Appendix A

Evaluation of Matlab code for FWI

To evaluate the result of the FWI coding and workflow, the test was made by using the true model as an initial model, (Marmousi model). The FWI workflow was then perform as normal with 19 shot points along surface with 200 active receivers, the receivers spacing is 10 meter and shot spacing is 160 meter. The result shows that the residual data has zero amplitude for all sampling, Figure A, and leads to zero magnitude when calculated a global gradient of misfit function calculation, Figure B, as well as the calculation of misfit function.

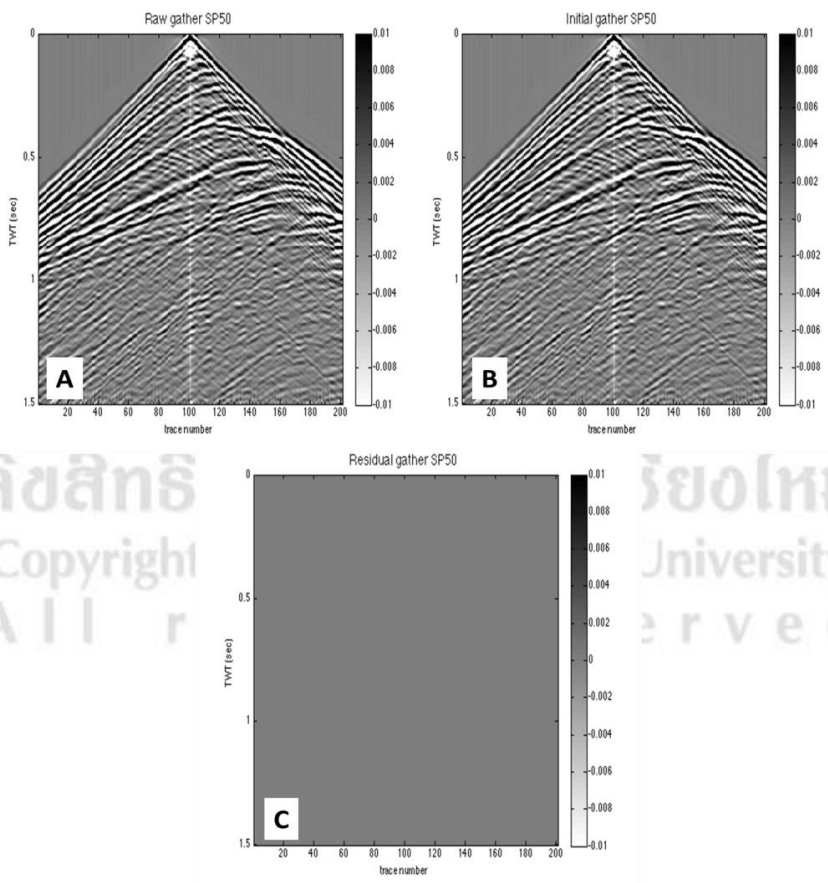

Figure A.1 example of shot gather (A) raw shot gathers 160 (B) initial shot gather 160 and (C) the residual gather 160 shows zero amplitude

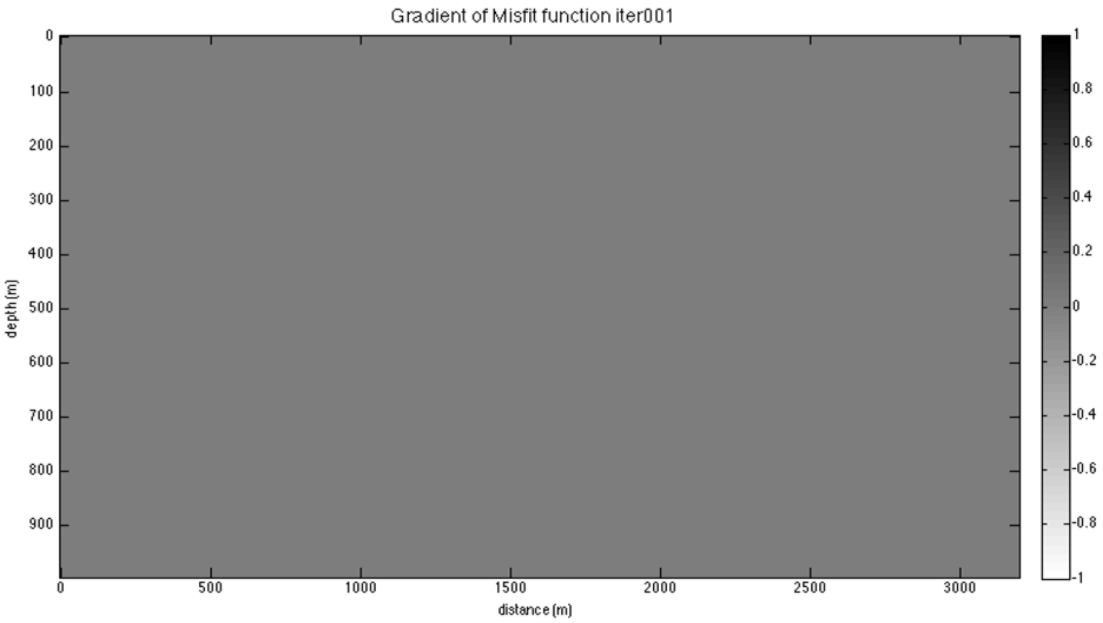

Figure A.2 A global gradient of misfit function compute from 19 shots show zero value for all cells.

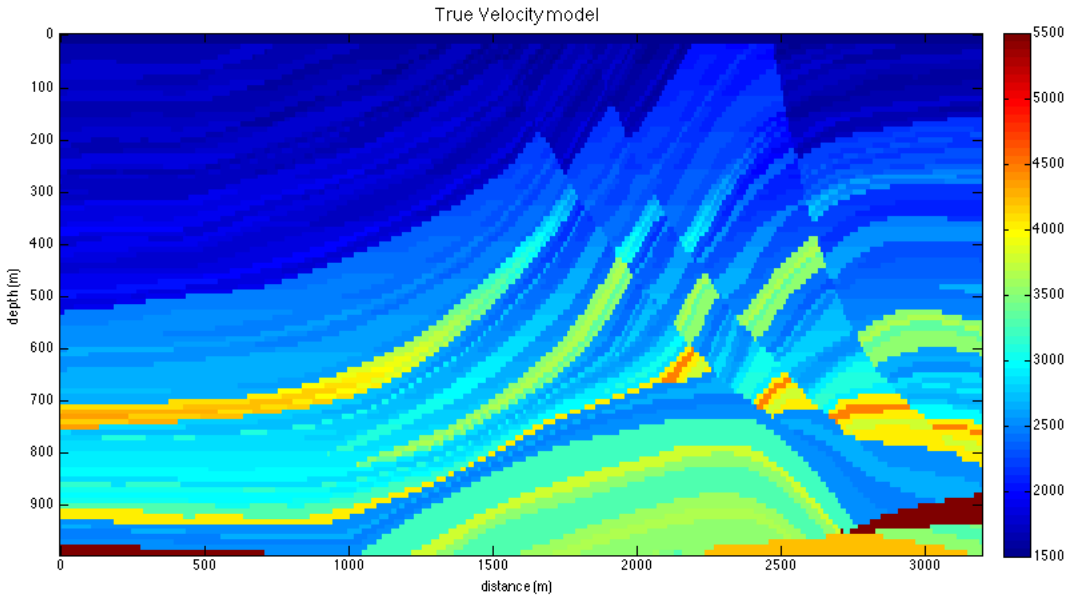

Figure A.3 selected of Marmousi model, which were used as true and initial velocity model.

Appendix B

Matlab code for FWI

a) Generating synthetic shot gathers with respect to the initial velocity model.

```
%-------------------------------------------------------------------------------------%
[inz,inx]=size(vmodel); %input velocity model
%extend velocity model
extx=200; extz=300;
nze=inz+extz; nxe=inx+extx;
vmodelx=zeros(nze,nxe);
vmodelx(1:inz,(extx+1):(extx+inx))=vmodel(1:inz,1:inx);
for ix=1:extx
   vmodelx(1:inz,ix)=vmodel(1:inz,1);
   vmodelx(1:inz,(extx+inx+ix))=vmodel(1:inz,inx);
end
for iz=1:extz
   vmodelx(inz+iz,:)=vmodelx(inz,:);
end
cd(intsp_path); % intsp_path
%crate shot record
dtstep=0.001;%time step
[nz,nx]=size(vmodelx);
x=(0:nx-1)*dx;z=(0:nz-1)*dx;
xrec=(extx:(extx+inx-1))*dx;
zrec=zeros(size(xrec));%receiver at zero depth
parfor count=1:ttsp
   % shoting every 16 cells
   sp_x=((count-1)*incsp)+incsp;
```

```
snap1=zeros(size(vmodelx));
xshot=dx*(extx+sp_x-1);%shot location
ix=near(x,xshot);
snap2=snap1;
snap2(1,ix(1))=1;%place the source
[seismogram4,seis4d,t]=afd_shotrec(dx,dtstep,dt,tmax,vmodelx,snap1,
                                   snap2,xrec,zrec, [0 5 35 50],0,2);
%trace selection
tr1=sp_x-100; tr2=sp_x+100;
if tr1 < 1
   tr1=1;
end
if tr2 > inx
   tr2=inx;
end
output=seismogram4(:,tr1:tr2);
% create trace header
th=zeros(120,size(output,2));
th(15,:)=1; th(18,:)=1; th(58,:)=size(output,1);
th(6,:)=sp_x;
th(8,:)=1:size(output,2);
% output segy
segy_name=['003_vinitSp' num2str(sp_x) '_' itr '.sgy'];
altwritesegy(segy_name,output,dt,[],[],[],[],[],[],th);
end
clear ix iz extx extz nze nxe vmodelx dtstep nz nx x z xrec zrec count th
%--------------------------------------------------------------------------------%
```

b) Creating residual data and calculate the misfit function

```
%-------------------------------------------------------------------------------------%
for count=1:ttsp
   % shoting every 16 cells
   spnb=((count-1)*incsp)+incsp;
   cd(rawsp_path);
   disp(['computed shot number:' num2str(spnb)])
   segy_name1=['002_RawSp' num2str(spnb) '.sgy'];
   [rawsp,~,ch,bh,th]=altreadsegy(segy_name1,'textheader','yes',
                                  'binaryheader','yes','traceheader','yes');
   cd(intsp_path);
   segy_name2=['003_vinitSp' num2str(spnb) '_' itr '.sgy'];
   initsp=altreadsegy(segy_name2);
   [nz1,nx1]=size(rawsp); [nz2,nx2]=size(initsp);
   if nz1-nz2==0 && nx1-nx2==0
      tmp1=rawsp-initsp;
      difsp=tmp1;
      segy_name=['004_diffSp' num2str(spnb) '_' itr '.sgy'];
      altwritesegy(segy_name,difsp,dt,[],[],[],[],[],[],th);
   else
      disp('data not compatible');
   end
   %plotimage(difsp);
   tmp01=difsp.^2;
   tmp02(count,1)=sum(sum(tmp01,2),1).*0.5;
end
% misfit function
misfit=sum(tmp02,1);
disp(['misfit function: ' num2str(misfit)]);
job1(3,:)=misfit;
%-------------------------------------------------------------------------------------%
```

c) Generate virtual source with respect to an initial model.

```
%-------------------------------------------------------------------------------------%
%crate data
cd(sp2rp_path);
%extended velocity model
[inz,inx]=size(vmodel);
extx=200; extz=300;
nze=inz+extz; nxe=inx+extx;
vmodelx=zeros(nze,nxe);
vmodelx(1:inz,(extx+1):(extx+inx))=vmodel(1:inz,1:inx);
for ix=1:extx
   vmodelx(1:inz,ix)=vmodel(1:inz,1);
   vmodelx(1:inz,(extx+inx+ix))=vmodel(1:inz,inx);
end
for iz=1:extz
   vmodelx(inz+iz,:)=vmodelx(inz,:);
end
%crate shot record
dtstep=0.001;%time step
[nz,nx]=size(vmodelx);
x=(0:nx-1)*dx;z=(0:nz-1)*dx;
%%%create receiver geometry%%%
% xrec=x; %receiver location
% zrec=zeros(size(xrec)); %receiver at zero depth
kk=0; xrec=zeros(1,(inz*inx)); zrec=zeros(1,(inz*inx));
for jj=1:inz
   for ii=extx+1:extx+inx
      kk=(kk+1);
      xrec(1,kk)=(ii-1)*dx; %receiver location
      zrec(1,kk)=(jj-1)*dx; %receiver at various depth
   end
end
```

```
parfor sp_x=1:inx % use parfor for parallel machine
   snap1=zeros(size(vmodelx));
   xshot=dx*(extx+sp_x-1);%shot location (location+extended grid)
   ix=near(x,xshot);
   snap2=snap1;
   snap2(1,ix(1))=1;%place the source
   %crate shot record
   %fourth order laplacian
   [sp2ssfrp,seis4,t]=afd_shotrec(dx,dtstep,dt,tmax,vmodelx,snap1,
                                  snap2,xrec,zrec,[0 5 35 50],0,2);
   %create number related to location and depth
   sp=1000+sp_x;
   % create trace header
   th=zeros(120,size(sp2ssfrp,2));
   th(15,:)=1; th(18,:)=1; th(58,:)=size(sp2ssfrp,1);
   th(6,:)=sp_x; th(8,:)=1:size(sp2ssfrp,2);
   % output segy
   disp(['output segy' num2str(sp)])
   segy_name=['010_sp2rp_' num2str(sp) '.sgy'];
   altwritesegy(segy_name,(sp2ssfrp),dt,[],[],[],[],[],[],th)
end
%-------------------------------------------------------------------------------%
% sort data
[nz,nx]=size(vmodel);
cd(sp2rp_path);
sp2rp=zeros(sample,nx,nz*nx);
output=zeros(sample,nx);
for kk=1:nx %from 1:nx
   sp=1000+kk;
   disp(['input sp2rp: ' num2str(sp)]);
   segy_name=['010_Sp2Rp_' num2str(sp) '.sgy'];
   tmp01=altreadsegy(segy_name); %read in segy file
```

```
    % change matrix dimention
    tmp02=permute(tmp01, [1 3 2]);
    sp2rp(:,kk,:)=tmp02;
end
clear tmp01 tmp02 kk sp segy_name;
% virtual source
cd(virtual_path);
for jj=1:(nz*nx)
    output=sp2rp(:,:,jj);
    spz=fix((jj-1)/nx)+1; spx=jj-((spz-1)*nx);
    vpoint=(((1000)*1000)+spx)*10000+spz;
    tmpv=(1000+spx)*10000+spz;
    % create trace header
    th=zeros(120,size(output,2));
    th(15,:)=1; th(18,:)=1; th(58,:)=size(output,1);
    th(6,:)=tmpv;
    th(8,:)=1:size(output,2);
    % output segy
    segy_out=['011_virtualsp_' num2str(vpoint) '.sgy'];
    disp(segy_out);
    altwritesegy(segy_out,output,dt,[],[],[],[],[],[],th);
end
%-------------------------------------------------------------------------------%
```

d) Calculation of the gradient of misfit function

```
%-------------------------------------------------------------------------------%
[nz,nx]=size(vmodel);
for count=1:ttsp
    grad_misfit=zeros(nz,nx); sp_nb=((count-1)*incsp)+incsp;
    cd(sp2rp_path);
    sp1=1000+sp_nb;
```

```
segy_name2=['012_firstArrivalTime_' num2str(sp_nb) '.sgy'];
fbpk=altreadsegy(segy_name2);% read in sp2rp gather
cd(intsp_path);
segy_name1=['004_diffSp' num2str(sp_nb) '_' itr '.sgy'];
tmpdiff=altreadsegy(segy_name1);% read in diff gather
diffsp=refor(tmpdiff,dt,2.0); % gain recovery for deeper part
tr1=sp_nb-100; tr2=sp_nb+100;
   if tr1 < 1
      tr1=1;
   end
   if tr2 > nx
      tr2=nx;
   end
cd(virtual_path);
for spz=3:nz
      trstt=sp_nb-(70+fix((spz)*40/nz));
      trend=sp_nb+(70+fix((spz)*40/nz));
      if trstt < 1
         trstt=1;
      end
      if trend > nx
         trend=nx;
      end
   parfor spx=trstt:trend %1:nx
      tmpfbpk=fbpk;
      sp2=(((1000)*1000+spx)*10000)+spz;
```

```
        segy_name3=['011_virtualsp_' num2str(sp2) '.sgy'];
        tmp5=altreadsegy(segy_name3);
        dif_pnt=tmp5(:,tr1:tr2);
       % generate partial derivative wavefields
        [iz,ix]=size(dif_pnt);
        jacob=zeros(iz,ix);
        kk=(spz-1)*nx+spx;
        v=tmpfbpk(:,kk);
        AA=conv2(dif_pnt,v);
        jacob(1:iz,:)=-AA(5:iz+4,:);
        tmp1=jacob;
       % gradient of misfit function
       % zero-lag cross correlation of partial wavefield & residual data
        tmp_xcrr=xcorr2(diffsp,tmp1);
        grad_misfit(spz,spx)=tmp_xcrr(size(jacob,1),size(jacob,2));
        tmp25=tmp_xcrr(size(jacob,1),size(jacob,2));
        disp(['calculating sp' num2str(sp_nb) ':' num2str(spz) '-'
              num2str(spx) '=' num2str(tmp25)]);
    end
  end
  cd(intsp_path);
  sp_out=(1000+sp_nb);
  segy_name4=['013_graMisFit_' num2str(sp_out) '_' itr '.sgy'];
  altwritesegy(segy_name4,grad_misfit,dx);
end
%-------------------------------------------------------------------------------%
```

e) Calculation of Global gradient of misfit function

```
%-------------------------------------------------------------------------%
[nz,nx]=size(vmodel);
tmp_all_grad=zeros(nz,nx,ttsp);
count=0;
for count=1:ttsp
   sp_nb=((count-1)*incsp)+incsp;
   sp_in=1000+sp_nb;
   disp(['input gradientMisFit Sp: ' num2str(sp_nb)]);
   cd(intsp_path);
   segy_name2=(['013_graMisFit_' num2str(sp_in) '_' itr '.sgy']);
   [tmp2,~,ch,bh,th]=altreadsegy(segy_name2,'textheader','yes',
                                 'binaryheader','yes','traceheader','yes');
   tmp_all_grad(1:nz,1:size(tmp2,2),count)=tmp2(1:nz,1:size(tmp2,2));
end
disp('stacking all gradientMisFit matrix');
all_grad=(sum(tmp_all_grad,3));
% update trace number or CDP number 1:nx
th(12,:)=1:(size(all_grad,2));
% update offset X
th(20,:)=0:dx:((size(all_grad,2))-1)*dx;
segy_name11=(['014_GlobalGradientMisFit_' itr '.sgy']);
altwritesegy(segy_name11,all_grad,dx,[],[],[],[],ch,bh,th);
%-------------------------------------------------------------------------%
```

f) Calculation of model perturbation and model updating

```
%-------------------------------------------------------------------------%
%input velocity model
cd([proj_path project '\MODEL']);
rmodel=altreadsegy('vmodel_raw.sgy');
offx=[0, ((size(rmodel,2)-1)*dx)];
offz=[0, ((size(rmodel,1)-1)*dx)];
```

```
figure(1);
imagesc(offx,offz,rmodel,[1500 5500]); colorbar;
title('Raw velocity model'); xlabel('Distance (m)'); ylabel('depth (m)');
%input velocity model
vint=vmodel;
[nz,nx]=size(vint);
figure(2);
imagesc(offx,offz,vint,[1500 5500]); colorbar;
title('Initial velocity model'); xlabel('Distance (m)'); ylabel('depth (m)');
% input gradient of misfit function
cd(intsp_path);
segy_name11=(['014_GlobalGradientMisFit_' itr '.sgy']);
[tmp,~,ch,bh,th]=altreadsegy(segy_name11,'textheader','yes','binaryheader','
yes','traceheader','yes');
grad=tmp;
% scale search
cd([proj_path project '\JOBS\master_jobs\']);
t041_scale_search_ver04;
dmodel=grad.*scale;
nmodel=vint+dmodel;
figure(3);
imagesc(offx,offz,nmodel,[1500 5500]); colorbar;
title(['Velocity model ' itr]); xlabel('Distance (m)'); ylabel('depth (m)');
cd([proj_path project '\MODEL']);
segy_out=(['vmodel_' itr '.sgy']);
altwritesegy(segy_out,nmodel,dx,[],[],[],[],ch,bh,th);
job1(3,:)=scale;
%-------------------------------------------------------------------------------%
```

g) Scale search for calculation of model perturbation

```
%-------------------------------------------------------------------------------------%
calsp=9; % calculated 11 shots
cal_scale=[0;5;10;20;40;80];
output=zeros(length(cal_scale),2);
for ii=1:length(cal_scale)
   tmp_scale=cal_scale(ii,1);
   output(ii,1)=tmp_scale; %output scale
   disp(['comput scale:' num2str(tmp_scale)])
   vtmp=vint+(grad.*tmp_scale);
   %extended velocity model
   [inz,inx]=size(vtmp);
   extx=200; extz=300;
   nze=inz+extz; nxe=inx+extx;
   vmodelx=zeros(nze,nxe);
   vmodelx(1:inz,(extx+1):(extx+inx))=vtmp(1:inz,1:inx);
   for ix=1:extx
      vmodelx(1:inz,ix)=vtmp(1:inz,1);
      vmodelx(1:inz,(extx+inx+ix))=vtmp(1:inz,inx);
   end
   for iz=1:extz
      vmodelx(inz+iz,:)=vmodelx(inz,:);
   end
   %crate shot record
   dtstep=0.001;%time step
   [nz,nx]=size(vmodelx);
   x=(0:nx-1)*dx;z=(0:nz-1)*dx;
   xrec=(extx:(extx+inx-1))*dx;
   zrec=zeros(size(xrec));%receiver at zero depth
   parfor count=1:calsp
      % shoting every 32 cells
      sp_x=((count-1)*32)+32;
```

```
    snap1=zeros(size(vmodelx));
    xshot=dx*(extx+sp_x-1);%shot location
    ix=near(x,xshot);
    snap2=snap1;
    snap2(1,ix(1))=1;%place the source
   [seismogram4,seis4d,t]=afd_shotrec(dx,dtstep,dt,tmax,vmodelx,snap1,
                                      snap2,xrec,zrec,[0 5 35 50],0,2);
    %trace selection
    tr1=sp_x-100; tr2=sp_x+100;
    if tr1 < 1
       tr1=1;
    end
    if tr2 > inx
       tr2=inx;
    end
  intsp=seismogram4(:,tr1:tr2);
  %input Raw shot gathers
  cd(rawsp_path);
  disp(['input raw shot number:' num2str(sp_x)])
  segy_name1=['002_RawSp' num2str(sp_x) '.sgy'];
  rawsp=altreadsegy(segy_name1);
  difsp=rawsp-intsp;
  tmp01=difsp.^2;
  tmp02(count,1)=sum(sum(tmp01,2),1).*0.5;
  end
  misfit=sum(tmp02,1);
  output(ii,2)=misfit; %output misfit
end
% calculated equation fit with this dataset
y1=output(:,2); sc1=output(:,1);
eq=polyfit(sc1,y1,4); % polynomial with power 4 equation
sc2=(0:1:100)'; y2=polyval(eq,sc2); output2=[sc2,y2];
```

```
% output xlsx file
cd([proj_path project '\LISTS']);
xlswrite(['scale_search_' project '_' itr '.xlsx'],output,1,['A1:B'
num2str(length(output))]);
xlswrite(['scale_search_' project '_' itr '.xlsx'],output2,1,['D1:E'
num2str(length(output2))]);
cd([proj_path project]);
[A,SI]=min(output2);
scale=output2(SI(1,2),1);
%--------------------------------------------------------------------------%
```

CURRICULUM VITAE

Name	Mr. Nopadol Phromsaeng
Date of Birth	May 18th, 1981
Education	Bachelor of Science (2003) in Physics, Faculty of Science, Chiang Mai University
Scholarship	Scholarship from PTT Exploration and Production Public Company Limited (PTTEP), Thailand for study at International program of Petroleum Geophysics, Chiangmai University (August 2014 – January 2016)
Publications	Nopadol U., Siriporn C., and Naghadeh D. H., (2015). Gradient of Misfit Function Approximation from Seismic Shot Records for Full Waveform Inversion, International Graduate Research Conference 2015 (iGRC2015), 11 December 2015, ST188-ST192.