LAB 5- Migration

GOPH557-March 20, 2020

This lab has two parts. The first part (first week) is comparing time and depth migration. The second part (second week) is to comparing post and prestack migration. Paste your code and figures and explain what the figures are.

Part A: Time and Depth migrations.

You can use the Demo_Kirchhoff_Migration from D2L (lecture 21) as a template for this lab. You don't need to do all the steps on that demo, just take the parts you need (essentially just creating a zero offset section and migrate it with Kirchhoff (time only).

Take the velocity model 'thrustmodel' with a cell size of 10 m (dx=10m). You can refine to 5m later.

```
[vel,x_thrust,z_thrust]=thrustmodel(dx);
```

and perform an exploding reflector model with finite differences to generate the zero-offset section data. Follow the example in the demo to set w, tw, etc. For example:

```
[seismogram,seis,t]=afd_explode(dx,dtstep,dt,tmax,vel,xrec,zrec,w,tw,laplacia
n,boundary);
```

Then try two different post-stack migrations, time and depth, and paste the reflectivity results below:

a) Kirchhoff Time Migration:

```
[seismig,tmig,xmig]=kirk_mig(seismogram,vrms,t,xrec,params);
```

To generate the Vrms you will need to convert true velocities to vrms with

```
[vrms,tv,vint]=vz2vrms(vel,z,t(2)-t(1),max(t));
```

In the demo from lecture 21 there are several additional tests, like migrating one trace and few traces. You can play with that but those are not needed for this report.

b) Depth Migration: for making a fair comparison we should use a Kirchhoff depth migration, but we don't have one in the Crewes toolbox. The one used in the demo from lecture 21 is really a time migration stretched to time (it does not use ray tracing). To compare time and depth we will use PSPI depth, which is a wave equation migration method we will see in the lectures in a few days. For this lab, all you need to do is to run the crewes version as follows:

```
pspimig=pspi_stack(seismogram,t,xrec,vel,xvel,zvel);
```

You can now compare the image in both cases and write conclusions. Although PSPI is somewhat better than Kirchhoff migration, most of the differences we see are because of using time vs depth migration. To understand how much the structure affects time migration, take

another velocity model from the Crewes toolbox, or build your own, and repeat the tests above with this simpler structure (examples, flatmodel, synclinalmodel, channelmodel, wedgemodel). NOTE: some of these simple models show poor results in time migration. That is (probably) because the RMS velocities we are using are suboptimal. You can try changing the models with smaller velocity contrasts and see if time image improves. Fell free to play with the tools, there isn't a unique response for this lab. Also migrations functions have several parameters that you can change.

Part B: Pre-stack time and Depth migrations

In this part you will generate shots with different shots and receiver locations using

```
snap1=zeros(size(vel));
snap2=snap1;
snap2=(index_ts,index_xs)=1;
[shot,seis,t]=afd_shotrec(dx,dtstep,dt,tmax,vel,snap1,snap2,xrec,zrec,w
,tw,laplacian);
```

The location of the shot is specified by a spike on snap2 (please review Matlab lab at the beginning of the semester). You can put the shots anywhere you want on the surface.
It is a good idea to use only one shot first until all is working and then add extra shots.

To obtain a better result from migration is necessary to eliminate the directwave.
This is usually done during processing with mutes. However, for this lab, since you are generating the data from synthetic models, you can generate the directwave using a velocity model with constant velocity equal to the velocity on the top layer (where shot and receivers are). Then you can subtract the directwave from shots. Notice each shot requires you to generate the directwave for the location of the shot, so effectively you have to run two finite difference modelling for each shot, one with the original velocity, another one with constant velocity.
For example:

```
%create shot locations vector;
ixshots=zeros(nshots,1);
for is=1:nshots
    ixs=firstShot+is*dxs;
    ixshots(is)=ixs;
end
xshots=ixshots*dx;
nt=round(tmax/dt+1);
data=zeros(nt,nx,nshots);
for is=1:nshots
      snap1=zeros(size(vel));snap2=snap1;
      snap2(5,ixshots(is))=1;%snap2 is zero except at the source location

      [shotrecord,seis,t]=afd_shotrec(dx,dtstep,dt,tmax,vel,snap1,snap2,x,zer
      os(size(x)),w,tw,laplacian);
      snap1=zeros(size(vel));snap2=snap1;
```

```
      snap2(5,ixshots(is))=1;

      [directwave,seis,t]=afd_shotrec(dx,dtstep,dt,tmax,veldirectwave,snap1,s
      nap2,x,zeros(size(x)),w,tw,laplacian);
      data(:,:,is)=(shotrecord-directwave);
end
```

Prestack Kirchhoff Time Migration

For Kirchhoff time migration, you can use the following:

```
[vrms,tv,vint]=vz2vrms(vel,z,t(2)-t(1),max(t)); % create RMS velocity
 [shotmig,tmig,xmig]=kirk_shot(data(:,:,is),t,x,xshots(is),vrms,tv,x);
```

Notice that the output of kirk_shot has axis of time because it is a time migration. Also, notice that the xmig axes is automatically calculated using the dxr/2 where dxr is the receiver interval. If you use dxr=dx (one receiver on each velocity cell), then the image will have a 2*nx-1 length along x (which is okay).

Because each migration produces the image for one shot, you can add them together inside the shot loop. Example:

```
[vrms,tv,vint]=vz2vrms(vel,z,t(2)-t(1),max(t)); % create RMS velocity
migkir=zeros(nt,2*nx-1);
for is=1:nshots
    [shotmig,tmig,xmig]=kirk_shot(data(:,:,is),t,x,xshots(is),vrms,tv,x);
    migkir=migkir+shotmig;
end
```

Prestack PSPI Depth Migration

As a second test, you will use prestack PSPI depth migration. In this case you use:

```
[shotmig,shotmig_cc,illumination]=pspi_shot(data(:,:,is),t,x,vel,x,z,xshots(i
s),frange,stab);
```

Again, each pspi_shot outputs the image for one shot, so you must add them together in a loop as shown for Kirchhoff.