Geol 335.3

Lab #5: Reflection Seismic Synthetics

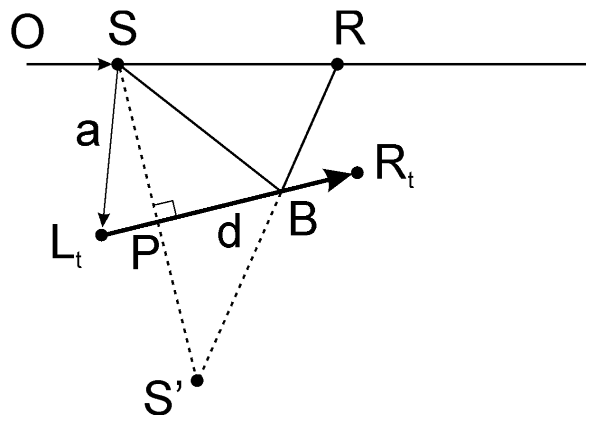
In this lab, you will write a Matlab program to study reflections from a complexly shaped reflecting interface. The program will also be used to produce a zero-offset section that will be inverted in the next lab.

Theory

As usual when writing a reasonably complex code, it is important to decide on the adequate description of the model and data. Identify Matlab data structures that most naturally represent the geometrical and physical entities used in the project. In Matlab, the main data structure is a matrix (vector), and it offers natural ways for adding vectors, calculating dot and vector products, taking matrix inverses, etc. It makes sense to take advantage of these capabilities by using vector/matrix notation. For example:

To represent:	*Use:*
Coordinates and vectors: (x,z)	2-element arrays `[x,y]`
Reflectors	2-dimensional arrays (e.g., matrices; arrays of coordinates)
Time series	1-dimensional arrays
Output data sections	2-dimensional array (arrays of time series, each associated with a surface location)

Accordingly, the problem can be treated and encoded in Matlab best with the use of vectors and linear algebra. Consider the following reflection geometry:



To find the reflection travel time from a single segment of the interface, you need to determine the distance from the mirror image of the source **S'** to the receiver **R**. The projection of the source onto the reflector is given by: **SP** = a + γ**d**, where $0 \le \gamma \le 1$. The value of γ can be found from orthogonality condition: **d**·**SP**=0, from which: γ=-**d**·**a**/(**d**·**d**). The position of the image of the source is then:

$$\mathbf{OS'} = \mathbf{OS} + 2\mathbf{SP},$$

which can be easily calculated in Matlab.

Once you know the position of the source mage **S'**, the reflection travel times becomes simply t=|**S'R**|/V. However, in order to plot the ray, you will also need to find the position of reflection point B (which is the intersection of **S'R** and $\mathbf{L}_i\mathbf{R}_i$). Once again, by using some parameter λ, point B lying on d can be represented by vector: **SB**=**a**+λ**d**. Vector **S'B** is then: **S'B= SB**-2**SP= a**+λ**d**-2 **SP**. For point B to lie on **S'R**, cross-product of **S'R** and **SB** must equal 0 (recall that vector cross-product is proportional to sine of the angle between the vectors):

$(\mathbf{a}+\lambda\mathbf{d}-2\,\mathbf{SP})\times\mathbf{S'R}=0.$

Only the transverse (across the plane of the plot; let's label this axis `y`) component of this equation is not automatically zero, and from this component, we obtain:

$\lambda = -[(\mathbf{a} - 2\,\mathbf{SP})\times\mathbf{S'R}]_y \,/\, [\mathbf{d}\times\mathbf{S'R}]_y.$

To determine whether the reflection point **B** lies within the reflecting segment $\mathbf{L}_t\mathbf{R}_t$, note that λ should be between 0 and 1.

Code

Try subdividing the code into a set of functions with well-defined functionalities. Common parameters, such as the model and receiver geometry, can be provided in common data space using the `global` declaration in Matlab:

```
global      receivers   % receiver locations
global      sources     % source locations
global      velocity    % velocity
global      layer       % array of reflecting interface points
global      time        % time sampling
```

Here, array `receivers` should contain a list of receiver positions, and array `time` – list of times at which the resulting records are sampled. `Velocity` is the overburden velocity. These variables should of course be set before calling any function using them.

The following functions are required:

1. Model setup, principally containing the global statements above; this script is provided in file `model.m`;
2. Interactive picking of a reflective interface (this function `picklayer` is provided);
3. Function returning reflection travel times from one source and several dipping reflecting segments. Its declaration would look like:

```
function [reflections] = diprefl(S, reflector)
```

Here, `S` represents the (x,z) coordinates of the source and `reflector` is the array of reflector points. The function should return the resulting reflection amplitudes in 2-D array `reflections`.

Make sure to document the function by providing commentaries describing the parameters, assumptions, and return values.

Assignments

1. Write and test the necessary auxiliary functions.
2. Execute script `model.m` and pick a reflecting boundary (5-8 points) with a valley, a crest, and an approximately horizontal segment in it. Make (save) a plot and save the model as Matlab namespace.
3. Write and execute a Matlab program to plot reflections from each of the shot points specified in file `model.m`. Each plot should include a plot of reflected rays and the resulting synthetic reflection section (using function `wavelet` from the previous lab).
   a. Describe the differences in the sections from shots over the different structures (up- and down dip);
   b. Save the plots and include them in the report.
4. Write and execute another Matlab program to create a synthetic “zero-offset section” resulting from collocated sources and receivers (one receiver for each shot). Use the receiver distribution in `model.m`.for both sources and receivers. *Hint:* The only modification you will have to make to you code would be to replace the `receivers` array with a single-point array in each call to `diprefl` function.
   a. Save the plot and include it in the report;
   b. Discuss the relation of the resulting section to the model of the reflector.

Hand in:

Zipped directories containing:

1. All Matlab codes (“m-files”);
2. Screen captures or Postscript/PDF figures;
3. Summary and discussion in a Word file.