Table of Contents


```
function v = my_step(u,tf,x,c,dz,flag)

% wavefield extrapolation by phase shift in 3D
%
% use:
%   v = my_step(u,t,x,c,dz,flag)
%
% input:
%   u    -
%       flag = 1, data is given in frequency domain(default, if not specified
%                    by user)
%                 - u(f,r,s)
%                 is the 3D data volumn of a 2D seismic survy observed at the
%                 surface z = 0 in frequency domain, first dimension is frequency,
%                 second is receiver, third is source
%                 - tf is the frequency coordinate
%
%       flag = 2, data is given in time domain
%                 - u(t,r,s)
%                 is the 3D data volumn of a 2D seismic survy observed at the
%                 surface z = 0 in time domain, the first dimension is time, the
%                 second is receiver, the third is source
%                 - tf is the time coordinate
%   t    - time coordinates in seconds as column vector
%   x    - space coordinates in meters as row vector(as source and receiver
%          are at the same grid)
%   v    - velocity in m/s (vector same length as x)
%   dz   - depth step in meters
%   flag-
%
% output:
%   v    - extrapolated wavefield as a matrix same dimension with input u
%
```

if length of t and x are not power of 2

pad zeros as fktran

transform to desired domain(w-g-s)

```
if not(exist('flag','var'))
```

```
    flag = 1;
end
Ft = opDFT(size(u,1));
It = opDirac(size(u,1));
Ir = opDirac(size(u,2));
Is = opDirac(size(u,3));
if flag == 1 %data is given in frequency domain
    U = vec(u);
    f = tf;
else % data is given in time domain
    F = opKron(Is,Ir,Ft);
    U = F*vec(u);
    t = tf;
    fnyq = 1. / (2*(t(2)-t(1)));
    f = [0:df:fnyq-df -fnyq:df:-df];
end

% w = f;
% f = 2*pi*f;
```

Error using my_step (line 41)
Not enough input arguments.

prepare for the second order derivative

```
dx = x(2) - x(1);
n = length(x);
D = (1/dx)^2.*spdiags([ones(n,1),-2*ones(n,1),ones(n,1)],-1:1,n,n);
D(1,2) = 0; D(end,end-1) = 0;
DDS = opKron(D,Ir,It);
DDR = opKron(Is,D,It);
```

compute DSR

```
[ff,vv,~] = ndgrid(f,c,x);
temp = (-1j.*ff./vv).^2;
temp = vec(temp);
U2 = U.^2;
DSR1 = sqrt(temp.*U2-DDR*U2);
DSR2 = sqrt(temp.*U2-DDS*U2);
DSR1 = -real(DSR1)+1i*abs(imag(DSR1));
DSR2 = real(DSR2)+1i*abs(imag(DSR2));
DSR = (DSR1 + DSR2);
%DSR        = -real(DSR)+1i*abs(imag(DSR));
```

propagate

```
U      = U + 1e0*dz*DSR;

v = reshape(U,size(u));
```

if padding zeros

```
v = v(1:length(t),1:length(x));
```

Published with MATLAB® 7.13