If it cannot find any stored value then it shows error.
23    Example:
24        geometry_src_StLine_Surf('DEPTH',300,'FIRST',260,'LAST',4400,'DIFF',20,'DX',5,'DZ',5,'
            BCTYPE','topABC','NAB',50);
25        geometry_src_StLine_Surf('WFP',path,'DEPTH',300,'FIRST',260,'LAST',4400,'DIFF',20)
26        geometry_src_StLine_Surf()
27    --------------------------------------------------------------------------------------------
```

Demonstration/Example :

```
 1   >> nvpair_geometry_src_st_line_surf('WFP',wf_path,'DEPTH',10,'FIRST',55,'LAST',330,'DIFF',5,'
        PlotON','y');
```

Output of command :

```
 1  FUNC: Source geometry (Along surface)
 2    First source location (m) : 220      (Default, stored)
 3    Last source location (m) : 1320       (Default, stored)
 4    Distance between two consecutive source : 100    (Default, stored)
 5  Position of source on grid
 6    Depth of all source (in terms of node) : 10      (Default, stored)
 7    First source location (in terms of node) : 55    (Default, stored)
 8    Last source location (in terms of node) : 330     (Default, stored)
 9    Distance between two consecutive source (in terms of node): 25  (Default, stored)
10  Note:
11    Node position in grid(N) & distances(X) related by (N= nAB + X/h), due to ABC layer.
```
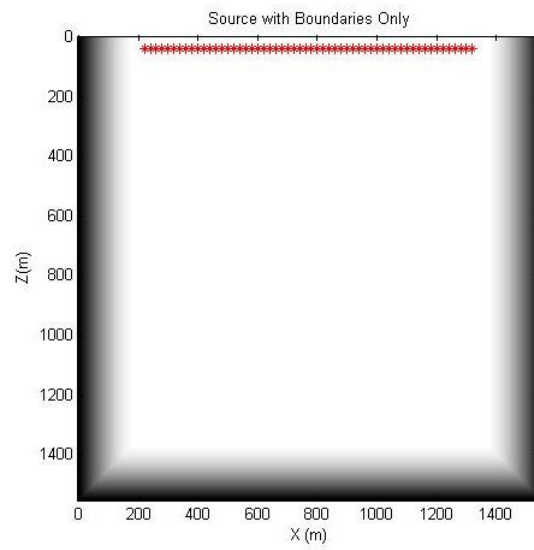
Figure 6.2: Placing an array of sources at the surface with the help of function nvpair_geometry_src_StLine_Surf

6.2.3 User defined array of sources

Description of command :

Demonstration/Example :

Output of command :

6.3 Receiver Geometry

6.3.1 Array of receivers at surface

Description of command :

```
1    GEOMETRY_REC_ST_LINE_SURF
2    It places recievers along the surface at a given depth.
3    Check is applied so that no reciever is placed in absorbing boundary.
4    Complete Syntax:
5         geometry_rec_st_line_surf('WFP',path,'DEPTH',value, 'FIRST',value, 'LAST',value,...
6         'DIFF',value, 'DX',value, 'DZ',value, 'BCTYPE',value, 'NAB',value,'PlotON',option)
7    Description of parameters:
8         WFP   : Path to working directory
9         DEPTH : Depth in terms of nodes
10        FIRST : Location of first Receiver along x in terms of nodes
11        LAST  : Location of last Receiver along x in terms of nodes
12        DIFF  : Difference in consecutive Receiver along x in terms of nodes
13        DX    : Grid size in x direction
14        DZ    : Grid size in y direction
15        BCTYPE: Type of boundary used
16        NAB   : Number of grid nodes used for boundaries.
17        PlotON: 'y'/'n'
18   Note:
19   1) Give all the position after adding/subtracting the ABC layer thickness according to side
         where they are added.
20   2) All parameters are mandatory.
21        In case any parameter is not provided the program will try to use the values stored in
             input folder.
22        If it cannot find any stored value then it shows error.
23   Example:
24        geometry_rec_st_line_surf('WFP',pwd,'DEPTH',300,'FIRST',260,'LAST',4400,'DIFF',20,'DX',5,
             'DZ',5,'BCTYPE','topABC','NAB',50);
25        geometry_rec_st_line_surf('DEPTH',300,'FIRST',260,'LAST',4400,'DIFF',20)
26        geometry_rec_st_line_surf()
```

Demonstration/Example :
For the model defined in introduction part we can define the

```
1   >> nvpair_geometry_rec_st_line_surf('WFP',wf_path,'DEPTH',10,'FIRST',55,'LAST',330,'DIFF',5);
```

Output of command :

```
1   FUNC: Reciever geometry (Along surface)
2        First receiver location (m) : 220      (Default, stored)
3        Last receiver location (m) : 1320       (Default, stored)
4        Distance between two consecutive Receiver : 8    (Default, stored)
5   Position of receiver on grid
6        Depth of all receiver (in terms of node) : 10      (Default, stored)
7        First receiver location (in terms of node) : 55    (Default, stored)
8        Last receiver location (in terms of node) : 330     (Default, stored)
9        Distance between two consecutive Receiver (in terms of node): 2  (Default, stored)
10  Note:
11       Node position in grid(N) & distances(X) related by (N= nAB + X/h), due to ABC layer.
```
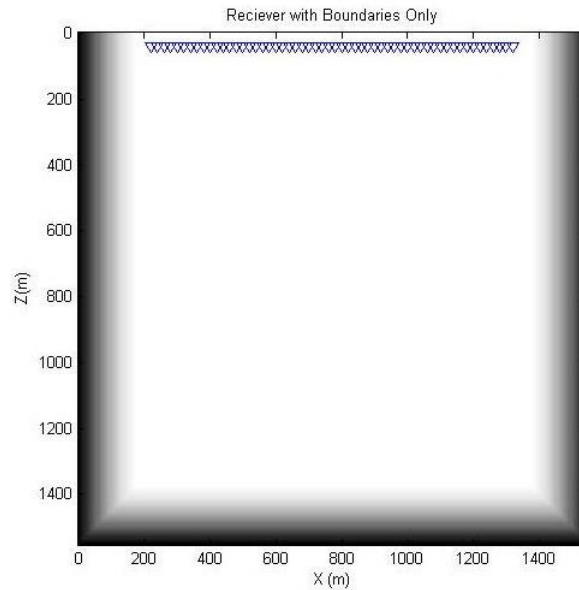
Figure 6.3: An arry of receiver placed at surface using function `nvpair_geometry_rec_StLine_Surf`.

6.3.2 Array of receivers along borehole/depth aka VSP

Description of command :

```
1   GEOMETRY_REC_STLINE_VSP
2    It places recievers vertically along the depth at some given x.
3    The configration is similar to the VSP configration of receivers.
4      Complete Syntax:
5          geometry_rec_StLine_VSP('HLOC',value,'FIRST',value,'LAST',value,'DIFF',value...
6                'DX',value,'DZ',value,'BCTYPE',option,'NAB',value,'PlotON',option)
7    Description of parameters:
8                HLOC  :  Horizontal location in terms of nodes
9                FIRST :  Location of first Receiver along x in terms of nodes
10               LAST  :  Location of last Receiver along x in terms of nodes
11               DIFF  :  Difference in consecutive Receiver along x in terms of nodes
12               DX    :  Grid size in x direction
13               DZ    :  Grid size in y direction
14               BCTYPE : Type of boundary used
15               NAB   :  Number of grid nodes used for boundaries.
16               PlotON : 'y'/'n'
17   Note:
18   1) Give all the position after adding/subtracting the ABC layer thickness according to side
          where they are added.
19     Node position in grid(N) & distances(X) related by (N= nAB + X/h), due to ABC layer.'
20   2) In case of free surface two topmost node layers are ignored (imaging technique for Free
          surface)
21     Hence a node vertical position is given by N= (X/h)-2;
22
23   3) All parameters are mandatory.
24       In case any parameter is not provided the program will try to use the values stored in
              input folder.
25       If it cannot find any stored value then it shows error.
26   Example:
27       geometry_rec_StLine_VSP('HLOC',500,'FIRST',5,'LAST',1000,'Diff',5,'DX',5,'DZ',5,'BCTYPE',
              'topABC','NAB',50,'PlotON','y')
28       geometry_rec_StLine_VSP('HLOC',500,'FIRST',5,'LAST',1000,'DIFF',5,'PlotON','y')
29       geometry_rec_StLine_VSP()
```

Demonstration/Example :

```
1  >> nvpair_geometry_rec_st_line_vsp('HLOC',250,'FIRST',5,'LAST',350,'DIFF',10,'PlotON','y');
```

Output of command :

```
1  FUNC: Receiver geometry Along Depth (or VSP type)
2      Horizontal position of receiver (m)   : 1000      (Default, stored)
3      First receiver depth (m)              : 12        (Default, stored)
4      Last receiver depth (m)               : 1392      (Default, stored)
5      Distance between two consecutive receivers : 40    (Default, stored)
6
7      Position of source on grid
8      Horizontal position of all receivers (in terms of node) : 250      (Default, stored)
9      First receiver location (in terms of node)              : 5        (Default, stored)
10     Last receiver location (in terms of node)               : 350      (Default, stored)
11     Distance between two consecutive receiver (in terms of node): 10  (Default, stored)
```
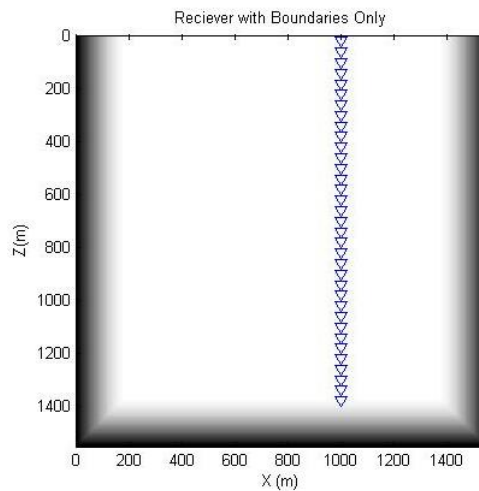

Figure 6.4: An array of receivers is placed along depth using function geometry_rec_st_line_vsp

6.3.3   User defined array of receivers

Description of command :

Demonstration/Example :

Output of command :

6.4   Plotting functions

In case of very dense arrays the plotting all objects (source and receivers) can obscure each other. For this reason separate functions are provided.

6.4.1   Source plotting function

Description of command :

```
1  GEOMETRY_PLOT_SRC
2      This function can be used to plot the source over the with/without model.
3  Complete syntax:
4      geometry_plot_src('figno',value,'hop',value,'snh_vec',vector,'snv_vec',vector)
5  Description of parameters:
6      FIGNO  : Figure no in which to be plotted.
```

```
7              If blank then plotted in new figure.
8       HOP    : No of object (sources) to be skipped
9       snx_vec : A vector containing X location of all sources
10      snz_vec : A vector containing Z location of all sources
11      BCPLOT  : To plot the source with boundries
12      MODELPLOT: To plot the source with boundaries and model
13 Example:
14          nvpair_geometry_plot_src('figno',3,'hop',5)
```

Demonstration/Example :

```
1 >> nvpair_geometry_src_st_line_surf('WFP',wf_path,'DEPTH',10,'FIRST',55,'LAST',330,'DIFF',5,'
     PlotON','y');
2 >> geometry_plot_src('WFP',wf_path,'HOP',5,'BCPLOT','y','MODELPLOT','y')
```
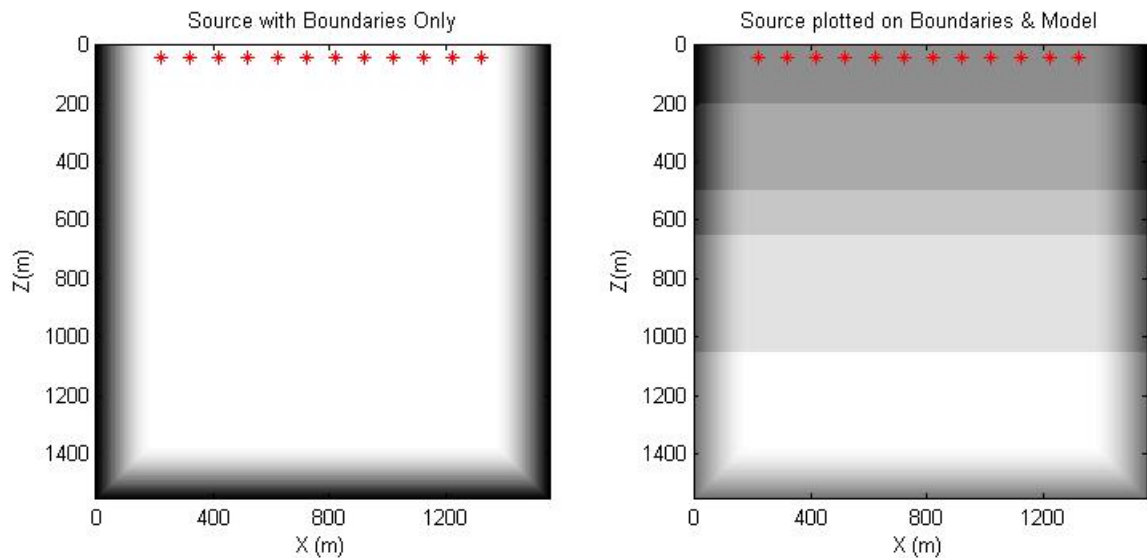
Output of command :

Figure 6.5: An array of receiver is plotted such that it doesn't show all sources by plotting every 5th only.

6.4.2   Receiver plotting function

Description of command :

```
1 GEOMETRY_PLOT_REC
2      This function can be used to plot the recievers over the with/without model.
3 Complete syntax:
4       geometry_plot_rec('WFP',path,'FIGNO',value,'HOP',value,'RNX_VEC',vector,'RNZ_VEC',vector)
5 Description of parameters:
6           WFP     : Path to working directory
7           FIGNO   : Figure no in which to be plotted.
8                     If blank then plotted in new figure.
9           HOP     : No of object (receivers) to be skipped
10          RNX_VEC : A vector containing X location of all sources
11          RNZ_VEC : A vector containing Z location of all sources
12 Example:
13          nvpair_geometry_plot_rec('WFP',pwd,'figno',3,'hop',5)
```

```
1  >> nvpair_geometry_rec_st_line_surf('WFP',wf_path,'DEPTH',10,'FIRST',55,'LAST',330,'DIFF',5);
2  >> geometry_plot_rec('WFP',wf_path,'BCPLOT','y','MODELPLOT','y');
```
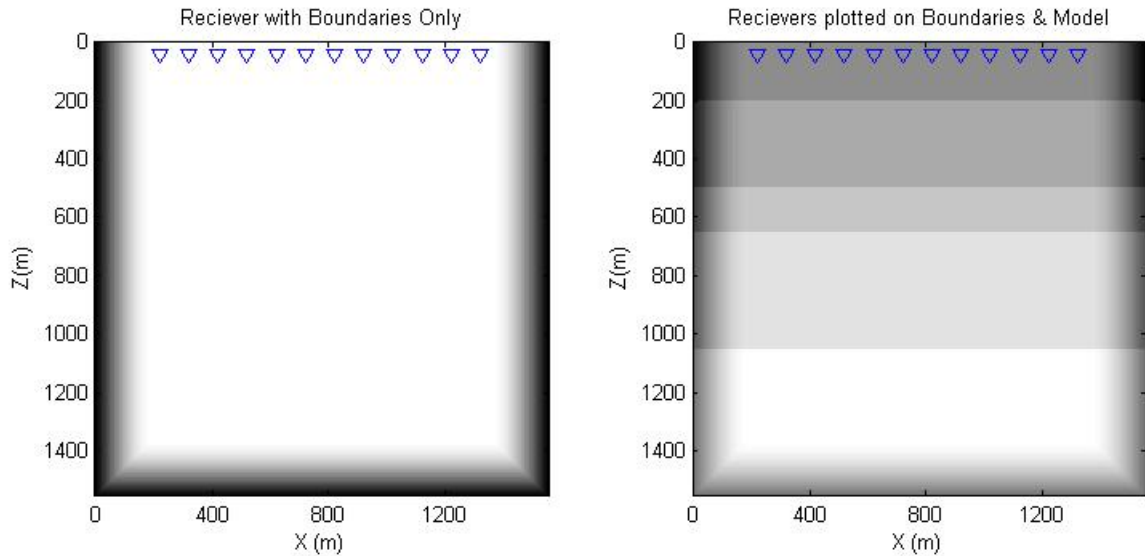
Output of command :

Figure 6.6: An array of receiver is plotted such that it doesn't show all sources by plotting every 5th only.

6.4.3 Plotting source and receiver in same figure

Description of command :

```
1   GEOMETRY_PLOT
2   The program to plot the source and reciever with and without model.
3   Complete Syntax:
4        geometry_plot('FIGNO',value,'HOP_S',value,'HOP_R',value)
5   Description of parameters:
6            FIGNO  : Plot in figure number given by.
7            HOP_S  : No of receiver to be skipped
8            HOP_R  : No of receiver to be skipped
9   Note:
10  1) Don't place source very close to surface.
11  2) Give all the position after adding/subtracting the ABC layer thickness according to side
        where they are added.
12  3) All parameters are mandatory.
13      In case any parameter is not provided the program will try to use the values stored in
            input folder.
14      If it cannot find any stored value then it shows error.
15  Example:
16      geometry_plot('FIGNO',3,'HOP_S',1,'HOP_R',5);
17      geometry_plot()
```

Demonstration/Example :

```
1  >> nvpair_geometry_src_st_line_surf('WFP',wf_path,'DEPTH',10,'FIRST',55,'LAST',330,'DIFF',5,'
      PlotON','y');
2  >> nvpair_geometry_rec_st_line_vsp('HLOC',250,'FIRST',5,'LAST',350,'DIFF',10,'PlotON','y');
3  >> nvpair_geometry_plot('FIGNO',3, 'HOP_S',5, 'HOP_R',4)
```
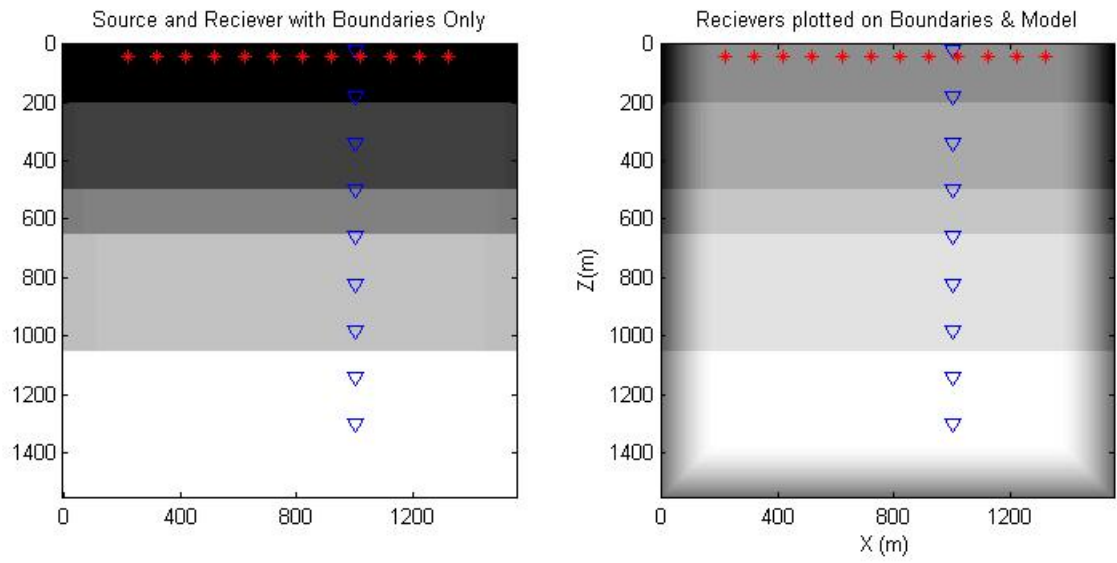
33

Output of command :

Figure 6.7: An array of receiver is plotted such that it doen't show all sources by plotting every 5th only.

Chapter 7

Other functions

7.1 Analaysis

This is function is essential to find out the stability of the current simulation for the given parameters. The stability is checked using the CFL conditions [1].

The dispersion condition is also analysed by this code. This condition is important to keep dispersion below 5% level, however, it is not necessary for simulation to run [2].

This can be simply checked by running following function

```
>> analyse_elastic('WFP',wf_path)
```

If everything is ok it will produce following type of output:

```
1 FUNC: Analysis for Elastic wave
2      Time step size,      dt = 0.0003       (Default, stored)
3      Source Frequency,    f0 = 15        (Default, stored)
4      Grid spacing Taken,  dx = 3        (Default, stored)
5      Total time duration, dz = 3        (Default, stored)
6      Vp model supplied      (Default, stored)
7      Vs model supplied      (Default, stored)
8            Analysis Results:
9      Given time step: ok;     "dt" Present value( 0.000300) < Required( 0.000387).
10     Given grid spacing: ok;    "dh" Present value( 3.000000) < Required( <3.624569).
11     CFL and Grid size requirement analysis done.
```

In case time step is higher then a *error* message will come up as following

```
1 FUNC: Analysis for Elastic wave
2      Time step size,      dt = 0.0008       (Default, stored)
3      Source Frequency,    f0 = 15         (Default, stored)
4      Grid spacing Taken,  dx = 5        (Default, stored)
5      Total time duration, dz = 5        (Default, stored)
6      Vp model supplied      (Default, stored)
7      Vs model supplied      (Default, stored)
8            Analysis Results:
9 Error using nvpair_analyse_elastic (line 61)
10     Revise time step...!!!   "dt" Present value( 0.000800) > Required( 0.000645).
```

In case of grid spacing is not satisfying the dispersion condition then a *warning* message will come up as following

```
1 FUNC: Analysis for Elastic wave
2      Time step size,      dt = 0.0003       (Default, stored)
3      Source Frequency,    f0 = 15         (Default, stored)
4      Grid spacing Taken,  dx = 5        (Default, stored)
5      Total time duration, dz = 5        (Default, stored)
6      Vp model supplied      (Default, stored)
```

---

[1] Courant, R.; Friedrichs, K.; Lewy, H. (March 1967) [1928], "On the partial difference equations of mathematical physics", IBM Journal of Research and Development, 11 (2): 215–234,

[2] Levander, A., 1988, Fourth-order finite-difference P-SV seismograms, Geophysics, 53, 1425-1436.

```
  7         Vs model supplied      (Default, stored)
  8              Analysis Results:
  9         Given time step: ok;     "dt" Present value( 0.000300) < Required( 0.000645).
 10   Warning:         Revise grid spacing...!!!!  "dh" Present value( 5.000000) > Required( <3.624569)
            .
 11
 12   > In nvpair_analyse_elastic at 78
 13         CFL and Grid size requirement analysis done.
```

7.2   Derive Material parameters

The material parameters has to calculated according to the grid position of the material parameters [3].
It is an essential step hence this function has to be called before running the simulation.

   The parameters for staggered grid can be derived using following function

```
>> model_derived_elastic_g1('WFP',wf_path)
```

   It shows following output

```
FUNC: Deriving Elastic Parameter for grid type g1
      Vp model supplied        (Default, stored)
      Vs model supplied        (Default, stored)
      Density model supplied        (Default, stored)
      Derived parameters generated successfully
```

   Which shows the parameter generation for the staggered nodes was successful.

7.3   Simulation

The last part of it is to carry out the simulation which can be done using following function.

Complete syntax of code   :

```
  1   CALCULATION_ELASTIC
  2   The solution of wave equation is done here using follownig grid.
  3   Complete Syntax:
  4         calculation_elastic_g1('SRC_I',value, 'DN_W',value, 'DN_SS',value, 'PlotON',value)
  5   Description of parameters:
  6         SRC_I   : Source No for which the simulation will be carried out.
  7         DN_W    : No time steps of wavefield to skip
  8         DN_SS   : No time steps of synthetic seismogram to skip
  9         PlotON  : Show wave propagation while simulation
 10         DN_P    : No of time steps to skip plotting.
 11   Example:
 12         calculation_elastic_g1('SRC_I',1, 'DN_W',1, 'dN_SS',1, 'PlotON','y','dN_P',10)
 13         calculation_elastic_g1('SRC_I',1, 'DN_W',10000, 'dN_SS',4, 'PlotON','y')
 14         calculation_elastic_g1()
 15   Note:
 16   1) It is advised to keep dN_W to very large value if you don't want to save entire wavefield
 17   2) Plotting itself takes much time so it is advisable to skip few steps in case of very fine
         steps.
 18   3) All other parameters are taken from the input folder.
 19   4) The grid arrangement used in calculation is given following
 20   %%%%%%%%%%%%%%%%%%%% Grid arrangement%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 21                 txx,tzz----------vx---------txx,tzz------>
 22             lbd,mu |        bh          |
 23                    |         .          |
 24                    |         .          |
```

---

[3] Ajay Malkoti*, Nimisha Vedanti, Praveen Kunagu, and R.K. Tiwari (2015) Modeling viscoelastic seismic wave propagation in Deccan flood basalt, western India. SEG Technical Program Expanded Abstracts 2015: pp. 3764-3768.doi: 10.1190/segam2015-5898555.1

```
25                      |          .             |
26               vz  |..........txz         |
27               bv  |          muvh         |
28                   |                       |
29                   |                       |
30                   |                       |
31         txx,tzz|-----------------------|
32                   |
33                  \|/
```

Demonstration/Example :

If an array of source is provides and we want to carry out simulation with following considerations:

1. Carry out simulation only for given source (say no 5).

2. Do not save wave field (set dN_W to very high value).

3. Skip a few samples in synthetic seismogram to save space (set dN_SS to some value).

4. Visualize the wave propagation (set plotON to 'y')

```
1  >> nvpair_calculation_elastic_g1('src_i',5,'dN_W',100000,'dN_SS',4,'plotON','y');
```

Plotting of Results :

```
CALCULATION_PLOT
   This function is used for plotting the synthetic seimogram with scaling
   and clipping the amplitudes.
   Complete syntax:
       calculation_plot('Wave_Type',option,'ShotGather',matrix,'FigNo',value,...
           'ShotNo',value,'clip',value,'Scale')
   Description of Parameters
       ShotGather  :  The synthetic seimogram (Optional)
       Wave_Type   :  'Acoustic', 'Elastic', etc.
       FigNo       :  Figure No on which you want to plot
       ShotNo      :  Source No, if there are more than one (e.g. for an array)
       Clip        :  To remove very high amplitudes
       Scale       :  To enhance deeper/later reflections.
   Example
       calculation_plot('wave_type','elastic','shotno',5)
       calculation_plot('wave_type','elastic','shotno',5, 'clip',.95,'scale',6)
```

Chapter 8

Acknowledgemets

The MARMOUSI model used for the synthetic modelling can be downloaded from UoH website [1]
The SEGYMAT MATLAB package [2] was used in this pack to read the segy file.

---

[1] http://www.agl.uh.edu/downloads/downloads.htm
[2] http://segymat.sourceforge.net/