Wave Inversion Technology, Report No. 4, pages 19-26

Data-driven macro-velocity model – a real data example

I. Koglin, K.-U. Vieth[1]

keywords: *inversion, macro-velocity model, picking, smoothing*

ABSTRACT

A special inversion algorithm makes use of data-derived common-reflection-surface stack attributes to provide a 2-D macro-velocity model of the subsurface. High frequency and low frequency fluctuations of the common-reflection-surface stack attributes along an event require a sophisticated smoothing method. Thus, the inversion performed after the smoothing gets more robust. A data-driven macro-velocity model for a real data set is presented.

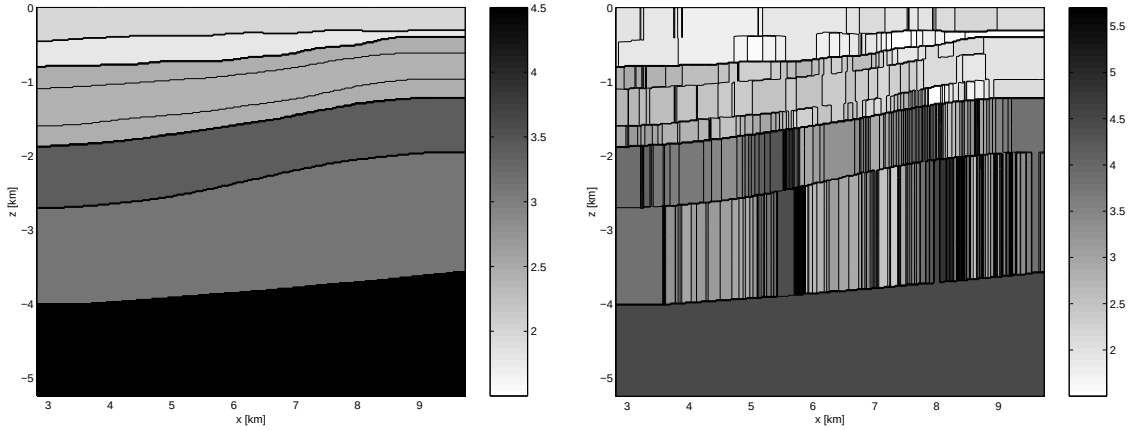
INTRODUCTION

The aim of this work is to get a 2-D layered velocity model of the subsurface which serves, e.g., as macro-velocity model in time or depth migration. We use a special inversion algorithm of (Majer, 2000) which takes as input data-derived common-reflection-surface (CRS) stack attributes (Jäger et al., 2001). Each triplet of CRS stack attributes (α, R_{NIP}, R_N) determines a stacking surface to simulate a zero-offset (ZO) sample for point $P(x_0, t_0)$. Here, x_0 denotes the surface location in terms of the mid-point coordinate x where the normal ray emerges. t_0 is the two-way ZO traveltime. The angle α is the emergence angle of the normal ray measured versus the surface normal. Two theoretical eigenwave experiments (Hubral, 1983) are associated with the radii of curvature R_{NIP} and R_N. R_{NIP} is the radius of wavefront curvature at x_0 originating from a point source at the normal incidence point (NIP). This NIP is the endpoint of the normal ray in the depth domain. Analogously, an exploding reflector experiment in the vicinity of the NIP yields the so-called normal wave emerging with radius R_N at x_0.

[1]**email:** Ingo.Koglin@gpi.uni-karlsruhe.de

INVERSION BY MEANS OF CRS ATTRIBUTES

The inversion algorithm uses the CRS stack attributes to back-propagate the ray associated with these attributes.



(a) (b)

Figure 1: 2-D macro-velocity model derived from a real data-set. (a) shows a velocity model with constant layer velocity. It is the mean velocity that is obtained from many traces and its corresponding attributes associated with the same event. (b) shows layers with laterally inhomogeneous velocities. The half-space beneath the last interface was filled with a constant velocity. The colour corresponds to the layer velocities [km/s].

In addition, picked ZO times divided by two, are needed to find the endpoints of the rays of one event. We use the algorithm for horizon inversion of (Majer, 2000) for continuously layers separated by smooth curved interfaces. The emergence angle α determines the take-off angle of the back-propagated ray. The velocity for each individual ray, j, of the first layer is obtained by $v_j = \frac{R_{NIP,j}}{t_{0,j}/2}$. With this velocity and the known ZO traveltime, the endpoint is determined. All endpoints that correspond to one picked event are used to calculate a smooth interface by means of spline approximation. In this approach, R_N is not used for constructing the interfaces. The curvature of the interfaces is computed with spline approximation. The next layers are obtained by applying the transmission law and Snell's law to the back-propagated rays associated with the next event.

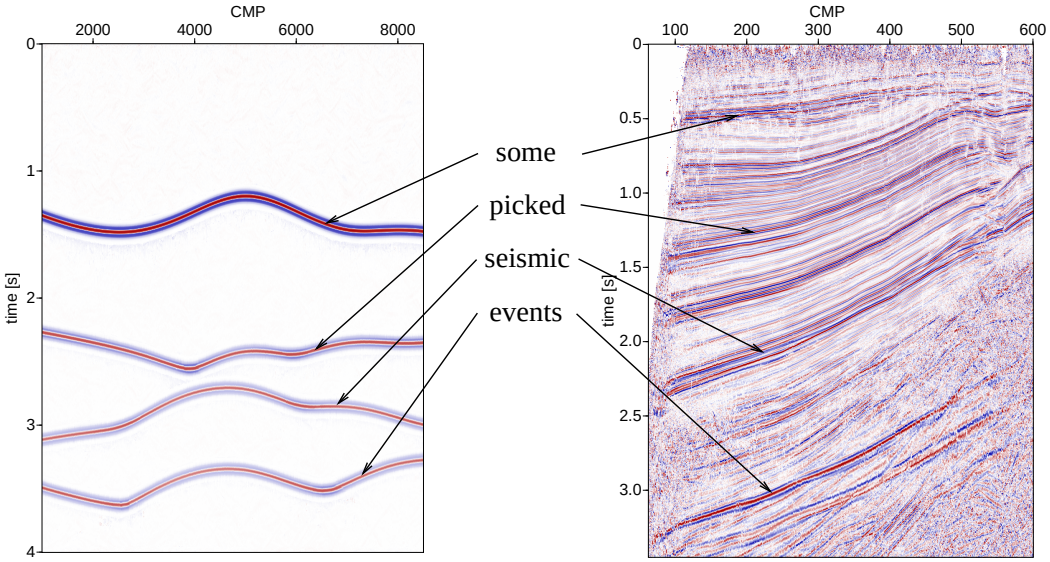
Now, the macro-velocity model can be built up in two alternative ways: Firstly, it can consist of layers with mean constant velocities separated by the smoothed interfaces, see Figure 1 (a). Those mean constant velocities are the arithmetic mean of all interval velocities determined for all ray segments in a layer. Secondly, the layers can have laterally varying velocities. For the first layer, e.g., the velocities are given by $v_j = \frac{R_{NIP,j}}{t_{0,j}/2}$. The layer is then filled with these velocities, see Figure 1 (b). The half-space beneath the last smoothed interface is filled with a constant velocity provided by

the user.

To obtain a macro-velocity model, we have to execute the following four steps:

- Pick seismic events,
- extract the corresponding CRS stack attributes,
- smooth the attributes, and
- perform the inversion.

PICKING OF SEISMIC EVENTS



(a) (b)

Figure 2: Two examples of simulated ZO time sections as a result of the CRS stack: (a) Synthetic data with four interfaces and constant velocity layers. The section was generated by ray tracing using a zero-phase Ricker wavelet with a peak frequency of 15 Hz. (b) Real data scaled with automatic gain control. The arrows indicate some seismic events, which where picked by the picking program we used.

Here, picking means to follow a seismic (primary) reflection event from trace to trace in the time domain. The time samples of one and the same event are found by comparing the phase of adjacent traces. We pick the maximum amplitude because robust CRS stack attributes are found by a coherence analysis that is more reliable at the extrema of the wavelet than at the zero-crossings.

For synthetic data, it is easy to pick these events, see Figure 2 (a). As there are no conflicting events and the signal-to-noise (S/N) ratio is high, the picker follows the same event over the entire ZO section. The synthetic data have been generated using a zero-phase Ricker wavelet with a peak frequency of 15 Hz. Picking the maximum amplitude yields the two-way traveltime from the source to the interface and back to the receiver at the surface. If seismic events in a real data time section, Figure 2 (b), have to be followed continuously, problems occur. Usually, the S/N ratio decreases with increasing traveltime. Events at small traveltimes of the simulated ZO section might be disturbed by mute zones. Regions with conflicting dips or where scattering prevails, interrupt the automatic picking, see Figure 2 (b) in the vicinity of CMP no. 550. Thus, half-automatic picking is requested in order to overcome these problems.

A little error is included while picking the maximum amplitude. In the synthetic and real data-set the actual local maximum cannot always be picked because of the discrete time sampling. But the picked maximum is very close to it, hence, the error is small. The traveltime that corresponds to the maximum amplitude of the wavelet differs slightly from the traveltime that actually represents the spatial location of the impedance contrast in the time domain. To prevent those errors, one suggestion is to pick the time sample where the wavelet starts to separate from the noise but that is even more difficult because one has to define a threshold for the noise.

It is advantageous to pick as much events as possible because the more events are picked the more interfaces can be inverted and the more precise the model will be. Another reason is that the inversion algorithm provides only constant velocities within the layers along the traced rays. Hence, it is preferable to have more picked events with smaller time intervals in order to construct a more complex model.

SMOOTHING BY MEANS OF STATISTICAL METHODS

The inversion algorithm requires smooth CRS stack attributes. The deviation of the angle α from trace to trace along an event is small and, thus, the data of the emergence angles do not require much smoothing. In contrast, the radius of the NIP wavefront R_{NIP} often varies strongly from trace to trace, see Figure 3 (a). To perform a more stable inversion, the data have to be smoothed.

There are several filters available for smoothing purposes. The smoothing is done within a pre-defined window of length $2m + 1$. This yields a symmetrical window around the picked time sample that can be set independently in time and/or trace direction.

Here, possible filter choices are:

1. The arithmetic mean, $\bar{x}$, is the normalised sum of all values $(x_{\tau,\xi})$ within the

pre-defined window. Here, τ denotes the individual values within the window in time direction and ξ is the index for all values of the window in trace direction.

2. The median sorts the data of the window in an increasing order and takes the value in the middle of the sequence.

3. A combination of the arithmetic mean and the median, called 'mean difference cut', calculates at first the arithmetic mean, $\bar{x}$, for the entire window. Then, the deviation of the values from the arithmetic mean is computed. If the deviation is greater than a given percentage, the value is excluded from further calculations. If data remain in this interval, the arithmetic mean is calculated again. In the case that no data are left in this interval, the median is taken as output.

4. The weighted arithmetic mean, $\bar{x}_W$, is also a sum over all values $(x_{\tau,\xi})$ of the pre-defined window. Before the values are summed up, they are multiplied with a triangular weight function.

Robust locally weighted regression

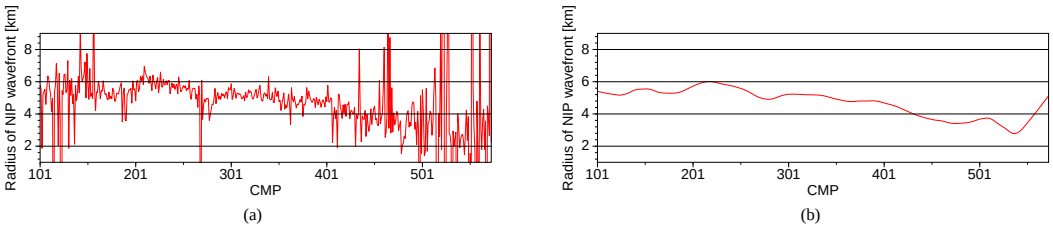


Figure 3: A real data example that has been smoothed with robust locally weighted regression; (a) shows the original R_{NIP} data for one picked event. (b) shows the result of the robust locally weighted regression filter for smoothing. The parameters used for the filter were: `f=0.1`, `nsteps=2`.

A special filter is the robust locally weighted regression of (Cleveland, 1979). The first step is to choose a weight function with the following properties:

- $W(x) > 0$ for $|x| < 1$,
- $W(-x) = W(x)$,
- $W(x)$ is a non-increasing function for $x \geq 0$, and
- $W(x) = 0$ for $|x| \geq 1$.

Examples of such a weight function are a boxcar, a triangle or the cosine function. The second step is to fit a polynomial of d^{th} order to the points (x_i, y_i) within the window using weighted least squares with weights $w_k(x_i)$. $i = 1, \ldots, n$ denotes all points of one event where n is the maximum number of points of that picked event. $k = 1, \ldots, n$ represents the k^{th} weight function corresponding to the i^{th} point. This initial fit, $\hat{y}_i$, is the locally weighted regression. Now, the residual $(y_i - \hat{y}_i)$ is calculated to get a new weight function, δ_i, that has large weights for small residuals and small weights for large residuals. The fitted values are calculated again with a new set of weights, $\delta_i w_k(x_i)$, which are multiplied with the original data. The last step is repeated several times and the result is the robust locally weighted regression. The number of iterations is given by the parameter $nsteps$. The length of the smoothing window is obtained by $r = fn$, where r is rounded to the nearest integer neighbour and f is a factor between zero and one. If f is close to zero, the window length for smoothing is short. Thus, the curve of fitted points is more roughly. If f gets closer to one, more points are considered for the fit. That leads to a smoother curve. This filter was designed to gain the best fit for data for which $y_i = g(x_i) + \epsilon_i$, where g is a smooth function and ϵ_i is a random variable with mean zero and constant scale.

In contrast to the robust locally weighted regression, a spline approximation, a spline interpolation, or a local polynomial regression shows a basic problem. They cannot merge the local parts such that the resulting curve is smooth in first and second order. That means the endpoints and/or their first derivative from one local part to the next must be continuous. To overcome this problem, e.g., a global polynomial regression could be computed for the whole reflector with a certain order. Choosing then a small order yields a polynomial that cannot follow high frequency fluctuations and a large order might lead to a polynomial with more fluctuations than demanded.

The robust locally weighted regression is also a kind of a polynomial fitting procedure.There, the problem of smoothness is solved by improving the weight functions.Hence, the robust locally weighted regression is in favour, see Figure 3 (b).

REAL DATA EXAMPLE

Figure 2 (b) shows a small simulated ZO subset of a real data-set. We had picked ten events between CMP number 150 and 480 of which we used seven to generate the two macro-velocity models in Figure 1 (a) and (b). This CMP window was chosen because there we were able to pick continuous events for times between 0.3 and 3.2 seconds. As mentioned before, some events could not be followed continuously to the left or right because scattering prevails or another event crosses the picked one. We could not pick continuously at smaller times because parts of the events were muted. The arrows in Figure 2 (a) and (b) indicate some picked events used for the inversion. At later times in Figure 2 (b), the noise is too strong to find more continuous events.

Both macro-velocity models or smoothed versions of them can be used for migration. The velocity for the half-space beneath the last interface has to be chosen manually.

CONCLUSIONS

To our knowledge, it is the first time that a data-driven model is presented which is not a result of an iterative improvement of an initial model. The model is calculated directly from the picked data and the model independent CRS stack attributes. We also presented a sophisticated smoothing algorithm, the robust locally weighted regression. This algorithm is suited best to prepare the CRS stack attributes in order to ensure a stable inversion.

REFERENCES

Cleveland, W. S., 1979, Robust locally weighted regression and smoothing scatterplots: Journal of the American Statistical Association, **74**, 829–836.

Hubral, P., 1983, Computing true amplitude reflections in a laterally inhomogeneous earth: Geophysics, **48**, no. 8, 1051–1062.

Jäger, R., Mann, J., Höcht, G., and Hubral, P., 2001, Common-reflection-surface Stack: Image and attributes: Geophysics, **66**, no. 1.

Majer, P., 2000, Inversion of seismic parameters: Determination of the 2-D iso-velocity layer model: Master's thesis, Universität Karlsruhe, Germany.