Derived moveout operators

[1]In this chapter we describe the implementation of various types of moveout operations, based on the general moveout operator introduced in ??. We will see that the coding task is fairly simple, since all we have to do is define our specific moveout operator as a class derived from the general moveout `fopMO` class (see page ??), so that it will inherit all its properties. The virtual function, `float fopMO::mapping()`, creates a dynamic binding between the two classes, telling the general moveout how to perform the squeezing/stretching operation.

We will discuss in the next sections the linear moveout operator, the normal moveout, the anelliptic moveout and the Kirchhoff moveout. Figure 1 shows an inheritance sketch of these three classes and its respective mapping equations. Arrows indicate derivation.

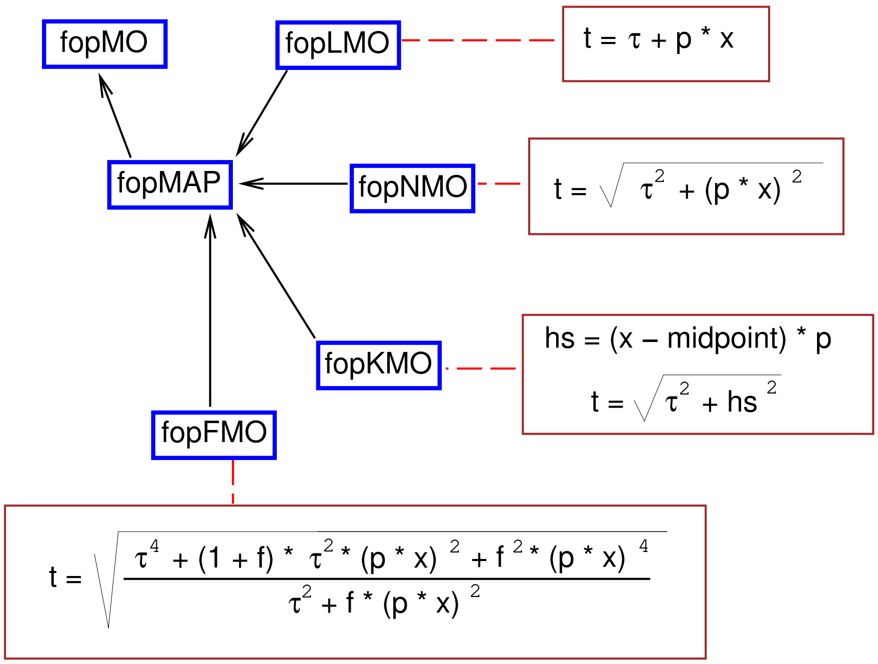

Figure 1: Graph of dependencies and definition of the mapping equations. Moveout-mapping [NR]

[1]Dave Nichols, Martin Karrenbach, Hector Urdaneta

LINEAR MOVEOUT

[2]To do linear moveout correction (LMO), we need to time-shift data within each trace. Shifting data requires us to interpolate it. The easiest interpolation method is the nearest-neighbor method. For a given offset, we shift the data using the equation $\tau = t - px$. Given the location τ of the desired value, we backsolve for an integer. In the CLOP library this is done by calling the function `index()` (see man page ??), a member of the `Axis` class. This gives the nearest neighbor. The adjoint operation copies τ space back to t space. Figure 2 shows a CMP gather shifted upwards using LMO.

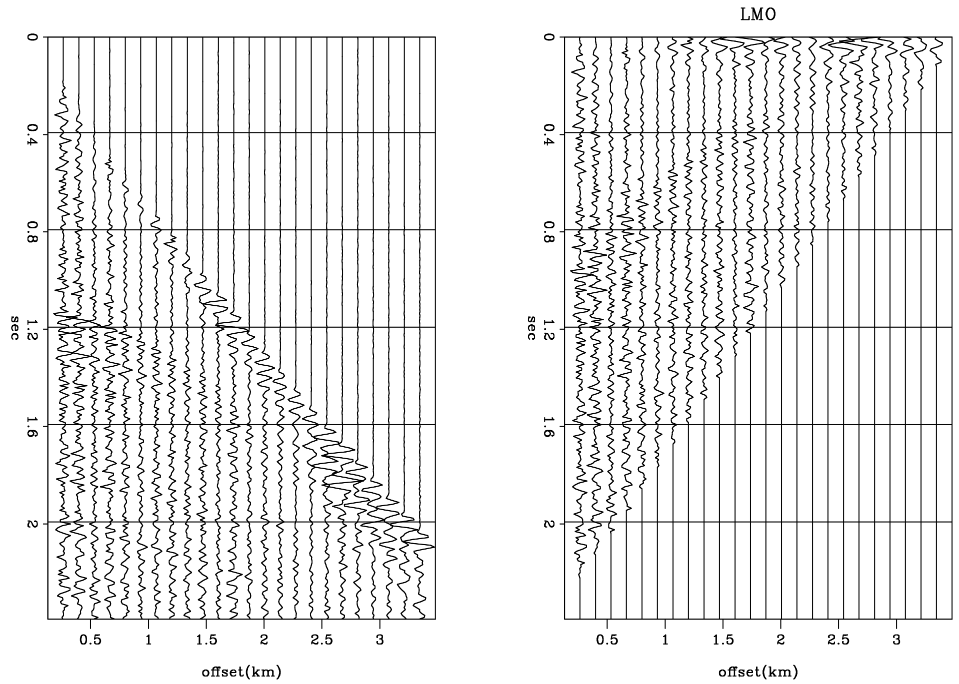

Figure 2: A common midpoint gather from the Gulf of Mexico before (left) and after (right) linear moveout at water velocity. Moveout-wglmo [NR]

The mapping function for the linear moveout operation is given in `foplmo` (page 3).

```
class fopLMO : public fopMAP {
  public:

    float mapping( float, float ) const;

    // Constructors for LMO: parameters are:
    // Time axis, offset axis, slowness value
    // or array
    fopLMO( Axis &, Axis &, float );
    fopLMO( Axis &, Axis &, float a[] );

};


float fopLMO::mapping( float tau, float x ) const {
```

[2]Hector Urdaneta

```
        float slowness = slow[fopMO::inlist[0].index( tau )];
        float t = tau + x * slowness;
        return t;
};
```

NORMAL MOVEOUT

[3] The operator `fopNMO` implements a moveout operation defined by the hyperbolic traveltime curve,

$$t^2 \quad = \quad \tau^2 \ + \ \frac{x^2}{v^2} \tag{1}$$

where τ is traveltime depth. This equation gives either time from a surface source to a receiver at depth τ, or it gives time to a surface receiver from an image source at depth τ.

A seismic trace is a signal $d(t)$ recorded at some constant x. We can convert the trace to a "vertical propagation" signal $m(\tau) = d(t)$ by stretching t to τ. This process is called "normal moveout correction" (NMO).

Typically we have many traces at different x distances each of which theoretically produces the same hypothetical zero-offset trace.

These concepts are implemented in the class `fopNMO`. The header `fopnmo.h` (page 3) shows the public definitions. The operator is constructed from two Axis objects that specify the time and offset axes and a slowness (1/velocity) value. The slowness can either be a constant slowness or an array of slownesses, one for each zero offset time sample.

```
class fopNMO : public fopMAP {

  public:
    float mapping( float tau, float x ) const;
    fopNMO( Axis & t, Axis & x, float slowness    );
    fopNMO( Axis & t, Axis & x, float slowarray[]  );
    fopNMO( Axis & t, Axis & x, Axis & to, Axis & xo, float slowness    );
    fopNMO( Axis & t, Axis & x, Axis & to, Axis & xo, float slowarray[]  );

};
```

The `fopNMO` class that implements NMO is derived from the class `fopMAP` (page ??). The parent class handles the actual stretching of the trace, in the derived class we merely need to supply a virtual mapping function that implements the definition of the NMO transform.

Following the usual convention the forward operator is a modeling operation that models data at finite offset from data at zero-offset time. The adjoint operation takes data at finite offset and applies NMO.

The following figures illustrate, modeling, adjoint processing and inverse processing for NMO.

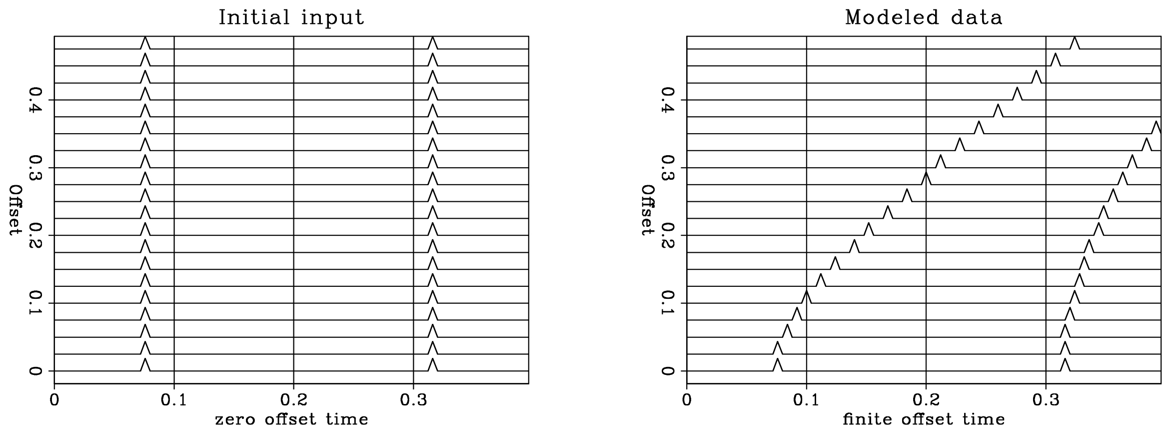

Figure 3: Left frame is input data, two events that are constant on all traces, the time coordinate is t_0, zero offset time. Right frames shows the NMO operator applied forward, the time coordinate is now t, time at each fixed offset. Moveout-nmoout [NR]

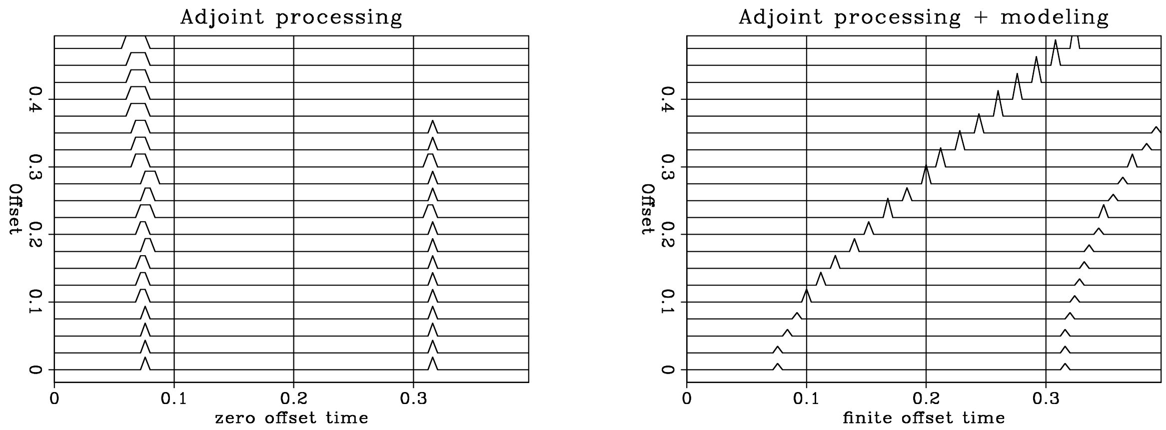

Figure 4: Left frame shows adjoint processing applied to the output of modeling, the time coordinate is t_0, zero-offset time. Right frame shows modeling applied to the adjoint processing, the original data *is not* recovered. Moveout-nmoadj [NR]

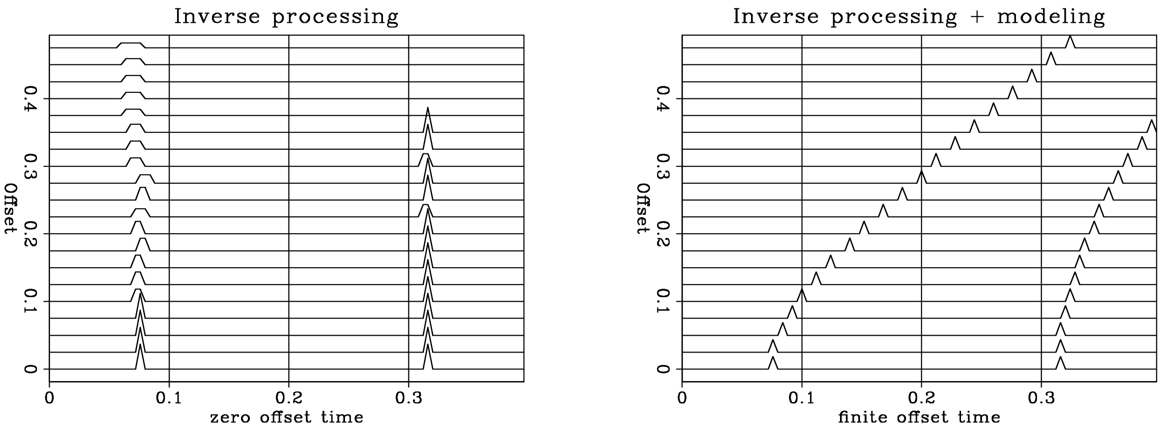

Figure 5: Left frame shows inverse processing applied to the previous figure, the time coordinate is t_0, zero-offset time. This result does not reproduce 3 because some of the original model is in the null space of the operator . Right frame shows modeling applied to the inverse processing, the original data *is* recovered even though the input to this process did not match 3. Moveout-nmoinv [NR]

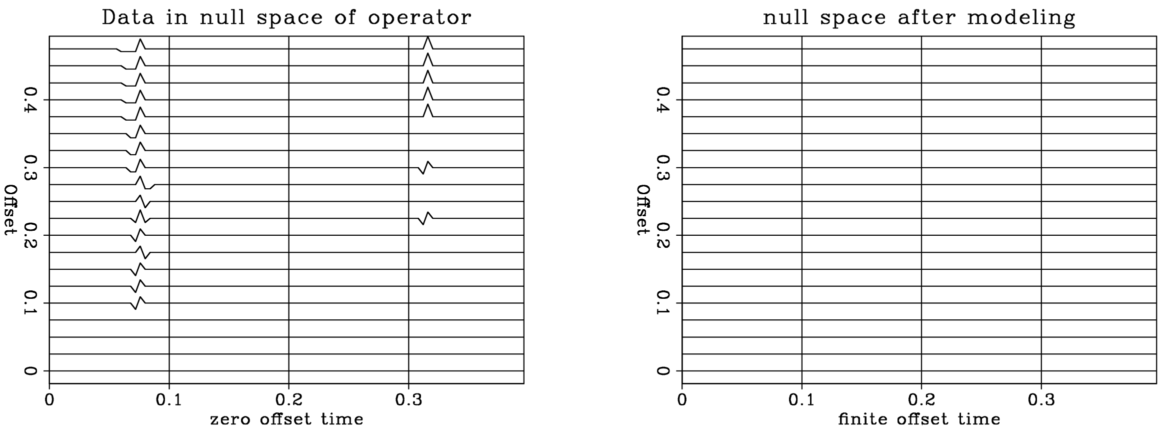

Figure 6: Some data in null space of NMO operator, this is the difference between the input data and the result of the inversion. The right hand frame is the result of applying the forward operator to the data in the null space. The result is zero (as expected). Moveout-nmonull [NR]

ANELLIPTIC MOVEOUT

[4]Normal moveout is based on a horizontal layer model with isotropic velocities in each layer. Nature however can be more complex than that simple model. Each layer can be constituted of many more layers of very small thickness. The resulting velocity then will be angle dependent (anisotropic).

Dix equation is derived from an isotropic layered model and can be extended to anisotropic layers. An anisotropic moveout equation is derived by Dellinger et al. (1993), which adds one more parameter f to the conventional NMO computation. The first-anelliptic processing-NMO equation is:

$$t^2 = \frac{\tau^4 + (f+1)(\tau\ s\ x)^2 + f^2\ (s\ x)^4}{\tau^2 + f(s\ x)^2} \tag{2}$$

where f is a parameter governing anellipticity of the moveout curve. τ is the travel time for the zero offset ray, s denotes the slowness. A practical implementation can be again derived from the moveout operator `fopMAP` by specifying a new anisotropic moveout curve. The moveout then is given as the mapping function in `fopfmo` (page 6).

```
float fopFMO::mapping( float tau, float x ) const {
        float slowness = slow[ fopMO::inlist[0].index( tau )];
        float sx = x * slowness; float sx2 = sx*sx   ; float sx4 = sx2*sx2;
                                 float z2  = tau*tau ; float z4  = z2*z2;
        float t  = sqrt( (z4 + (1.+ffn)*z2 *sx2 + ffn*ffn*sx4)/(z2+ffn*sx2) );
        return t;
    };
```

The anisotropic moveout operator can be called using the following constructors:

```
class fopFMO : public fopMAP {
    public:
    float mapping( float tau, float x ) const;
    fopFMO( Axis & t, Axis & x, float slowness, float ff);
    fopFMO( Axis & t, Axis & x, float slowarray[]  ,float ff );
    fopFMO( Axis & t, Axis & x, Axis & to, Axis & xo, float slowness , float ff);
    fopFMO( Axis & t, Axis & x, Axis & to, Axis & xo, float slowarray[] , float ff);
    }
```

The following Figures 7 and 8 illustrate, modeling, adjoint processing and inverse processing for anelliptic moveout. Especially on the first event in the moved out gather we can see a clear deviation from the normal moveout curve at far offsets.

[3]Dave Nichols, Hector Urdaneta, Martin Karrenbach
[4]Martin Karrenbach

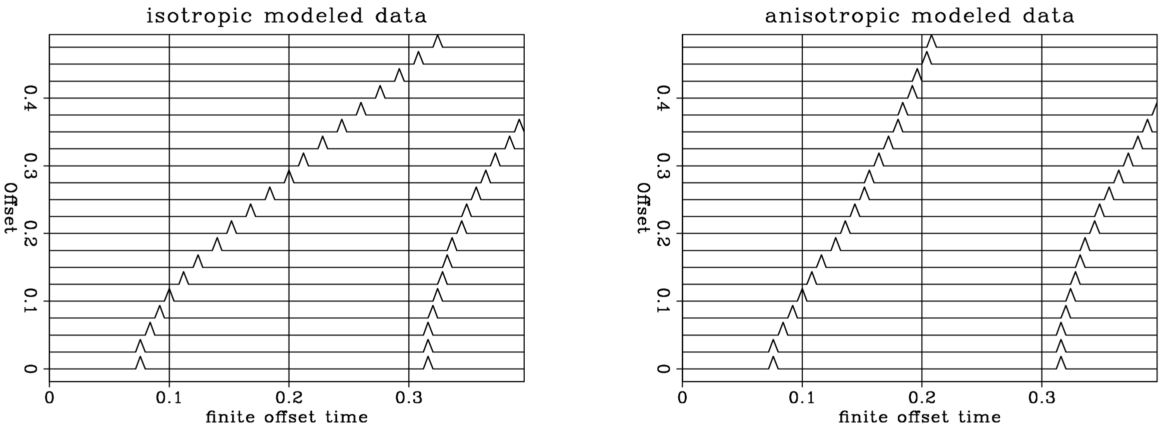

Figure 7: Again the input data are two events that are constant on all traces, the time coordinate is t_0, zero offset time. The left frame shows the isotropic NMO, while the right frame shows the FMO operator applied to the data, the time coordinate is now t, time at each fixed offset. Moveout-fmoout [NR]

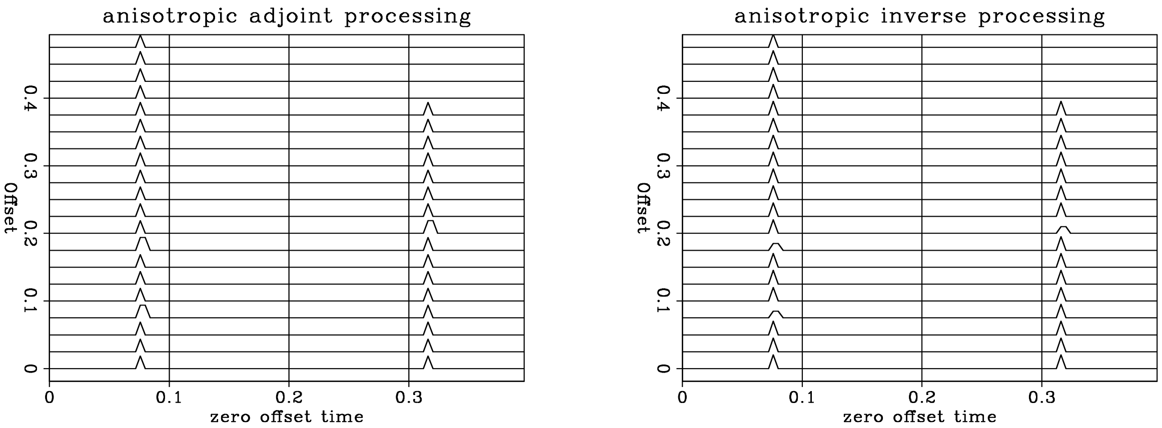

Figure 8: Left frame shows adjoint processing applied to the output of modeling, the time coordinate is t_0, zero offset time. The right frame shows the application of inverse processing to the same data. The differences between adjoint and inverse processing are relatively small for the chosen parameter setting. Moveout-fmoadjinv [NR]

REFERENCES

Dellinger, J., Muir, F., and Karrenbach, M., 1993, Anelliptic approximations for TI media: Journal of Seismic Exploration, **2**, 23–40.

KIRCHHOFF MOVEOUT

[5]This operator implements summing along a trajectory in midpoint-offset space. Modeling or migration of seismic data after stack can be carried out by summing along hyperbolic trajectories. This integral technique (Kirchhoff Modeling/Migration) requires the knowlegde of the trace position in relation to the midpoint (CMP) to be migrated. The moveout curve is given by:

$$t = \sqrt{\frac{\tau^2 + (x-m)^2}{v^2}} \tag{3}$$

where t and τ are two-way travel times with v as half the medium velocity. m is the migrated midpoint position and x denotes the spatial coordinate of the trace to be summed in. We show here only the basic Kirchoff moveout class. Later examples of poststack migration and modeling will be derived from it. The Kirchoff moveout operator is derived here from the operator `fopMAP`, since it has a mapping function which uses a slowness model parameter.

```
float fopKMO::mapping( float tau, float x ) const {
        float twiceslow = slow[ fopMO::inlist[0].index( tau )];
        float hs =  ( x - midpoint ) * twiceslow;
        float  t =  sqrt( tau * tau  +  hs * hs );
        return t;
    };
```

The operator is constructed according to

```
class fopKMO : public fopMAP {
    public:
    float mapping( float tau, float x ) const;
    fopKMO(Axis & t ,Axis & x ,float slowness , float m   );
    fopKMO(Axis & t ,Axis & x ,float *slowarray  , float m );
    float midpoint ;   /* midpoint */
};
```

Figure **??** shows the application of the forward operator to a CMP gather that has horizontal events.

[5]Martin Karrenbach

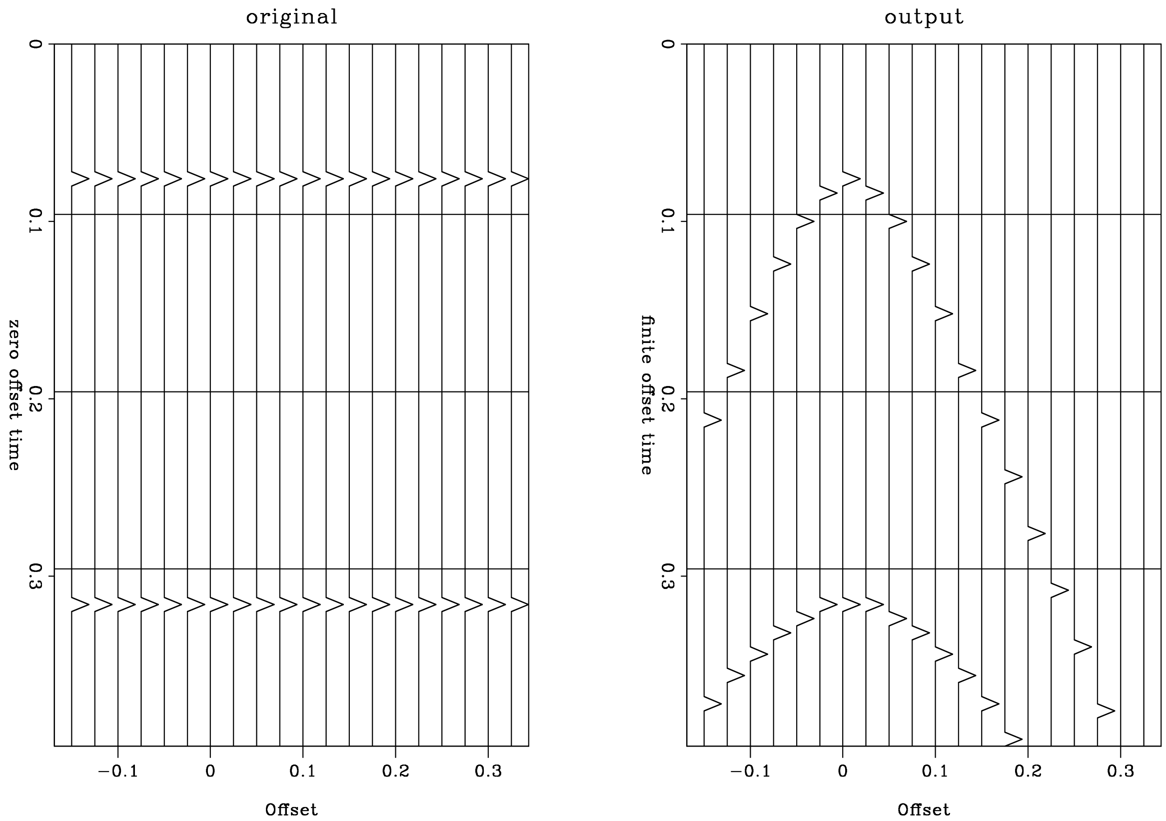

Figure 9: A gather is moved out with the Kirchhoff moveout equation. The CMP midpoint location coincides with the top of the hyperbola. Moveout-kmo [NR]