Manual for

FDwave3D

A Matlab solver for the **3D** anisotropic **wave** equation using the finite-difference (**FD**) method

October 9, 2020

Contents

List of contributors

Lei Li, Ajay Malkoti, Ivan Abakumov

Characteristics of the code

- In- time domain
- For- anisotropic elastic media and moment tensor sources
- Over- staggered grid
- Uses- vectorized finite-difference operator
- With- Matlab

References

Malkoti, A., Vedanti, N., Tiwari, R.K.: An algorithm for fast elastic wave simulation using a vectorized finite difference operator. Computers & Geosciences. 116, 23–31 (2018). `https://doi.org/10.1016/j.cageo.2018.04.002`
Li, L., Tan, J., Zhang, D., Malkoti, A., Abakumov, I., Xie, Y.: FDwave3D: A MATLAB solver for the 3D anisotropic wave equation using the finite-difference method. Submitted Manuscript (the current study).

Contact

Lei Li: leileely@126.com
Ajay Malkoti: ajmalkoti@gmail.com

Copyright

Chapter 1

FDwave3D package contents

The directory structure and files are as follows:

```
FDwave3D
|   compare_cost.m
|   compare_homo.m
|   showrecord_layer.m
|   showrecord_overthrust.m
|   showwavefield_layer.m
|   showwavefield_overthrust.m
|   show_overthrust.m
|   Simulation3_homo_downhole.m
|   Simulation3_homo_FulAni.m
|   Simulation3_layer3_TI.m
|   Simulation3_overthrust_TI.m
|   LICENSE.txt
|   manual_FDwave3D.pdf
|
+--- FDwave
|   |
|   +--- Analyse
|   |       FDwave_analyse_3Delastic.m
|   |
|   +--- BC
|   |       bc_3Ddamp.m
|   |       bc_3Dpml.m
|   |       FDwave_bc_3Dselect.m
|   |
|   +--- Calculate
|   |       DiffCoef.m
|   |       FDwave_calculation_3Delastic_2N_vTI.m
|   |       FDwave_calculation_3Delastic_2N_vFulAni.m
|   |       FDwave_calculation_3Dimage_plot.m
|   |       FDwave_calculation_3Dwiggle_plot.m
|   |
|   +--- Geometry
|   |       FDwave_3Dgeometry_rec_downhole.m
|   |       FDwave_3Dgeometry_rec_general_surface.m
|   |       FDwave_3Dgeometry_rec_st_line_surf.m
```

```
|   |       FDwave_3Dgeometry_src_single.m
|   |       FDwave_geometry_plot.m
|   |       geometry_3Dplot_rec.m
|   |       geometry_3Dplot_recxz.m
|   |       geometry_3Dplot_src.m
|   |       geometry_3Dplot_srcxz.m
|   |
|   +--- Model
|   |       FDwave_model_3Dload.m
|   |       FDwave_model_derived_3Delastic_2N.m
|   |       FDwave_model_n_3Dlayers_TI.m
|   |       FDwave_model_n_3Dlayers_FulAni.m
|   |       model_2Dplot.m
|   |       model_3Dplot.m
|   |       plotmat2.m
|   |       thomsen_to_c.m
|   |
|   +--- Others
|   |       wiggle.m
|   |       slicen.m
|   |
|   +--- Source
|           FDwave_source_ricker.m
|           FPgenMT.m
|           source_plot.m
|
À--- C-MEX
|       FD_TC_cpp.cpp
|       FD_TC_cpp.mexw64
|       FD_TC_cpp.mexa64
|       get_ASOFI_signal.m
|       get_ricker.m
|       GridClass.m
|       Simulation3_homo_downhole_dc1.m
```

`manual_FDwave3D.pdf: (this file)`
Instruction and documentation of the code package.

`FDwave`:
This directory contains all the programs & functions related to seismic modeling.
Please note that no additional libraries are required for the code package.

`C-MEX`:
The MEX function is provided as a more efficient option in the context of Matlab.

SEG/EAGE overthrust model
`https://wiki.seg.org/wiki/SEG/EAGE_Salt_and_Overthrust_Models`
recommend to download the processed model via the link below (about 140 MB):
`https://drive.google.com/file/d/1vTemFS0poXAUMhHfea-nVGRSvuLPNa5K/view?usp=sharing`

The steps for reproducing the records and figures in the related paper are also introduced.
PLEASE NOTE: The processing time for reproducing the snapshots and seismograms varies from several hours to tens of hours on a normal standalone computer. All the results can be directly downloaded via the link below (about 150 MB) for a quick validation.
`https://drive.google.com/file/d/1mCKpKfma-oWOW9pfu3bvknVzmX1hsEFk/view?usp=sharing`

Chapter 2

Theory and method

Elastic wave modeling using finite-difference method

The basic equations for elastodynamic wave motion include the momentum conservation equation (equation of motion) which is also the governing equation, and the constitutive law (Hooke's law) which specifies the relation between the stress and strain tensors. These two equations for generally anisotropic elastic media can be written as

$$\rho \frac{\partial v_i}{\partial t} = \frac{\partial \tau_{ij}}{\partial x_j} + \rho f,$$

$$\tau_{ij} = C_{ijkl} \varepsilon_{kl}.$$

where ρ is the density of the medium, v_i is the particle velocity, τ_{ij}, ε_{kl} are the stresses and strains, f_i is the external body force in the direction i, C_{ijkl} denotes the tensor of elastic constants, and $i, j, k, l \in x, y, z$. The fourth-order elastic tensor has only 21 independent parameters in a generally anisotropic medium due to the symmetry, and an isotropic medium has only 2 independent elastic parameters, i.e. Lame parameters λ and μ.

Source implementation

A point source is generally utilized for the simulation of small local or near-regional earthquakes, where the dominant wavelength is much larger than the source dimensions (Jost and Herrmann, 1989). A point displacement discontinuity (e.g., dislocation) is assumed at the source, which can be mathematically represented with the body-force term in the equation of motion for the continuous medium (Burridge and Knopoff, 1964).

A moment tensor is a mathematical representation of a seismic source, given by nine pairs of force couples that act at a point and expressed as a symmetric second-order tensor (Gilbert, 1971). The moment-tensor source can be implemented by loading the equivalent body-force term on particle velocity components (e.g., Graves, 1996) or stress components (e.g., Pitarka, 1999). Besides, due to the relationship between moment tensor and the fault plane solution, arbitrary fault planes solutions consist of (strike, slip, rake) are also applicable in our package.

Regarding the source-time function, the commonly-used Ricker wavelet is adopted. Anyone who wants to use alternative source-time functions, just add them into the **Source** folder and replace the current function `FDwave_source_ricker` with the preferred ones in the scripts .

Moment tensor

Examples:

```
MT=[1 0 0; 0 1 0; 0 0 1]; %the ISO source
MT=sqrt(1/2)*[0 1 0; 1 0 0; 0 0 0]; %a common strike-slip (DC) source
MT=sqrt(1/2)*[0 0 0; 0 0 -1; 0 -1 0]; %a common dip-slip (DC) source

% one node scheme
% txx, tyy, tzz, txy, txz, tyz are stress components, src(t) is the
    source wavelet
txx_x(srcnz,srcnx,srcny)=txx_x(srcnz,srcnx,srcny)+(-MT(1,1)/3)*src(t);
txx_y(srcnz,srcnx,srcny)=txx_y(srcnz,srcnx,srcny)+(-MT(1,1)/3)*src(t);
txx_z(srcnz,srcnx,srcny)=txx_z(srcnz,srcnx,srcny)+(-MT(1,1)/3)*src(t);

tyy_x(srcnz,srcnx,srcny)=tyy_x(srcnz,srcnx,srcny)+(-MT(2,2)/3)*src(t);
tyy_y(srcnz,srcnx,srcny)=tyy_y(srcnz,srcnx,srcny)+(-MT(2,2)/3)*src(t);
tyy_z(srcnz,srcnx,srcny)=tyy_z(srcnz,srcnx,srcny)+(-MT(2,2)/3)*src(t);

tzz_x(srcnz,srcnx,srcny)=tzz_x(srcnz,srcnx,srcny)+(-MT(3,3)/3)*src(t);
tzz_y(srcnz,srcnx,srcny)=tzz_y(srcnz,srcnx,srcny)+(-MT(3,3)/3)*src(t);
tzz_z(srcnz,srcnx,srcny)=tzz_z(srcnz,srcnx,srcny)+(-MT(3,3)/3)*src(t);

txy_x(srcnz,srcnx,srcny)=txy_x(srcnz,srcnx,srcny)+(-MT(1,2)/2)*src(t);
txy_y(srcnz,srcnx,srcny)=txy_y(srcnz,srcnx,srcny)+(-MT(1,2)/2)*src(t);

txz_x(srcnz,srcnx,srcny)=txz_x(srcnz,srcnx,srcny)+(-MT(1,3)/2)*src(t);
txz_z(srcnz,srcnx,srcny)=txz_z(srcnz,srcnx,srcny)+(-MT(1,3)/2)*src(t);

tyz_y(srcnz,srcnx,srcny)=tyz_y(srcnz,srcnx,srcny)+(-MT(2,3)/2)*src(t);
tyz_z(srcnz,srcnx,srcny)=tyz_z(srcnz,srcnx,srcny)+(-MT(2,3)/2)*src(t);


txx=txx_x+txx_y+txx_z;
tyy=tyy_x+tyy_y+tyy_z;
tzz=tzz_x+tzz_y+tzz_z;
txy=txy_x+txy_y;
txz=txz_x+txz_z;
tyz=tyz_y+tyz_z;
```

Fault plane solution

Example: a dip-slip (DC) source. Input angles are in degrees

```
strike =30;
dip = 70;
rake = 90;
gamma = 0; %tensile angle
sigma = 0.25; %Poisson's ratio
MT=FPgenMT(strike, dip, rake, gamma, sigma)
```

Boundary condition

The code package adopts the stress imaging technique (Levander, 1988; Graves, 1996) to implement the free surface condition, and offers two absorbing boundary conditions, i.e., the damping boundary condition and PML. The damping boundary condition is also called as classical sponge method, in which the amplitudes are multiplied by a damping factor to eliminate spurious wave reflection at the boundaries (e.g., Cerjan et al., 1985). Alternatively, the classical (split) PML is used. The wavefield is first split into three orthogonal directions and then the PML is applied to the respective field.

C-MEX programming

C-MEX programming involves writing fast implementation in C/C++ and compile then into MEX functions, which can be called in MATLAB just like normal MATLAB scripts or functions. The code package include a MEX function `FD_TC_cpp`, along with the test on the homogeneous model for reference. Please note that the related C++ code is not fully optimized (e.g., vectorized or parallelized), which is beyond the scope of the current study.

References

Burridge, R., and L. Knopoff, 1964, Body force equivalents for seismic dislocations: Bulletin of the Seismological Society of America, 54, 1875–1888.
Cerjan, C., D. Kosloff, R. Kosloff, and M. Reshef, 1985, A nonreflecting boundary condition for discrete acoustic and elastic wave equations: GEOPHYSICS, 50, 705–708.
Gilbert, F., 1971, Excitation of the normal modes of the Earth by earthquake sources: Geophysical Journal International, 22, 223–226.
Graves, R. W., 1996, Simulating seismic wave propagation in 3D elastic media using staggered-grid finite differences: Bulletin of the Seismological Society of America, 86, 1091–1106.
Jost, M. L., and R. B. Herrmann, 1989, A student's guide to and review of moment tensors: Seismological Research Letters, 60, 37–57.
Levander, A. R., 1988, Fourth-order finite-difference P-SV seismograms: GEOPHYSICS, 53, 1425–1436.
Pitarka, A., 1999, 3D elastic finite-difference modeling of seismic motion using staggered grids with nonuniform spacing: Bulletin of the Seismological Society of America, 89, 54–68.

Chapter 3

Running the code

Steps for running the code

FDwave3D requires 8 basic steps to conduct seismic modeling as follows:

1. Setup the FDwave3D code. This step mostly remains unaltered since the scripts will load the whole packake into the work path automatically.

2. Define the model by assigning stiffness coefficients for the media C_{ij}. Users can either generate simple layered models with the functions (`FDwave_model_n_3Dlayers_TI`) provided in the package, or load exsiting complex models from outside with the function `FDwave_model_3Dload`. The code package provides two ways to define an anisotropic model, by setting the stiffness coefficients directly, or transforming the Thomsen parameters to the stiffness coefficients (with `thomsen_to_c`). There are two options for setting the TI models with stiffness coefficients. Option I: define the stiffness coefficients of VTI or HTI medium directly; Option II: define the HTI model by rotating the VTI anticlockwize pi/2 along Y-axis.

3. Define the source wavelet signature and the source mechanism. The Ricker wavelet is used in the current package. Users have to add new wavelet functions if they prefer to use other source-time functions.

4. Analyze/check if the parameters are assigned correctly or not. Check the potential numerical stability and grid dispersion issues with the function `FDwave_analyse_3Delastic`. When the parameters and models are not defined correctly, e.g., the numerical stability condition is not satisfied), adjust the corresponding parameters based on the displayed warnings.

5. Adjust properties for the simulation. Set and adjust other parameters for the simulation, e.g., the total number of time steps and the results need to be displayed or stored.

6. Select the boundary conditions, e.g. Free surface / absorbing. The absorbing boundary condition (ABC) and perfectly matched layer (PML) condition are provided, and the PML condition is reccomended.

7. Select source and receiver geometry. The current package supports single source simulation, and both regular (e.g., a receiver line) and general (e.g., scatter surface monitoring array) receiver geometries.

8. Carry out the simulation.

Getting help

Details about a function in the FDwave folder can be obtained by typing "`help fun_name`" in the command window.More details on the vectorized FD operator can be found in above references and the manual file.

Reproduce the figures in the accompanying manuscript

Figure 2: Comparison with Analytical Solution using Isotropic-Homogeneous model

`Simulation3_homo_downhole_dc1.m`: for the 3D homogeneous and isotropic model
`compare_homo.m`: for comparison between numerical and anlystic solutions

1. run `Simulation3_homo_downhole_dc1.m` to generate the results of the two double-couple sources.

2. run `compare_homo.m` to generate Figure 2.

Script 3.1: `Simulation3_homo_downhole.m`

```
% Script for generating the synthetic waveforms for isotropic and
    homogeneous model
% This part will setup all the FDwave program
clc; close all; clear all;
code_path=['.' filesep, 'FDwave'];              % Path of FD code files
addpath(genpath(code_path));            % Add the code folder to the
    current command space
wf_path=pwd;                    % where you want to store your data


%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preprocessing  %%%%%%%%%%%%%%%%%%%%%%%%
% create/modify model
T=200;
vp=3000;
vs=vp/1.67;
rho=2500;

eps=[0];        % TI parameters (for isotropic media, equal to zero)
gamma=[0];      % TI parameters (for isotropic media, equal to zero)
delta=[0];      % TI parameters (for isotropic media, equal to zero)

for i=1:length(vp)
    thomsen_parameters=[vp(i),vs(i),eps(i),gamma(i),delta(i)];
    cc = thomsen_to_c(thomsen_parameters);
    cc11(i)=cc(1)*rho(i);
    cc33(i)=cc(2)*rho(i);
    cc44(i)=cc(3)*rho(i);
    cc66(i)=cc(4)*rho(i);
    cc13(i)=cc(5)*rho(i);
end

% Option I: define the TI model directly
cc22=cc11;cc55=cc44;cc23=cc13;cc12=cc11-2*cc66;    %% VTI
%cc22=cc33;cc12=cc13;cc23=cc22-2*cc44;cc55=cc66;   %% HTI
```

```
CC=[cc11' cc12' cc13' cc22' cc23' cc33' cc44' cc55' cc66'];     % VTI &
    HTI, given by Thomsen parameters

% Option II: define the HTI model by rotating the VTI anticlockwize (Y)
    pi/2
%CC=[cc33' cc13' cc13' cc11' cc12' cc11' cc66' cc55' cc55'];    % HTI
    rotated by VTI anticlockwize (Y) pi/2

nab=20;                          % Number of nodes for absorbing
    boundaries
M=10;                            % the order of FD operator M=2N

FDwave_model_n_3Dlayers_TI('WFP',wf_path,'WAVE_TYPE','Elastic','DX',2.5,
    'DY',2.5,...
        'DZ',2.5,'THICKNESS',T,'XZ_RATIO',1,'NAB',nab,'Vp',vp,'VS',vs,'
            RHO',rho,...
        'C',CC,'PlotON','y3d','verbose','y')

% source wavelet
FDwave_source_ricker('WFP',wf_path,'T',0.24,'DT',.0003,'F0',60,'T0'
    ,0.03,'PlotON','y','verbose','y');

% analyze/check the parameters
FDwave_analyse_3Delastic('WFP',wf_path,'2N',M,'verbose','y')

% derived model
FDwave_model_derived_3Delastic_2N('WFP',wf_path,'verbose','y')

% boundary conditions
FDwave_bc_3Dselect('WFP',wf_path,'BCNAME','PML','BCTYPE','topABC','NAB',
    nab,'PlotON','y3d','verbose','y') ;

% source and receiver geometry
load([wf_path,[filesep,'model']],'dx','dy','dz','nx','ny','nz');
FDwave_3Dgeometry_src_single('WFP',wf_path,'SX',round(nx/2),'SY',round(
    ny/2),'SZ',nz-nab,'PlotON','y3d','verbose','y');

FDwave_3Dgeometry_rec_downhole('XLOC',1+nab,'YLOC',1+nab,'FIRST',1+nab,'
    LAST',nz-nab,'DIFF',1,'PlotON','y3d')

for testno=1:2
% source implementation
if testno==1
    MT0=sqrt(1/2)*[0 1 0;1 0 0;0 0 0];%DC1
    outputname='homo_iso_dc1_2n';
else
    MT0=sqrt(1/2)*[0 0 0;0 0 -1;0 -1 0];%DC2
    outputname='homo_iso_dc2_2n';
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Simulation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Do the time stepping calculations of wavefield
tic;
FDwave_calculation_3Delastic_2N_vTI('src_i',1,'MT',MT0,'wfp',pwd,'dN_W'
    ,1000,'DN_SS',1,'DN_P',10,'2N',M,'plotON','y','verbose','n');

prot=toc;
%%%%%%%%%%%%%%%%%%%%%%%%%%%% Post-processing  %%%%%%%%%%%%%%%%%%%%%%%
```

```
80 str='SS_1.mat';          % this can be changed by user if required
81
82 %plotting the seismogram in image form
83 FDwave_calculation_3Dimage_plot('wfp',wf_path,'SSFileName',str)
84
85 %plotting the seismogram in wiggle traces form
86 FDwave_calculation_3Dwiggle_plot('wfp',wf_path,'SSFileName',str,'scale'
       ,3)
87
88 load (str);
89 save(outputname,'Sx','Sy','Sz','dN_SS','dt','N','M','dx','dy','dz','nx',
       'ny','nz','prot','-v7.3');
90
91 end
```

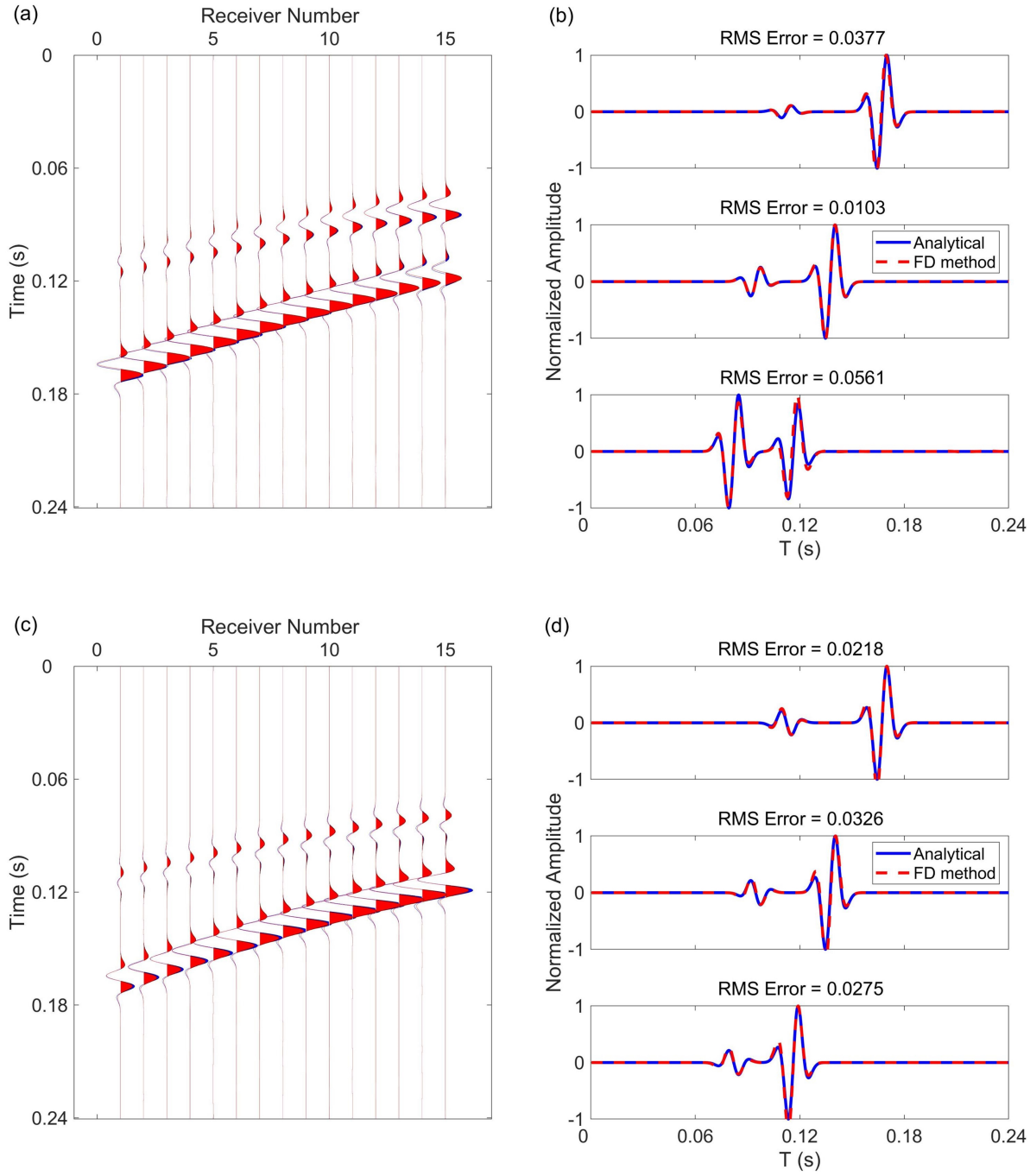

Figure 3.1: Comparison of synthetic seismograms with analytical solutions of two double-couple sources. (a) and (b) are results of a strike-slip source, (c) and (d) are results of a dip-slip source.

Figure 3: Synthetic Wavefields and Seismograms of the Layerd Model

Simulation3_layer3_TI.m: for the layered anisotropic model
showwavefield_layer.m: show original wavefields of the layered model (Figures 3a-c)
showrecord_layer.m: show original seismograms of the layered model (Figures 3d-f)

1. run `Simulation3_layer3_TI.m` to generate the wavefields and records of the isotropic, VTI, and HTI layered models.

2. run `showwavefield_layer.m` and `showrecord_layer.m` to generate Figures 3a-c and 3d-f.

Script 3.2: `Simulation3_layer3_TI.m`

```
% This code is for generating the synthetic waveforms for 3 layers (anisotropic) case
% This part will setup all the FDwave program
clc; close all; clear all;
code_path=['.' filesep, 'FDwave'];              % Path of FD code files
addpath(genpath(code_path));            % Add the code folder to the current command space
wf_path=pwd;                    % where you want to store your data


%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preprocessing  %%%%%%%%%%%%%%%%%%%%%%
% create/ modify model
scale=1; % scale size of the small model
T=[750,1000,750]/scale;
vp=[3724,4640,5854];%m/s
vs=[1944,2583,3251];
rho=[2450,2490,2680];%kg/m3


for testno=1:3

%% ISO model
if testno==1
% TI parameters (for isotropic media, the three parameters are equal to zero)
eps=[0,0,0];%
gamma=[0,0,0];%
delta=[0,0,0];%
for i=1:length(vp)
        thomsen_parameters=[vp(i),vs(i),eps(i),gamma(i),delta(i)];
        cc = thomsen_to_c(thomsen_parameters);
        cc11(i)=cc(1)*rho(i);
        cc33(i)=cc(2)*rho(i);
        cc44(i)=cc(3)*rho(i);
        cc66(i)=cc(4)*rho(i);
        cc13(i)=cc(5)*rho(i);
end
% Option I: define the TI model directly
cc22=cc11;cc55=cc44;cc23=cc13;cc12=cc11-2*cc66;    %% VTI
%cc22=cc33;cc12=cc13;cc23=cc22-2*cc44;cc55=cc66;   %% HTI
CC=[cc11' cc12' cc13' cc22' cc23' cc33' cc44' cc55' cc66'];     % VTI & HTI, given by Thomsen parameters
record_out='layer3_iso_dc1';
wavefield_out='wavefield_layer3_iso';
```

```
41
42 %% VTI model
43 else if testno==2
44 % TI parameters
45 eps=[0,0.334,0];%
46 gamma=[0,0.575,0];%
47 delta=[0,0.75,0];%
48 for i=1:length(vp)
49         thomsen_parameters=[vp(i),vs(i),eps(i),gamma(i),delta(i)];
50         cc = thomsen_to_c(thomsen_parameters);
51         cc11(i)=cc(1)*rho(i);
52         cc33(i)=cc(2)*rho(i);
53         cc44(i)=cc(3)*rho(i);
54         cc66(i)=cc(4)*rho(i);
55         cc13(i)=cc(5)*rho(i);
56 end
57
58 % Option I: define the TI model directly
59 cc22=cc11;cc55=cc44;cc23=cc13;cc12=cc11-2*cc66;    %% VTI
60 %cc22=cc33;cc12=cc13;cc23=cc22-2*cc44;cc55=cc66;   %% HTI
61 CC=[cc11' cc12' cc13' cc22' cc23' cc33' cc44' cc55' cc66'];     % VTI &
       HTI, given by Thomsen parameters
62 record_out='layer3_vti_dc1';
63 wavefield_out='wavefield_layer3_vti';
64
65 %% HTI model
66 else if testno==3
67 % TI parameters
68 eps=[0,0.334,0];%
69 gamma=[0,0.575,0];%
70 delta=[0,0.75,0];%
71 for i=1:length(vp)
72         thomsen_parameters=[vp(i),vs(i),eps(i),gamma(i),delta(i)];
73         cc = thomsen_to_c(thomsen_parameters);
74         cc11(i)=cc(1)*rho(i);
75         cc33(i)=cc(2)*rho(i);
76         cc44(i)=cc(3)*rho(i);
77         cc66(i)=cc(4)*rho(i);
78         cc13(i)=cc(5)*rho(i);
79 end
80
81 % Option I: define the TI model directly
82 cc22=cc11;cc55=cc44;cc23=cc13;cc12=cc11-2*cc66;    %% VTI
83 %cc22=cc33;cc12=cc13;cc23=cc22-2*cc44;cc55=cc66;   %% HTI
84 %CC=[cc11' cc12' cc13' cc22' cc23' cc33' cc44' cc55' cc66'];     % VTI &
       HTI, given by Thomsen parameters
85
86 % Option II: define the HTI model by rotating the VTI anticlockwize (Y)
       pi/2
87 CC=[cc33' cc13' cc13' cc11' cc12' cc11' cc66' cc55' cc55'];    % HTI
       rotated by VTI anticlockwize (Y) pi/2
88 record_out='layer3_hti_dc1';
89 wavefield_out='wavefield_layer3_hti';
90     end
91     end
92 end
93
94
```

```
nab=20;
M=10;              % the order of FD operator M=2N

FDwave_model_n_3Dlayers_TI('WFP',wf_path,'WAVE_TYPE','Elastic','DX',10,'DY',10,...
        'DZ',10,'THICKNESS',T,'XZ_RATIO',1.2,'NAB',nab,'Vp',vp,'VS',vs,'RHO',rho,...
        'C',CC,'PlotON','y3d','verbose','y')

% source wavelet
FDwave_source_ricker('WFP',wf_path,'T',1.2,'DT',.0005,'F0',30,'T0',0.03,'PlotON','y','verbose','y');

% analyze/check the parameters
FDwave_analyse_3Delastic('WFP',wf_path,'2N',M,'verbose','y')

% derived model
FDwave_model_derived_3Delastic_2N('WFP',wf_path,'verbose','y')

% boundary conditions
FDwave_bc_3Dselect('WFP',wf_path,'BCNAME','PML','BCTYPE','topABC','NAB',nab,'PlotON','y3d','verbose','y') ;

% source and receiver geometry
load([wf_path,[filesep,'model']],'dx','dy','dz','nx','ny','nz');
FDwave_3Dgeometry_src_single('WFP',wf_path,'SX',round(nx/2),'SY',round(ny/2),'SZ',round(nz/2),'PlotON','y3d','verbose','y');

%FDwave_3Dgeometry_rec_st_line_surf('WFP',wf_path,'DEPTH',nab+1,'FIRST',nab+1,'LAST',nx-nab,'DIFF',5,'PlotON','y3d','verbose','y');
FDwave_3Dgeometry_rec_downhole('XLOC',round(nx/2)+nab,'YLOC',round(ny/2)+nab,'FIRST',1+nab,'LAST',nz-nab,'DIFF',1,'PlotON','y3d')


% source implementation
MT0=sqrt(1/2)*[0 1 0;1 0 0;0 0 0];%DC1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Simulation %%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Do the time stepping calculations of wavefield
tic;
FDwave_calculation_3Delastic_2N_vTI('src_i',1,'MT',MT0,'wfp',pwd,'dN_W',1000,'DN_SS',1,'DN_P',10,'2N',M,'plotON','y','verbose','n');

prot=toc;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Post-processing  %%%%%%%%%%%%%%%%%%%%%%%%
strrec='SS_1.mat';          % this can be changed by user if required
strwav='wavefield_1.mat';
% plotting the seismogram in image form
%FDwave_calculation_3Dimage_plot('wfp',wf_path,'SSFileName',strrec)

% plotting the seismogram in wiggle traces form
%FDwave_calculation_3Dwiggle_plot('wfp',wf_path,'SSFileName',strrec,'scale',3)

load (strrec);
load (strwav);
save(record_out,'Sx','Sy','Sz','dN_SS','dt','N','M','dx','dy','dz','nx','ny','nz','prot','-v7.3');
```

```
save(wavefield_out,'wavefield');

end
```

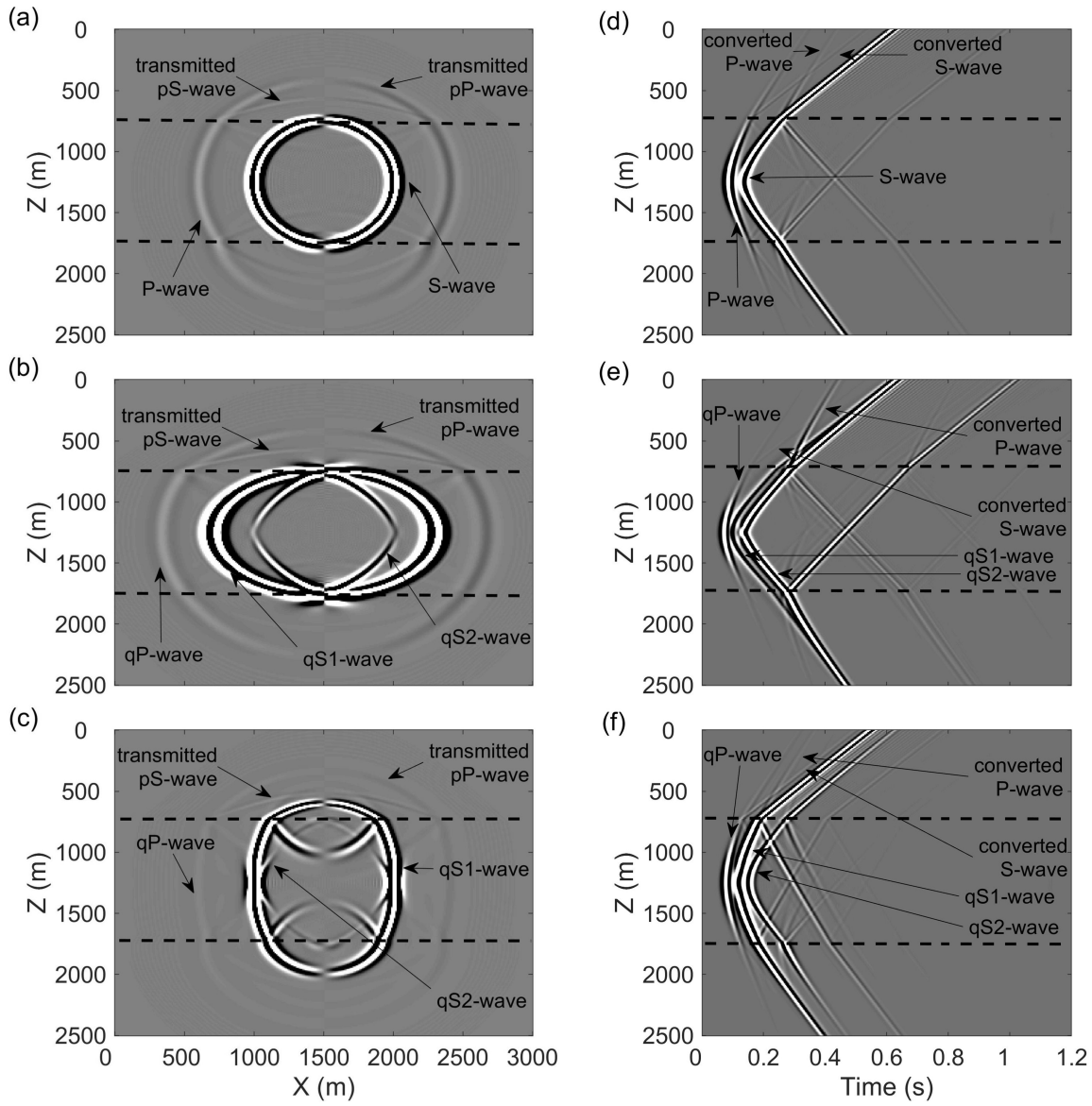

Figure 3.2: Wavefield snapshots of velocity component in Y direction for isotropic (a), VTI (b), and HTI (c) model. The slices are taken at the simulation time of 0.23 s and lateral position of y=1500 m. The corresponding seismograms of the three different models are shown in (d), (e), and (f), respectively

Figure 4: The Overthrust Model

show_overthrust.m: show the overthrsut model
run this script to produce Figure 4.

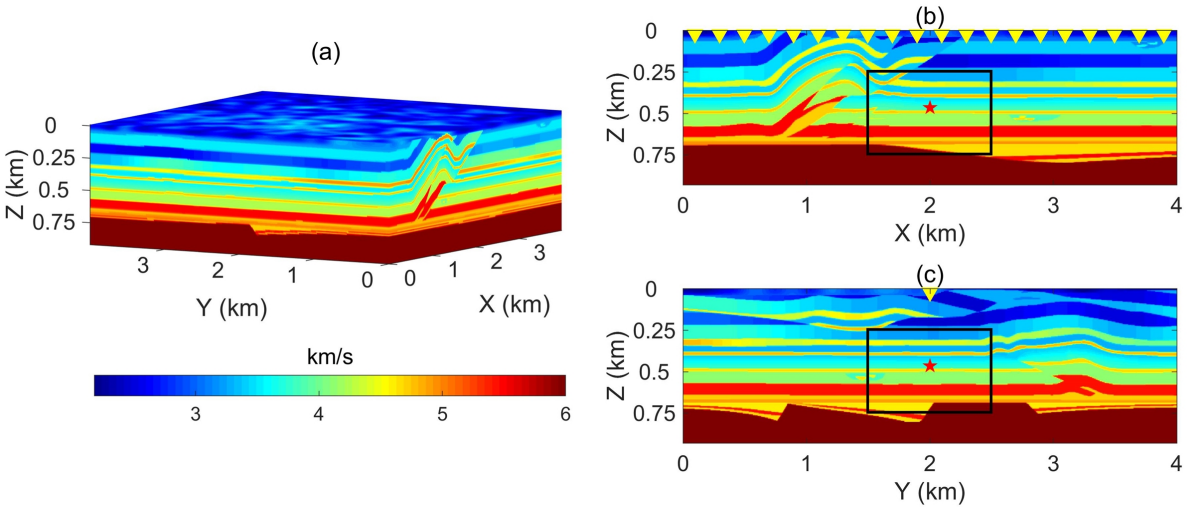

Figure 3.3: (a) P-wave velocity of the 3D overthrust model. (b) Velocity profile at 2 km in the Y direction. (c) Velocity profile at 2 km in the X direction. The red star indicates the source position, the yellow reversed triangles indicate the surface array, and the black lines show the boundary of the anisotropic region in the model.

Figure 5: Synthetic Wavefields and Seismograms of the Overthrust Model

Simulation3_overthrust_TI.m: for the anisotropic overthrust model
showwavefield_overthrust.m: show original wavefields of the overthrust model (Figures 5a-c)
showrecord_overthrust.m: show original seismograms of the overthrust model (Figures 5d-f)

Steps to produce Figure 5 is very similar to those of Figure 3:
1. run `Simulation3_loverthrust_TI.m` to generate the wavefields and records of the isotropic, VTI, and HTI layered models.

2. run `showwavefield_overthrust.m` and `showrecord_overthrust.m` to generate Figures 5a-c and 5d-f.

Script 3.3: `Simulation3_overthrust_TI.m`

```
% This code is for generating the synthetic waveforms for overthrust
    model
% This part will setup all the FDwave program
clc; close all; clear all;
code_path=['.' filesep, 'FDwave'];        % Path of FD code files
addpath(genpath(code_path));         % Add the code folder to the current
    command space
wf_path=pwd;                 % where you want to store your data


%%%%%%%%%%%%%%%%%%%%%%%%% Preprocessing  %%%%%%%%%%%%%%%%%%%%
% create/ modify model
load overthrust_3d_vp;
vp=data(1:2:end,1:2:end,1:2:end);
clear data;
vp=permute(vp,[1,3,2]);
vs=vp./1.7;
rho=1.741*(vp/1000).^(0.25)*1000;

for testno=1:3

%% ISO model
if testno==1
% TI parameters (for isotropic media, the three parameters are equal to
    zero)
eps=zeros(size(vp));%[0,0,0];%.334
gamma=zeros(size(vp));%[0,0,0];%.575
delta=zeros(size(vp));%[0,0,0];%.75


for i=1:size(vp,1)
    for j=1:size(vp,2)
        for k=1:size(vp,3)
            thomsen_parameters=[vp(i,j,k),vs(i,j,k),eps(i,j,k),gamma(i,j
                ,k),delta(i,j,k)];
            cc = thomsen_to_c(thomsen_parameters);
            cc11(i,j,k)=cc(1)*rho(i,j,k);
            cc33(i,j,k)=cc(2)*rho(i,j,k);
            cc44(i,j,k)=cc(3)*rho(i,j,k);
            cc66(i,j,k)=cc(4)*rho(i,j,k);
            cc13(i,j,k)=cc(5)*rho(i,j,k);
        end
```

```
    end
end
% Option I: define the TI model directly
cc22=cc11;cc55=cc44;cc23=cc13;cc12=cc11-2*cc66;    %% VTI
%cc22=cc33;cc12=cc13;cc23=cc22-2*cc44;cc55=cc66;   %% HTI
CC={cc11 cc12 cc13 cc22 cc23 cc33 cc44 cc55 cc66};    % VTI & HTI
record_out='overthrust_iso_dc1';
wavefield_out='wavefield_overthrust_iso';

%% VTI model
else if testno==2
% TI parameters (for isotropic media, the three parameters are equal to
    zero)
eps=zeros(size(vp));%[0,0,0];%.334
gamma=zeros(size(vp));%[0,0,0];%.575
delta=zeros(size(vp));%[0,0,0];%.75

% anisotropy surrounding the source area (for isotropic media, the three
    parameters are equal to zero)
eps(26:75,151:250,151:250)=.334;
gamma(26:75,151:250,151:250)=.575;
delta(26:75,151:250,151:250)=.75;

for i=1:size(vp,1)
    for j=1:size(vp,2)
        for k=1:size(vp,3)
            thomsen_parameters=[vp(i,j,k),vs(i,j,k),eps(i,j,k),gamma(i,j
                ,k),delta(i,j,k)];
            cc = thomsen_to_c(thomsen_parameters);
            cc11(i,j,k)=cc(1)*rho(i,j,k);
            cc33(i,j,k)=cc(2)*rho(i,j,k);
            cc44(i,j,k)=cc(3)*rho(i,j,k);
            cc66(i,j,k)=cc(4)*rho(i,j,k);
            cc13(i,j,k)=cc(5)*rho(i,j,k);
        end
    end
end
% Option I: define the TI model directly
cc22=cc11;cc55=cc44;cc23=cc13;cc12=cc11-2*cc66;    %% VTI
%cc22=cc33;cc12=cc13;cc23=cc22-2*cc44;cc55=cc66;   %% HTI
CC={cc11 cc12 cc13 cc22 cc23 cc33 cc44 cc55 cc66};    % VTI & HTI
record_out='overthrust_vti_dc1';
wavefield_out='wavefield_overthrust_vti';

%% HTI model
else if testno==3
% TI parameters (for isotropic media, the three parameters are equal to
    zero)
eps=zeros(size(vp));%[0,0,0];%.334
gamma=zeros(size(vp));%[0,0,0];%.575
delta=zeros(size(vp));%[0,0,0];%.75

% anisotropy surrounding the source area (for isotropic media, the three
    parameters are equal to zero)
eps(26:75,151:250,151:250)=.334;
gamma(26:75,151:250,151:250)=.575;
delta(26:75,151:250,151:250)=.75;

```

```
for i=1:size(vp,1)
    for j=1:size(vp,2)
        for k=1:size(vp,3)
            thomsen_parameters=[vp(i,j,k),vs(i,j,k),eps(i,j,k),gamma(i,j
                ,k),delta(i,j,k)];
            cc = thomsen_to_c(thomsen_parameters);
            cc11(i,j,k)=cc(1)*rho(i,j,k);
            cc33(i,j,k)=cc(2)*rho(i,j,k);
            cc44(i,j,k)=cc(3)*rho(i,j,k);
            cc66(i,j,k)=cc(4)*rho(i,j,k);
            cc13(i,j,k)=cc(5)*rho(i,j,k);
        end
    end
end
% Option I: define the TI model directly
cc22=cc11;cc55=cc44;cc23=cc13;cc12=cc11-2*cc66;    %% VTI
%cc22=cc33;cc12=cc13;cc23=cc22-2*cc44;cc55=cc66;   %% HTI
%CC={cc11 cc12 cc13 cc22 cc23 cc33 cc44 cc55 cc66};    % VTI & HTI
% Option II: define the HTI model by rotating the VTI anticlockwize (Y)
    pi/2
CC={cc33 cc13 cc13 cc11 cc12 cc11 cc66 cc55 cc55};   % HTI rotated by
    VTI
record_out='overthrust_hti_dc1';
wavefield_out='wavefield_overthrust_hti';
    end
    end
end

nab=20;              % Number of nodes for absorbing boundaries
M=10;                % the order of FD operator M=2N

FDwave_model_3Dload('WFP',wf_path,'WAVE_TYPE','Elastic','DX',10,'DY',10,
    'DZ',10,...
          'NAB',nab,'Vp',vp,'VS',vs,'RHO',rho,'C',CC,'PlotON','n','
             verbose','y')

% source wavelet
FDwave_source_ricker('WFP',wf_path,'T',1,'DT',.0005,'F0',30,'T0',0.03,'
    PlotON','n','verbose','y');

% analyze/check the parameters
FDwave_analyse_3Delastic('WFP',wf_path,'2N',M,'verbose','y')

% derived model
FDwave_model_derived_3Delastic_2N('WFP',wf_path,'verbose','y')

% boundary conditions
FDwave_bc_3Dselect('WFP',wf_path,'BCNAME','PML','BCTYPE','topABC','NAB',
    nab,'PlotON','n','verbose','y') ;

% source and receiver geometry
load([wf_path,[filesep,'model']],'dx','dy','dz','nx','ny','nz');
FDwave_3Dgeometry_src_single('WFP',wf_path,'SX',round(nx/2),'SY',round(
    ny/2),'SZ',round(nz/2),'PlotON','y3d','verbose','y');

FDwave_3Dgeometry_rec_st_line_surf('WFP',wf_path,'DEPTH',nab+1,'FIRST',
    nab+1,'LAST',nx-nab,'DIFF',2,'PlotON','y3d','verbose','y');
%FDwave_3Dgeometry_rec_downhole('XLOC',round(nx/2)+20,'YLOC',round(ny/2)
```

```
    +20,'FIRST',1+nab,'LAST',nz-nab,'DIFF',1,'PlotON','y3d')

% source implementation
MT0=sqrt(1/2)*[0 1 0;1 0 0;0 0 0];%DC1


%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Simulation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Do the time stepping calculations of wavefield
tic;

FDwave_calculation_3Delastic_2N_vTI('src_i',1,'MT',MT0,'wfp',pwd,'dN_W'
    ,1000,'DN_SS',1,'DN_P',10,'2N',M,'plotON','y','verbose','n');

prot=toc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Post-processing  %%%%%%%%%%%%%%%%%%%%%%%%
strrec='SS_1.mat';          % this can be changed by user if required
strwav='wavefield_1.mat';
% plotting the seismogram in image form
%FDwave_calculation_3Dimage_plot('wfp',wf_path,'SSFileName',strrec)

% plotting the seismogram in wiggle traces form
%FDwave_calculation_3Dwiggle_plot('wfp',wf_path,'SSFileName',strrec,'
    scale',3)

load (strrec);
load (strwav);
save(record_out,'Sx','Sy','Sz','dN_SS','dt','N','M','dx','dy','dz','nx',
    'ny','nz','prot','-v7.3');
save(wavefield_out,'wavefield');

end
```

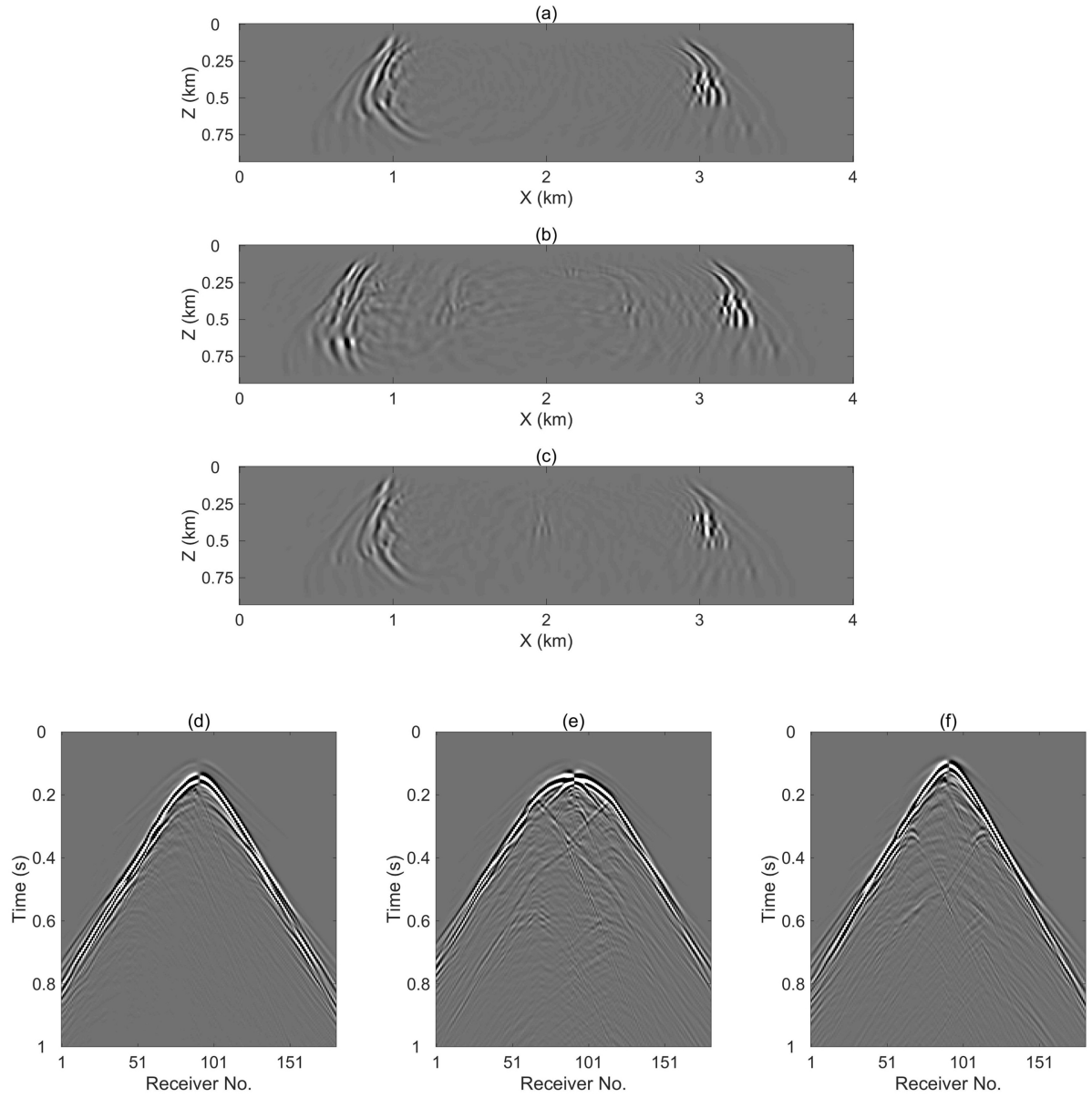

Figure 3.4: Wavefield snapshots of velocity component in Y direction for isotropic (a), VTI (b), and HTI (c) model. The slices are taken at the simulation time of 0.5 s and lateral position of y=2 km. The corresponding seismograms of the surface arrays from the three different models are shown in (d), (e), and (f).

Figure 6: Normalized Processing Time for Different Codes and Models.

compare_cost.m: show the normalized processing time
run this script to produce Figure 6.

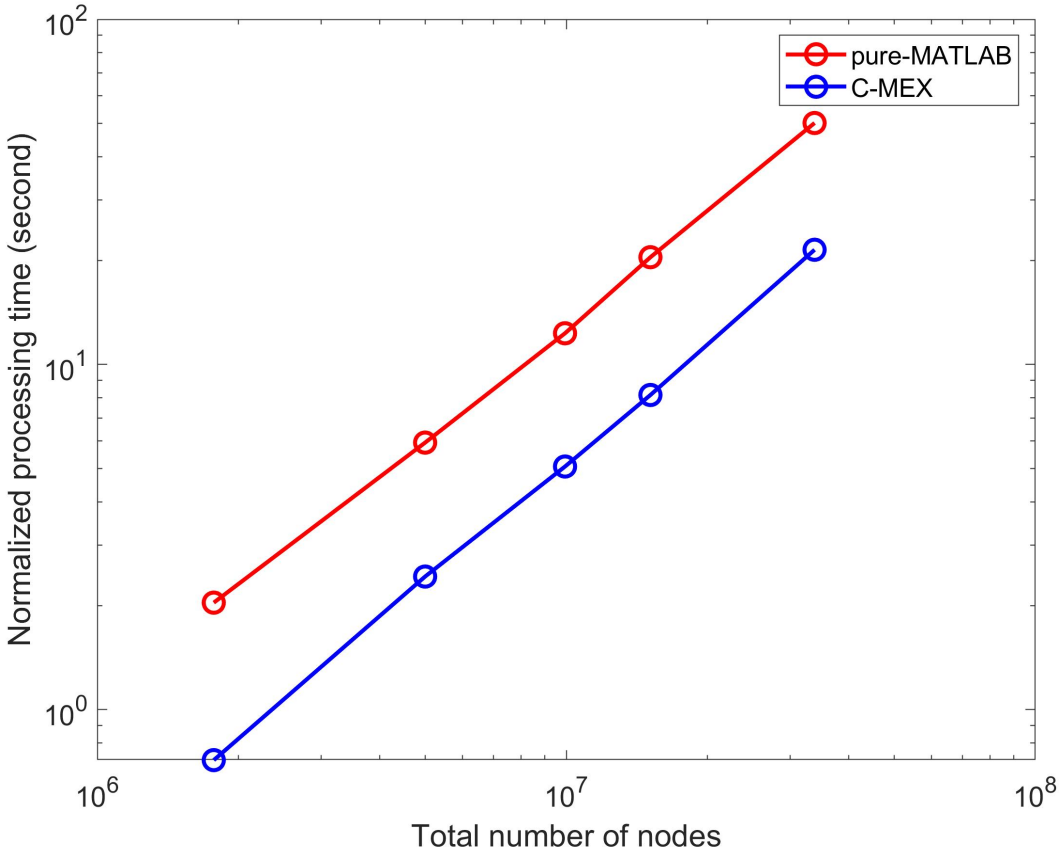

Figure 3.5: Normalized processing time for different codes and models.