Table of Contents

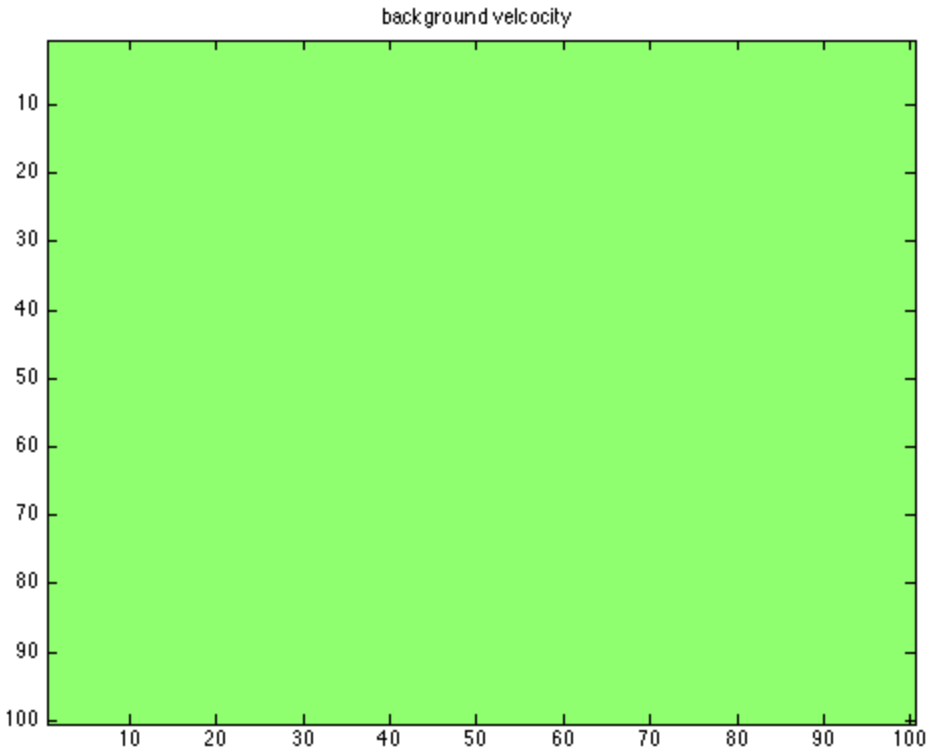

```
% addpath of some functions and operators
addpath /Volumes/Users/linamiao/Documents/Tools/Matlabtools/tuning/

clear
%close all
```

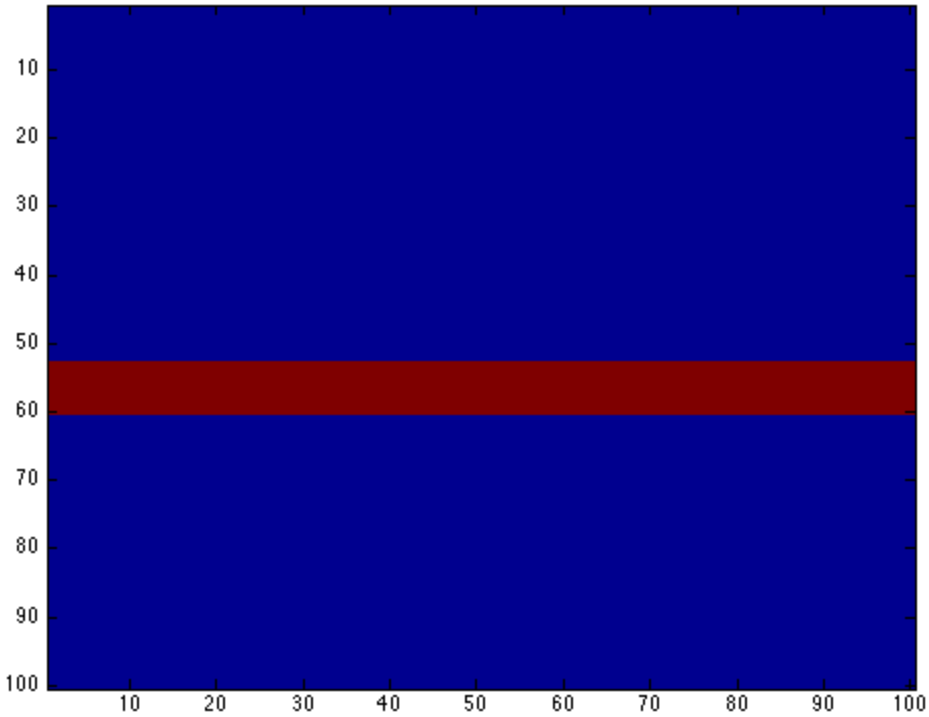
generate data

```
z = 0:10:990;
x = 0:10:990;
[zz,xx] = ndgrid(z,x);

% background velocity [m/s]
v0 = 1000 + 0.*zz;
figure;imagesc(v0);title('background velcocity')

% true velocity model
dv = 0*xx;
% dv(zz >= 0) = 1600;
% v(zz >= 300) = 2000;
dv(zz >= 520) = 600;
dv(zz >= 600) = 0;
figure; imagesc(dv);

% perturbation
%dv = v - v0;
figure;imagesc(dv);title('velocity perterbation')

% Modeling
model.o = [0 0];
model.d = [10 10];
model.n = [100 100];
model.nb = [30 30 0];

% frequencies [Hz]
model.freq = 5:20:125;
model.freq = 5:5:20;
nfreq = length(model.freq);

% Ricker wavelet peak frequency and phase shift
model.f0 = 20;
model.t0 = -.08;
```

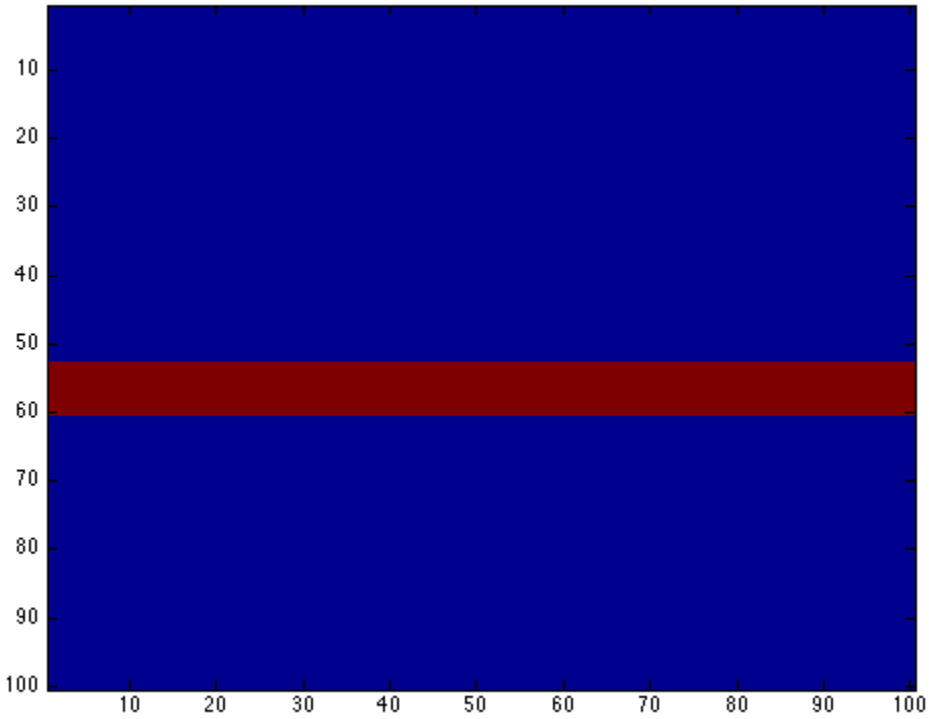
```
% source and receiver positions
model.zsrc = 10;
model.xsrc = 0:10:990; nsrc = length(model.xsrc);
model.zrec = 10;
model.xrec = 0:10:990; nrec = length(model.xrec);

% define point sources, each column of this matrix represents a source
% function defined on the grid {model.zsrc,model.xsrc}. A point source is
% represented as a spike on one of the gridp-points. If we take Q to be an
% identity matrix, each column represents a point-source on a different
% gridpoint.
Q = speye(nsrc);

% define model in [km^2/s^2]
m = 1e6./(v0(:) + dv(:)).^2;

% create data
[DD,J] = F(m,Q,model,1);

% linearize data
D = J*m;

% reshape vectorized data into data-cube for plotting purposes
D = reshape(D,[nrec,nsrc,nfreq]);

% plot frequency slices
figure;imagesc(real(D(:,:,1)));title('frequency slice of observed data')

% re-organize data into frequency-receiver-source order
D = permute(D,[3,1,2]);

% % plot one common shot gather in time domain
% % compensate negative frequency part
% n = length(model.freq);
% if mod(n,2) == 1
%    tmp = [D ; conj(D(end:-1:2,:,:))];
% else
%    tmp = [D ; conj(D(end-1:-1:2,:,:))];
% end
%
% Dt = ifft(tmp,[],1)*sqrt(n);
% figure; imagesc(real(Dt(:,:,51)));colormap(gray);


% if pad zero to other frequencies part
% frequency axis
% f = 0:5:125;
% nf = length(f);
% temp = zeros(nf,length(x),length(x));
% for i = 1:n
%     idx = find(f==model.freq(i));
%     temp(idx,:,:) = D(i,:,:);
% end
```

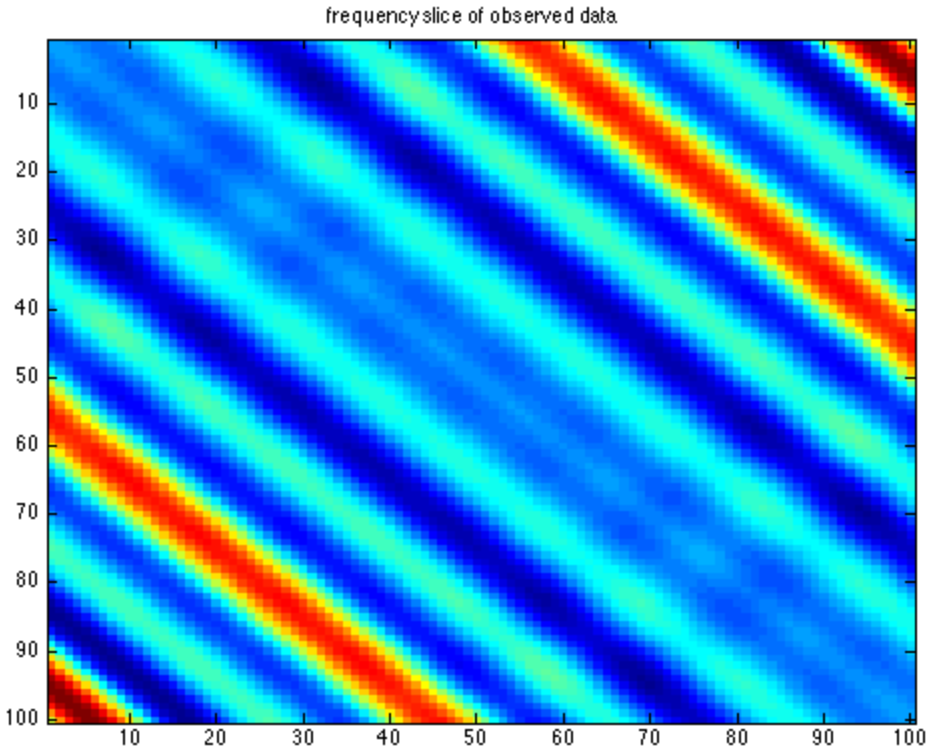
```
%
%
% if mod(n,2) == 0
%    tmp = [temp ; conj(temp(end:-1:2,:,:))];
% else
%    tmp = [temp ; conj(temp(end-1:-1:2,:,:))];
% end
% Dt = ifft(tmp,[],1)*sqrt(n);
% figure; imagesc(real(Dt(:,:,51)));colormap(gray);
%


% save modelled data
save model_data DD D model x z dv v0 ;




%keyboard;
```





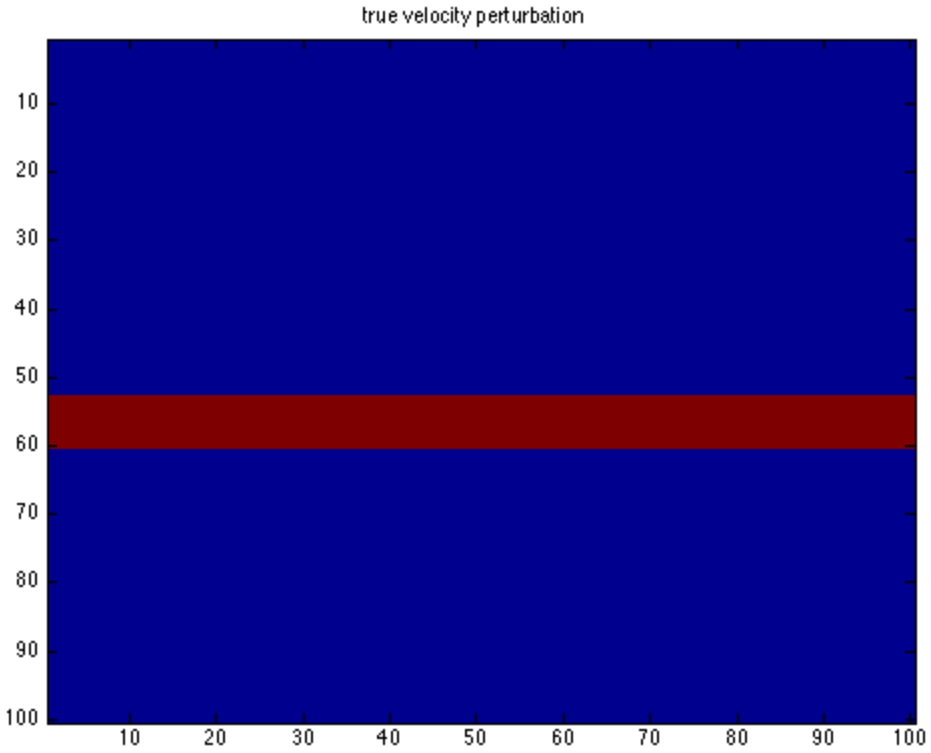
velocity perterbation

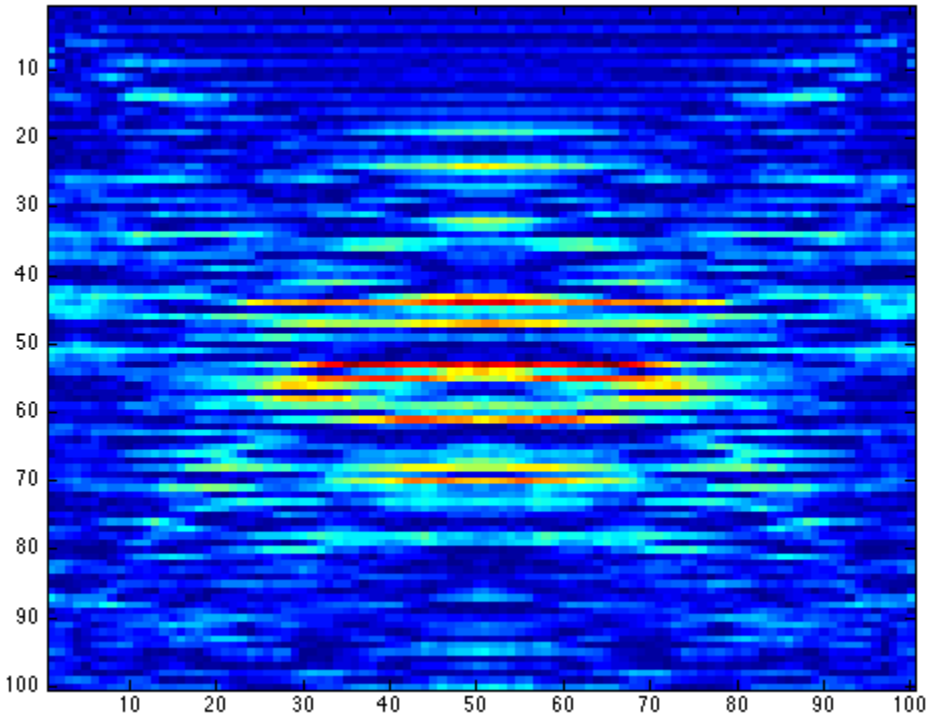




migration in frequency domain

```
clear;%close all;
load model_data;
figure;imagesc(dv);title('true velocity perturbation')
data = D;
image = DSR_mig_freq(data,model.freq,x,x,z,v0(1:length(z)));
figure; imagesc(abs(image));

save img image
```


true velocity perturbation
10
20
30
40
50
60
70
80
90
100
10
20
30
40
50
60
70
80
90
100


10
20
30
40
50
60
70
80
90
100
10
20
30
40
50
60
70
80
90
100

first transform data into time domain, then use DSR_mig in time domain

keyboard; clear;%close all; load model_data; figure;imagesc(dv);title('true velocity perturbation')

% pad zero to other frequencies part % frequency axis n = length(model.freq); faxis = 0:.5:120; nf = length(faxis); temp = zeros(nf,length(x),length(x)); for i = 1:n idx = find(faxis==model.freq(i)); temp(idx,:,:) = D(i,:,:); end

if mod(n,2) == 1 % cause faxis has 0 component inside tmp = [temp ; conj(temp(end:-1:2,:,:))]; else tmp = [temp ; conj(temp(end-1:-1:2,:,:))]; end Dt = ifft(tmp,[],1);%*sqrt(n);

data = Dt; fmax = faxis(end); df = faxis(2) - faxis(1); t = [0:.5/fmax:1/df]; figure; imagesc(real(Dt(:,:,51)));colormap(gray);

image = DSR_mig(data,t,x,x,z,v0);

figure;imagesc(real(image));

Published with MATLAB® 7.13