Table of Contents

..... 1
read data 1
simulate simutaneous experiment 2
sparsity transform 2
DSR extended migration 2
inversion via joint sparsity 2
evaluate result 2

time domain.

```
% addpath of some functions and operators
% addpath /Volumes/Users/linamiao/Documents/Tools/Matlabtools/tuning/
% addpath(genpath('./simu_functions'))
% addpath ../data
% addpath ../functions

clear
% close all
```

read data

```
[data,SuTraceHeaders,SuHeader]=ReadSu('data_ex3.su');

% source and receiver coordinates per trace
for k=1:length(SuTraceHeaders)
    xs(k)= SuTraceHeaders(k).SourceX;
    xr(k)= SuTraceHeaders(k).GroupX;
end

% note that the sample interval is given in miliseconds!
t = [0:SuHeader.ns-1]'*SuHeader.dt*1e-6;

% source and receiver coordinate vectors
xs = unique(xs);
xr = unique(xr);

% reshape data into cube
data = reshape(data,length(t),length(xr),length(xs));

% window and subsample
data(1:100,:,:) = 0;
data = data(101:101+127,1:10:end,:);
xr = xs;
t = t(101:127+101);
```

```
Attempt to reference field of non-structure array.

Error in time_domain (line 22)
```

```
t = [0:SuHeader.ns-1]'*SuHeader.dt*1e-6;
```

simulate simutaneous experiment

```
nt = length(t);
ns = length(xs);
nr = length(xr);

% measurement matrix
p = .5;
RM = opSimSourcePeriodTimeDither([nt nr ns], [floor(p*nt*ns),nr], ns);

% observed data from simutaneous experiment
simD = RM*vec(data);
simd = reshape(simD,64,11,11);
t = t(1:nt/2);
```

sparsity transform

wavelet operator along time axis

```
W = opSplineWavelet(nt, 1, nt, 3, 5);

% curvelet operator along source-receiver oordinates
C = opCurvelet(nr,ns,6,16,1,'ME',0);

% oppKron2Lo : kronecker tensor product to act on a distributed vector
S = oppKron2Lo(C, W', 1);
```

DSR extended migration

```
v = 1000*ones(length(z),1);
z = 10:10:1000;
K = opDSR(dim,t,x,y,z,v);

M = K*data; % DSR result if not sim
```

inversion via joint sparsity

```
A = RM*K'*S';
B = reshape(simD,nt*ns,nr);
fid = fopen('log_log.txt', 'w');
opts = spgSetParms('optTol',1e-4);
sigma = 1e-3;
[X_hat,R,G,INFO] = spg_mmv(A,B,sigma,opts);
```

evaluate result

```
M1 = S'*X_hat;
SNR1 = snr(vec(M),vec(M1));
```

```
D1 = K'*M1;
SNR2 = snr(vec(D1),vec(data));
```

Published with MATLAB® 7.13