**Geol 335.3**

# Lab #4: Seismic Modelling

In this lab, you will study the response of the direct, reflected and refracted waves in a simple layered Earth model, and generate a simple seismic plot using Matlab or Octave. The three-layer and a half-space model is given in the following table:

| Layer | Thickness (m) | Velocity (m/s) |
|---|---|---|
| 1 | 100 | 450 |
| 2 | 200 | 2000 |
| 3 | 200 | 3000 |
| 4 | ∞ | 6000 |

Copy to yourself and unpack the zipped archive. Modify the examples in script `lab4.m` (or create your own). In this

Let us simulate a seismic survey conducted over this layered structure with the following specifications:

- Source located at $x = 0$;
- Receivers located from $x = -1000$ m to 1000 m;
- Spacing between receivers 10 m.

This layout can be described in Matlab by writing:

```
x_source = 0.0;         % source position
x = -1000:10:100;       % geophone positions
```

The modelling function `trefr(…)` provided in the lab is most convenient to use if the depth/velocity model is provided in the form of arrays. The model above can then be described in two Matlab variables:

```
h = [ 100, 200, 200 ];                % layer thicknesses
v = [ 450, 2000, 3000, 6000 ];        % layer velocities
```

With these initial settings, perform the following modelling steps:

Task (1) (10%) **For each velocity contrast in the model, determine**: a) critical ray parameters; b) critical distances, c) intercept times for the refractions, d) $t_0$ times and RMS velocities for the reflections. In Matlab, assign identifiers to these quantities, such as:

```
p_crit_1 = …;     % critical ray parameter
```

```
x_crit_1 = …;     % critical distance
t_int_1 = …'      % intercept time
t_int_2 = …;
t0_1 = …;         % normal-incidence reflection time;
v_rms_1 = …;
```

and so on. <u>In the following, *use these identifiers* rather than constants</u>.

<u>Note:</u> the units for distance should be m, time – ms, velocity – accordingly, m/ms (or km/s).

Task (2) (25%) Using Matlab, **compute the arrival times** of the direct, reflected and refracted waves from all the acoustic discontinuities for all receivers. Plot these arrival times on a common T-X graph. It would be helpful to write two Matlab *functions* to calculate reflection and refraction travel times, respectively:

`trefl(t0,v_rms)` and `trefr(t_intercept,v_refractor)`

In the subsequent steps, you will simply use e functions instead of retyping the formulas.

Next, look into the function `wavelet_ft(t)`. This function returns a wavelet (array) of amplitude `1.0`, centred at time `t=0` and discretized at times specified in array `'t'` (in milliseconds). The wavelet is obtained by <u>summation of 81 cosine waves</u> of amplitude `1.0/81.0`, with frequencies from 20 Hz to 60 Hz with a 0.5 Hz interval[1]. Finally, the waveform is multiplied by a sine "taper" function to ensure that its values at the ends of the time interval equal zero.

<u>The obtained wavelet is your *source signature*</u> (aligned at 0 ms time). Such symmetric with respect to time = 0 waveforms are called "zero-phase".

Task (3) (10%) Test function `wavelet_ft()` and **precompute the source waveform** for 30-Hz dominant frequency by executing something like this:

```
global tref wref

tref = -500:1:500;
wref = wavelet_ft(tref,30);
plot(tref,wref,'b-')
xlabel('Time (ms)')
ylabel('Displacement')
title('Wavelet at 30 Hz')
save wavelet.mat tref wref
```

[1] We will return to this procedure of constructing pulses from sinusoidal functions when studying Fourier transforms in the lectures and lab 7.

These commands create an array of times `tref` centred at `t=0`, model wavelet `wref` on it, plot `wref`, and save it in a file. This precomputed wavelet can be loaded like this:

```
global tref wref
load wavelet.mat
```

This `.mat` file will be used later in this and other labs.

Task (4) (10%) Look into the second Matlab function `wavelet(a0,t0,t).` This function uses the precomputed `tref` and `wref` and returns the same wavelet but with arbitrary amplitude `'a0'` and centred at arbitrary time `'t0'` [ms]. The output is obtained by linear interpolation (function `interp1(…)`), and this function works is much faster than the cosine-wave summations above.

**Test this function** like this (you can change parameter values to achieve nice plots):

```
t = 0:2:1000;
plot(t, wavelet(5.0,100.0, t ) );
```

**Print out the resulting plot out and indicate** whether the time and amplitude of the waveform are as expected.

Task (5) (40%) Using the calculated arrival travel-times and `wavelet(…)` function, **use Matlab to plot a *<u>shot gather</u>*** – a set of seismic traces simulated at each of the 100–200 geophone locations.

For time sampling in the plots, use 0–1200 ms time range, at 2 ms intervals:

```
t = 0:2:1200;     % time sampling
```

Note that the provided function `trefl(…)` (modelling reflection times) uses an approximation which is accurate only for the single-layer case. Therefore, model reflections only from the uppermost boundary. As a result, you should have five waves in your section:

- Direct wave through the first layer;
- Three head waves from each refracting boundary;
- Reflection from the top of the second layer.

For plotting, you use Matlab's `hold on` command (to prevent from starting a new display window on every `plot()` call) and arrange a loop over the receivers:

```
figure(1)
clf          % clear the figure in case it gets reused
```

```
hold on

for i=1:numel(x)

  % place the plotting commands for i-th receiver
  % (see below) here …

end
```

In this loop, first determine the offset, then the travel for each wave, and then attach a wavelet to this travel time. Sum the waveforms in a common array `trace`, which will simulate the seismic record:

```
% determine the source-receiver distance for this trace
offset = abs(x(i) – x_source);

% put direct wave into new trace
trace = wavelet(amp, offset/v(1), t );

% add reflection from the first interface
trace = trace + wavelet(amp, trefl(t0_1,v_rms1, offset), t );

% add into 'trace' refraction from the first interface

if x > x_crit_1
  trace = trace + wavelet(amp, trefr(t_int_1,v(1)), t );
end

% ... from the second interface:

if x > x_crit_2
  trace = trace + wavelet(amp, trefr(t_int_1,v(1:2)), t );
end

% ... and similarly for the third interface …

% plot the resulting seismic trace:
plot(trace+x,t);
```

Here, `t_direct`, `t_head`, and `t_reflected` should be the times of your arrivals.

The horizontal axis of the plot will become the distance, and the shifts of `x` added to the '`trace`' values will place your wiggles ("traces") at the positions of geophones. Parameter `amp` becomes trace excursion (swing amplitude) and should be selected so that the plot looks nice (I suggest ~15-20).

Time should be plotted along the vertical axis downward, as it is commonly done in refraction/reflection record sections. This direction of the axis can be achieved in Matlab by command

```
set(gca,'ydir','reverse');
```

Task (6) (5%) **Note the interference (overlap)** between the different waves near critical and crossover distances, and at far offsets. **Mark these interfering events in your section**. Comment on the tuning effects (<u>enhancement or cancellation of arrival amplitudes</u> as they approach each other within a dominant period.

## *Theory:*

The cosine function of amplitude *A*, frequency *f*, and centered at time $t_0$ (used in `wavelet_ft()`) is:

$$u(t) = A\cos(2\pi f(t - t_0))$$

For a stack of layers, travel time of the reflection from the bottom of *n*-th of them is given by the same formula as within a single layer:

$$t(x) = \sqrt{t_0^2 + \left(\frac{x}{V_{RMS}}\right)^2},$$

where $t_0$ is the vertical-incidence travel time, and $V_{RMS}$ is the effective (average, "root-mean-square") velocity of the stack of layers:

$$V_{RMS}^2 = \frac{\sum_{i=1}^{n} V_i^2 \Delta t_i}{\sum_{i=1}^{n} \Delta t_i},$$

where $\Delta t_i = \Delta z_i / V_i$ is the vertical travel time for each layer.

## *Hand in:*

Code printouts, plots, and comments on paper or by email (zipped or merged in a Word, PDF, or PowerPoint file).