02_rtm

August 6, 2019

1 02 - Reverse Time Migration

This notebook is the second in a series of tutorial highlighting various aspects of seismic inversion based on Devito operators. In this second example we aim to highlight the core ideas behind seismic inversion, where we create an image of the subsurface from field recorded data. This tutorial follows on the modelling tutorial and will reuse the modelling operator and velocity model.

1.1 Imaging requirement

Seismic imaging relies on two known parameters:

- **Field data** - or also called **recorded data**. This is a shot record corresponding to the true velocity model. In practice this data is acquired as described in the first tutorial. In order to simplify this tutorial we will generate synthetic field data by modelling it with the **true velocity model**.
- **Background velocity model**. This is a velocity model that has been obtained by processing and inverting the field data. We will look at this methods in the following tutorial as it relies on the method we are describing here. This velocity model is usually a **smooth version** of the true velocity model.

1.2 Imaging computational setup

In this tutorial, we will introduce the back-propagation operator. This operator simulates the adjoint wave-equation, that is a wave-equation solved in a reversed time order. This time reversal led to the naming of the method we present here, called Reverse Time Migration. The notion of adjoint in exploration geophysics is fundamental as most of the wave-equation based imaging and inversion methods rely on adjoint based optimization methods.

1.3 Notes on the operators

As we have already described the creation of a forward modelling operator, we will use a thin wrapper function instead. This wrapper is provided by a utility class called `AcousticWaveSolver`, which provides all the necessary operators for seismic modeling, imaging and inversion. The `AcousticWaveSolver` provides a more concise API for common wave propagation operators and caches the Devito `Operator` objects to avoid unnecessary recompilation. Operators introduced for the first time in this tutorial will be properly described.

As before we initialize printing and import some utilities. We also raise the Devito log level to avoid excessive logging for repeated operator invocations.

[4]:
```
import numpy as np
%matplotlib inline

from devito import configuration
configuration['log-level'] = 'WARNING'
```

1.4 Computational considerations

Seismic inversion algorithms are generally very computationally demanding and require a large amount of memory to store the forward wavefield. In order to keep this tutorial as lightweight as possible we are using a very simple velocity model that requires low temporal and spatial resolution. For a more realistic model, a second set of preset parameters for a reduced version of the 2D Marmousi data set [1] is provided below in comments. This can be run to create some more realistic subsurface images. However, this second preset is more computationally demanding and requires a slightly more powerful workstation.

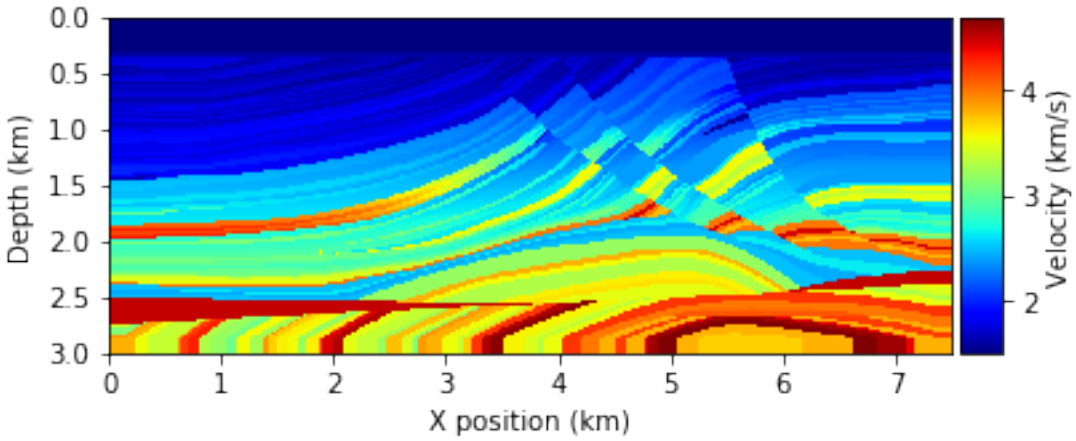
[7]:
```
# Configure model presets
from examples.seismic import demo_model

# Enable model presets here:
#preset = 'twolayer-isotropic'  # A simple but cheap model (recommended)
preset = 'marmousi2d-isotropic'  # A larger more realistic model

# Standard preset with a simple two-layer model
if preset == 'twolayer-isotropic':
    def create_model(grid=None):
        return demo_model('twolayer-isotropic', origin=(0., 0.), shape=(101,␣
 ↪101),
                          spacing=(10., 10.), nbpml=20, grid=grid, ratio=2)
    filter_sigma = (1, 1)
    nshots = 21
    nreceivers = 101
    t0 = 0.
    tn = 1000.  # Simulation last 1 second (1000 ms)
    f0 = 0.010  # Source peak frequency is 10Hz (0.010 kHz)


# A more computationally demanding preset based on the 2D Marmousi model
if preset == 'marmousi2d-isotropic':
    def create_model(grid=None):
        return demo_model('marmousi2d-isotropic', data_path='data/',
                          grid=grid, nbpml=20)
    filter_sigma = (6, 6)
    nshots = 301  # Need good covergae in shots, one every two grid points
    nreceivers = 601  # One recevier every grid point
    t0 = 0.
    tn = 3500.  # Simulation last 3.5 second (3500 ms)
```

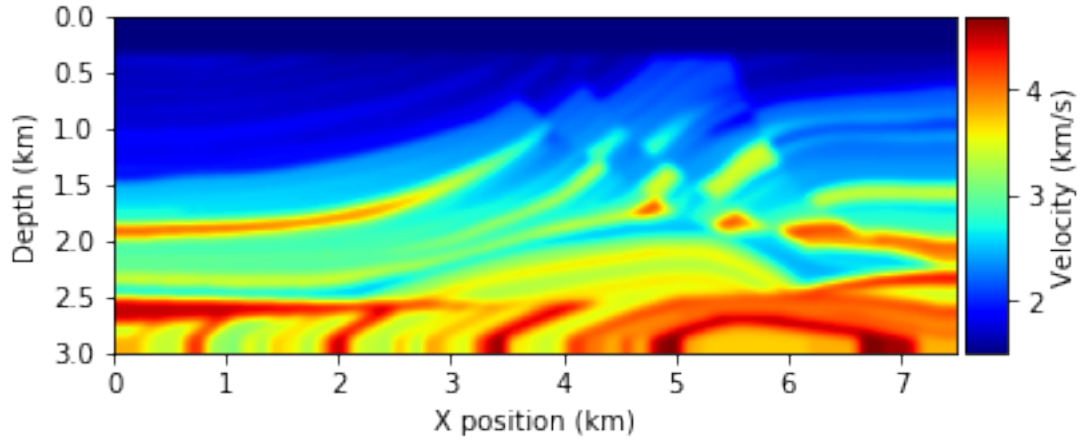
```
    f0 = 0.025  # Source peak frequency is 25Hz (0.025 kHz)
```

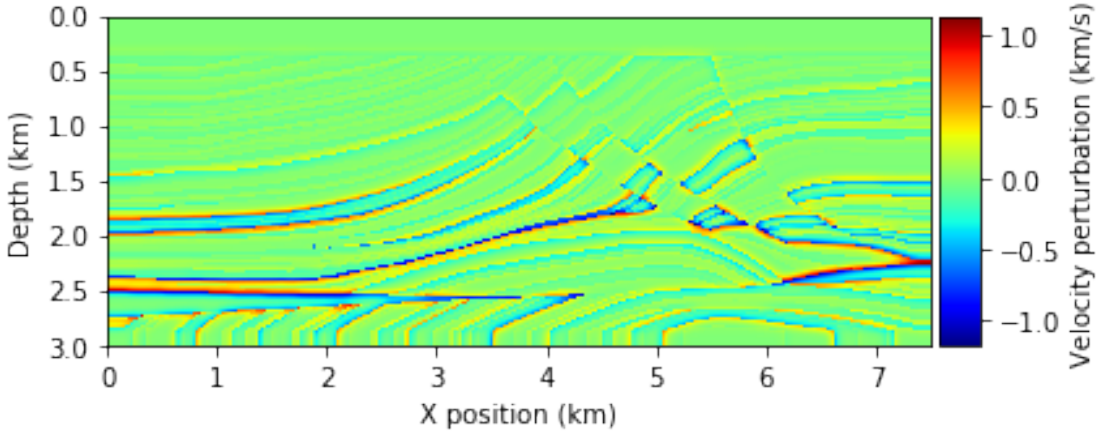
2 True and smooth velocity models

First, we create the model data for the "true" model from a given demonstration preset. This model represents the subsurface topology for the purposes of this example and we will later use it to generate our synthetic data readings. We also generate a second model and apply a smoothing filter to it, which represents our initial model for the imaging algorithm. The perturbation between these two models can be thought of as the image we are trying to recover.

[8]:
```
#NBVAL_IGNORE_OUTPUT
from examples.seismic import plot_velocity, plot_perturbation
from scipy import ndimage

# Create true model from a preset
model = create_model()

# Create initial model and smooth the boundaries
model0 = create_model(grid=model.grid)
model0.vp = ndimage.gaussian_filter(model0.vp.data, sigma=filter_sigma, order=0)

# Plot the true and initial model and the perturbation between them
plot_velocity(model)
plot_velocity(model0)
plot_perturbation(model0, model)
```





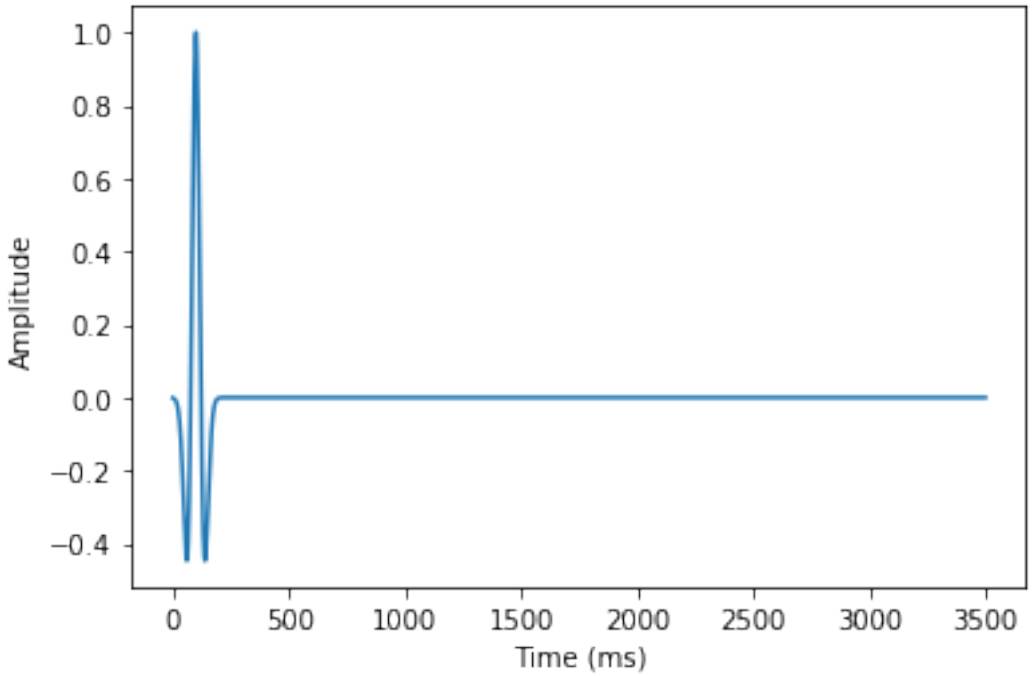


2.1 Acquisition geometry

Next we define the positioning and the wave signal of our source, as well as the location of our receivers. To generate the wavelet for our source we require the discretized values of time that we are going to use to model a single "shot", which again depends on the grid spacing used in our model. For consistency this initial setup will look exactly as in the previous modelling tutorial, although we will vary the position of our source later on during the actual imaging algorithm.

[9]:
```
#NBVAL_IGNORE_OUTPUT
# Define acquisition geometry: source
from examples.seismic import AcquisitionGeometry

# First, position source centrally in all dimensions, then set depth
src_coordinates = np.empty((1, 2))
src_coordinates[0, :] = np.array(model.domain_size) * .5
```

```
src_coordinates[0, -1] = 20.  # Depth is 20m


# Define acquisition geometry: receivers

# Initialize receivers for synthetic and imaging data
rec_coordinates = np.empty((nreceivers, 2))
rec_coordinates[:, 0] = np.linspace(0, model.domain_size[0], num=nreceivers)
rec_coordinates[:, 1] = 30.

# Geometry

geometry = AcquisitionGeometry(model, rec_coordinates, src_coordinates, t0, tn,␣
 ↪f0=.010, src_type='Ricker')
# We can plot the time signature to see the wavelet
geometry.src.show()
```



3 True and smooth data

We can now generate the shot record (receiver readings) corresponding to our true and initial models. The difference between these two records will be the basis of the imaging procedure.

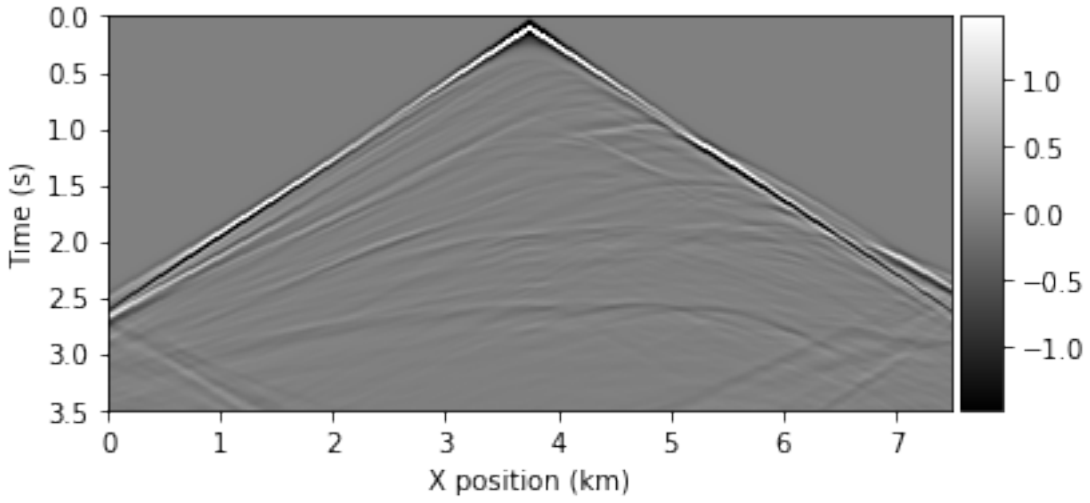
For this purpose we will use the same forward modelling operator that was introduced in the previous tutorial, provided by the `AcousticWaveSolver` utility class. This object instantiates a set

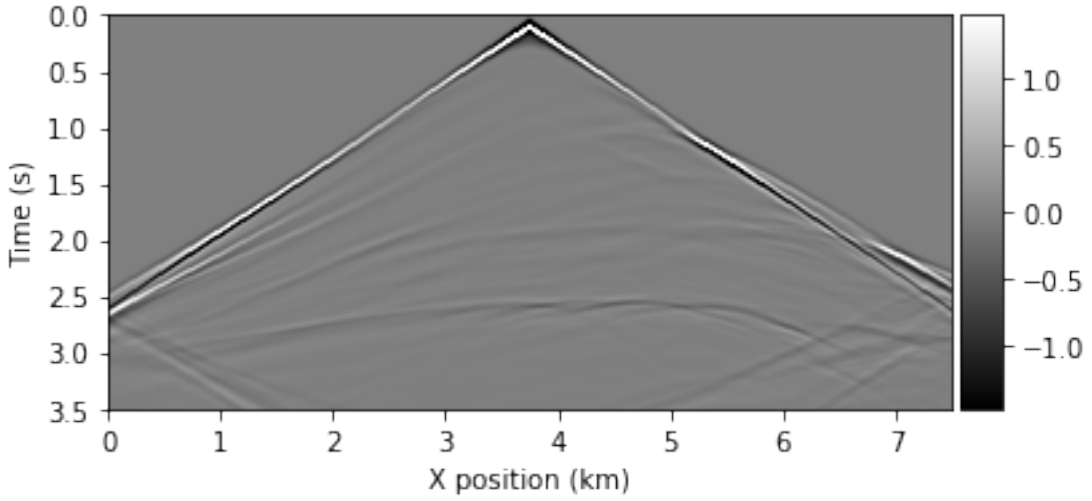
of pre-defined operators according to an initial definition of the acquisition geometry, consisting of source and receiver symbols. The solver objects caches the individual operators and provides a slightly more high-level API that allows us to invoke the modelling modelling operators from the initial tutorial in a single line. In the following cells we use this to generate shot data by only specifying the respective model symbol `m` to use, and the solver will create and return a new `Receiver` object the represents the readings at the previously defined receiver coordinates.

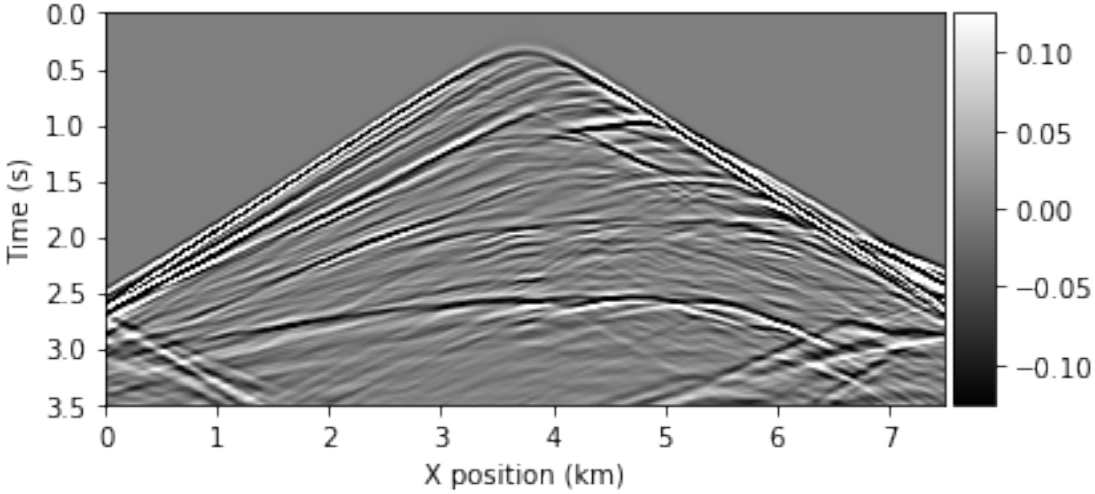
[10]:
```
# Compute synthetic data with forward operator
from examples.seismic.acoustic import AcousticWaveSolver

solver = AcousticWaveSolver(model, geometry, space_order=4)
true_d , _, _ = solver.forward(vp=model.vp)
```

[11]:
```
# Compute initial data with forward operator
smooth_d, _, _ = solver.forward(vp=model0.vp)
```

[13]:
```
#NBVAL_IGNORE_OUTPUT
# Plot shot record for true and smooth velocity model and the difference
from examples.seismic import plot_shotrecord

plot_shotrecord(true_d.data, model, t0, tn)
plot_shotrecord(smooth_d.data, model, t0, tn)
plot_shotrecord(smooth_d.data - true_d.data, model, t0, tn)
```







4 Imaging with back-propagation

As explained in the introduction of this tutorial, this method is based on back-propagation.

4.1 Adjoint wave equation

If we go back to the modelling part, we can rewrite the simulation as a linear system solve:

$$\mathbf{A}(\mathbf{m})\mathbf{u} = \mathbf{q} \tag{1}$$

where $\mathbf{m}$ is the discretized square slowness, $\mathbf{q}$ is the discretized source and $\mathbf{A}(\mathbf{m})$ is the discretized wave-equation. The discretized wave-equation matricial representation is a lower triangular matrix that can be solve with forward substitution. The pointwise writing or the forward substitution leads to the time-stepping stencil.

On a small problem one could form the matrix explicitly and transpose it to obtain the adjoint discrete wave-equation:

$$\mathbf{A}(\mathbf{m})^T\mathbf{v} = \delta\mathbf{d} \tag{2}$$

where $\mathbf{v}$ is the discrete **adjoint wavefield** and $\delta\mathbf{d}$ is the data residual defined as the difference between the field/observed data and the synthetic data $\mathbf{d}_s = \mathbf{P}_r\mathbf{u}$. In our case we derive the discrete adjoint wave-equation from the discrete forward wave-equation to get its stencil.

4.2 Imaging

Wave-equation based imaging relies on one simple concept:

- If the background velocity model is cinematically correct, the forward wavefield $\mathbf{u}$ and the adjoint wavefield $\mathbf{v}$ meet at the reflectors position at zero time offset.

The sum over time of the zero time-offset correlation of these two fields then creates an image of the subsurface. Mathematically this leads to the simple imaging condition:

$$\text{Image} = \sum_{t=1}^{n_t} \mathbf{u}[t]\mathbf{v}[t] \tag{3}$$

In the following tutorials we will describe a more advanced imaging condition that produces shaper and more accurate results.

4.3 Operator

We will now define the imaging operator that computes the adjoint wavefield $\mathbf{v}$ and correlates it with the forward wavefield $\mathbf{u}$. This operator essentially consists of three components: * Stencil update of the adjoint wavefield `v` * Injection of the data residual at the adjoint source (forward receiver) location * Correlation of `u` and `v` to compute the image contribution at each timestep

[14]:
```
# Define gradient operator for imaging
from devito import TimeFunction, Operator, Eq, solve
from examples.seismic import PointSource

def ImagingOperator(model, image):
    # Define the wavefield with the size of the model and the time dimension
    v = TimeFunction(name='v', grid=model.grid, time_order=2, space_order=4)

    u = TimeFunction(name='u', grid=model.grid, time_order=2, space_order=4,
                     save=geometry.nt)

    # Define the wave equation, but with a negated damping term
    eqn = model.m * v.dt2 - v.laplace - model.damp * v.dt

    # Use `solve` to rearrange the equation into a stencil expression
```

```
    stencil = Eq(v.backward, solve(eqn, v.backward))

    # Define residual injection at the location of the forward receivers
    dt = model.critical_dt
    residual = PointSource(name='residual', grid=model.grid,
                           time_range=geometry.time_axis,
                           coordinates=geometry.rec_positions)
    res_term = residual.inject(field=v, expr=residual * dt**2 / model.m)

    # Correlate u and v for the current time step and add it to the image
    image_update = Eq(image, image - u * v)

    return Operator([stencil] + res_term + [image_update],
                    subs=model.spacing_map)
```

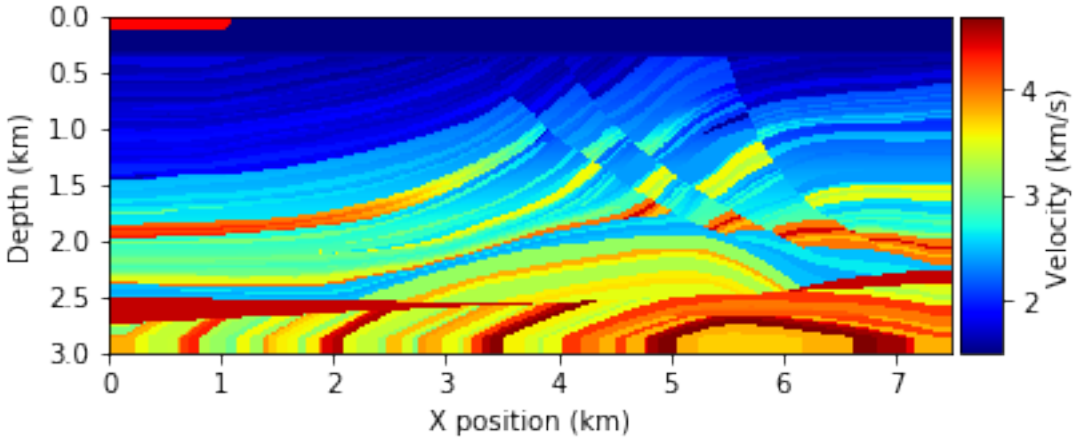
4.4 Implementation of the imaging loop

As just explained, the forward wave-equation is solved forward in time while the adjoint wave-equation is solved in a reversed time order. Therefore, the correlation of these two fields over time requires to store one of the two fields. The computational procedure for imaging follows:

- Simulate the forward wave-equation with the background velocity model to get the synthetic data and save the full wavefield **u**
- Compute the data residual
- Back-propagate the data residual and compute on the fly the image contribution at each time step.

This procedure is applied to multiple source positions (shots) and summed to obtain the full image of the subsurface. We can first visualize the varying locations of the sources that we will use.

[18]:
```
#NBVAL_IGNORE_OUTPUT

# Prepare the varying source locations
source_locations = np.empty((nshots, 2), dtype=np.float32)
source_locations[:, 0] = np.linspace(0., 1000, num=nshots)
source_locations[:, 1] = 30.

plot_velocity(model, source=source_locations)
```



[19]:
```
# Run imaging loop over shots
from devito import Function, clear_cache

# Create image symbol and instantiate the previously defined imaging operator
image = Function(name='image', grid=model.grid)
op_imaging = ImagingOperator(model, image)

# Create a wavefield for saving to avoid memory overload
u0 = TimeFunction(name='u', grid=model0.grid, time_order=2, space_order=4,
                  save=geometry.nt)

for i in range(nshots):
    # Important: We force previous wavefields to be destroyed,
    # so that we may reuse the memory.
    clear_cache()

    print('Imaging source %d out of %d' % (i+1, nshots))

    # Update source location
    geometry.src_positions[0, :] = source_locations[i, :]

    # Generate synthetic data from true model
    true_d, _, _ = solver.forward(vp=model.vp)

    # Compute smooth data and full forward wavefield u0
    u0.data.fill(0.)
    smooth_d, _, _ = solver.forward(vp=model0.vp, save=True, u=u0)

    # Compute gradient from the data residual
    v = TimeFunction(name='v', grid=model.grid, time_order=2, space_order=4)
    residual = smooth_d.data - true_d.data
    op_imaging(u=u0, v=v, vp=model0.vp, dt=model0.critical_dt,
```

```
                residual=residual)
```

```
Imaging source 1 out of 301
Imaging source 2 out of 301
Imaging source 3 out of 301
Imaging source 4 out of 301
Imaging source 5 out of 301
Imaging source 6 out of 301
Imaging source 7 out of 301
Imaging source 8 out of 301
Imaging source 9 out of 301
Imaging source 10 out of 301
Imaging source 11 out of 301
Imaging source 12 out of 301
Imaging source 13 out of 301
Imaging source 14 out of 301
Imaging source 15 out of 301
Imaging source 16 out of 301
Imaging source 17 out of 301
Imaging source 18 out of 301
Imaging source 19 out of 301
Imaging source 20 out of 301
Imaging source 21 out of 301
Imaging source 22 out of 301
Imaging source 23 out of 301
Imaging source 24 out of 301
Imaging source 25 out of 301
Imaging source 26 out of 301
Imaging source 27 out of 301
Imaging source 28 out of 301
Imaging source 29 out of 301
Imaging source 30 out of 301
Imaging source 31 out of 301
Imaging source 32 out of 301
Imaging source 33 out of 301
Imaging source 34 out of 301
Imaging source 35 out of 301
Imaging source 36 out of 301
Imaging source 37 out of 301
Imaging source 38 out of 301
Imaging source 39 out of 301
Imaging source 40 out of 301
Imaging source 41 out of 301
Imaging source 42 out of 301
Imaging source 43 out of 301
Imaging source 44 out of 301
Imaging source 45 out of 301
```

```
Imaging source 46 out of 301
Imaging source 47 out of 301
Imaging source 48 out of 301
Imaging source 49 out of 301
Imaging source 50 out of 301
Imaging source 51 out of 301
Imaging source 52 out of 301
Imaging source 53 out of 301
Imaging source 54 out of 301
Imaging source 55 out of 301
Imaging source 56 out of 301
Imaging source 57 out of 301
Imaging source 58 out of 301
Imaging source 59 out of 301
Imaging source 60 out of 301
Imaging source 61 out of 301
Imaging source 62 out of 301
Imaging source 63 out of 301
Imaging source 64 out of 301
Imaging source 65 out of 301
Imaging source 66 out of 301
Imaging source 67 out of 301
Imaging source 68 out of 301
Imaging source 69 out of 301
Imaging source 70 out of 301
Imaging source 71 out of 301
Imaging source 72 out of 301
Imaging source 73 out of 301
Imaging source 74 out of 301
Imaging source 75 out of 301
Imaging source 76 out of 301
Imaging source 77 out of 301
Imaging source 78 out of 301
Imaging source 79 out of 301
Imaging source 80 out of 301
Imaging source 81 out of 301
Imaging source 82 out of 301
Imaging source 83 out of 301
Imaging source 84 out of 301
Imaging source 85 out of 301
Imaging source 86 out of 301
Imaging source 87 out of 301
Imaging source 88 out of 301
Imaging source 89 out of 301
Imaging source 90 out of 301
Imaging source 91 out of 301
Imaging source 92 out of 301
Imaging source 93 out of 301
```

```
Imaging source 94 out of 301
Imaging source 95 out of 301
Imaging source 96 out of 301
Imaging source 97 out of 301
Imaging source 98 out of 301
Imaging source 99 out of 301
Imaging source 100 out of 301
Imaging source 101 out of 301
Imaging source 102 out of 301
Imaging source 103 out of 301
Imaging source 104 out of 301
Imaging source 105 out of 301
Imaging source 106 out of 301
Imaging source 107 out of 301
Imaging source 108 out of 301
Imaging source 109 out of 301
Imaging source 110 out of 301
Imaging source 111 out of 301
Imaging source 112 out of 301
Imaging source 113 out of 301
Imaging source 114 out of 301
Imaging source 115 out of 301
Imaging source 116 out of 301
Imaging source 117 out of 301
Imaging source 118 out of 301
Imaging source 119 out of 301
Imaging source 120 out of 301
Imaging source 121 out of 301
Imaging source 122 out of 301
Imaging source 123 out of 301
Imaging source 124 out of 301
Imaging source 125 out of 301
Imaging source 126 out of 301
Imaging source 127 out of 301
Imaging source 128 out of 301
Imaging source 129 out of 301
Imaging source 130 out of 301
Imaging source 131 out of 301
Imaging source 132 out of 301
Imaging source 133 out of 301
Imaging source 134 out of 301
Imaging source 135 out of 301
Imaging source 136 out of 301
Imaging source 137 out of 301
Imaging source 138 out of 301
Imaging source 139 out of 301
Imaging source 140 out of 301
Imaging source 141 out of 301
```

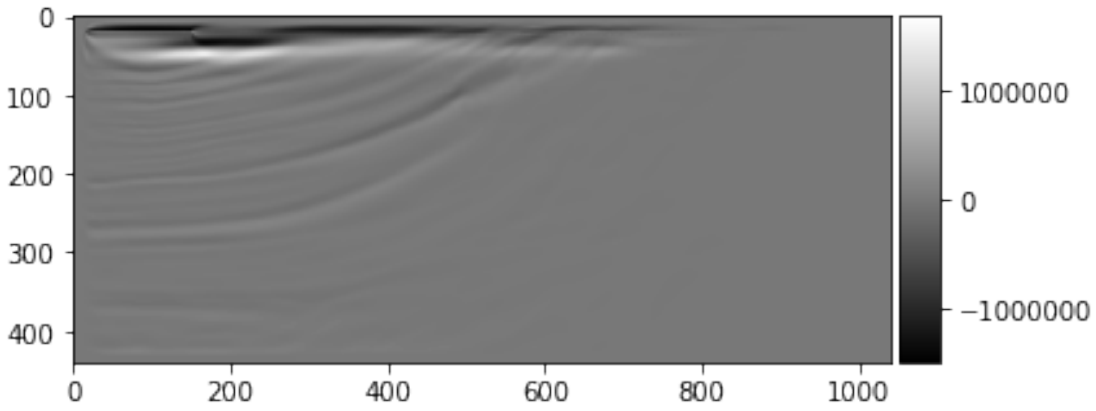
```
Imaging source 142 out of 301
Imaging source 143 out of 301
Imaging source 144 out of 301
Imaging source 145 out of 301
Imaging source 146 out of 301
Imaging source 147 out of 301
Imaging source 148 out of 301
Imaging source 149 out of 301
Imaging source 150 out of 301
Imaging source 151 out of 301
Imaging source 152 out of 301
Imaging source 153 out of 301
Imaging source 154 out of 301
Imaging source 155 out of 301
Imaging source 156 out of 301
Imaging source 157 out of 301
Imaging source 158 out of 301
Imaging source 159 out of 301
Imaging source 160 out of 301
Imaging source 161 out of 301
Imaging source 162 out of 301
Imaging source 163 out of 301
Imaging source 164 out of 301
Imaging source 165 out of 301
Imaging source 166 out of 301
Imaging source 167 out of 301
Imaging source 168 out of 301
Imaging source 169 out of 301
Imaging source 170 out of 301
Imaging source 171 out of 301
Imaging source 172 out of 301
Imaging source 173 out of 301
Imaging source 174 out of 301
Imaging source 175 out of 301
Imaging source 176 out of 301
Imaging source 177 out of 301
Imaging source 178 out of 301
Imaging source 179 out of 301
Imaging source 180 out of 301
Imaging source 181 out of 301
Imaging source 182 out of 301
Imaging source 183 out of 301
Imaging source 184 out of 301
Imaging source 185 out of 301
Imaging source 186 out of 301
Imaging source 187 out of 301
Imaging source 188 out of 301
Imaging source 189 out of 301
```

```
Imaging source 190 out of 301
Imaging source 191 out of 301
Imaging source 192 out of 301
Imaging source 193 out of 301
Imaging source 194 out of 301
Imaging source 195 out of 301
Imaging source 196 out of 301
Imaging source 197 out of 301
Imaging source 198 out of 301
Imaging source 199 out of 301
Imaging source 200 out of 301
Imaging source 201 out of 301
Imaging source 202 out of 301
Imaging source 203 out of 301
Imaging source 204 out of 301
Imaging source 205 out of 301
Imaging source 206 out of 301
Imaging source 207 out of 301
Imaging source 208 out of 301
Imaging source 209 out of 301
Imaging source 210 out of 301
Imaging source 211 out of 301
Imaging source 212 out of 301
Imaging source 213 out of 301
Imaging source 214 out of 301
Imaging source 215 out of 301
Imaging source 216 out of 301
Imaging source 217 out of 301
Imaging source 218 out of 301
Imaging source 219 out of 301
Imaging source 220 out of 301
Imaging source 221 out of 301
Imaging source 222 out of 301
Imaging source 223 out of 301
Imaging source 224 out of 301
Imaging source 225 out of 301
Imaging source 226 out of 301
Imaging source 227 out of 301
Imaging source 228 out of 301
Imaging source 229 out of 301
Imaging source 230 out of 301
Imaging source 231 out of 301
Imaging source 232 out of 301
Imaging source 233 out of 301
Imaging source 234 out of 301
Imaging source 235 out of 301
Imaging source 236 out of 301
Imaging source 237 out of 301
```

```
Imaging source 238 out of 301
Imaging source 239 out of 301
Imaging source 240 out of 301
Imaging source 241 out of 301
Imaging source 242 out of 301
Imaging source 243 out of 301
Imaging source 244 out of 301
Imaging source 245 out of 301
Imaging source 246 out of 301
Imaging source 247 out of 301
Imaging source 248 out of 301
Imaging source 249 out of 301
Imaging source 250 out of 301
Imaging source 251 out of 301
Imaging source 252 out of 301
Imaging source 253 out of 301
Imaging source 254 out of 301
Imaging source 255 out of 301
Imaging source 256 out of 301
Imaging source 257 out of 301
Imaging source 258 out of 301
Imaging source 259 out of 301
Imaging source 260 out of 301
Imaging source 261 out of 301
Imaging source 262 out of 301
Imaging source 263 out of 301
Imaging source 264 out of 301
Imaging source 265 out of 301
Imaging source 266 out of 301
Imaging source 267 out of 301
Imaging source 268 out of 301
Imaging source 269 out of 301
Imaging source 270 out of 301
Imaging source 271 out of 301
Imaging source 272 out of 301
Imaging source 273 out of 301
Imaging source 274 out of 301
Imaging source 275 out of 301
Imaging source 276 out of 301
Imaging source 277 out of 301
Imaging source 278 out of 301
Imaging source 279 out of 301
Imaging source 280 out of 301
Imaging source 281 out of 301
Imaging source 282 out of 301
Imaging source 283 out of 301
Imaging source 284 out of 301
Imaging source 285 out of 301
```

```
Imaging source 286 out of 301
Imaging source 287 out of 301
Imaging source 288 out of 301
Imaging source 289 out of 301
Imaging source 290 out of 301
Imaging source 291 out of 301
Imaging source 292 out of 301
Imaging source 293 out of 301
Imaging source 294 out of 301
Imaging source 295 out of 301
Imaging source 296 out of 301
Imaging source 297 out of 301
Imaging source 298 out of 301
Imaging source 299 out of 301
Imaging source 300 out of 301
Imaging source 301 out of 301
```

[20]:
```
#NBVAL_IGNORE_OUTPUT
from examples.seismic import plot_image

# Plot the inverted image
plot_image(np.diff(image.data, axis=1))
```



And we have an image of the subsurface with a strong reflector at the original location.

4.5 References

[1] *Versteeg, R.J. & Grau, G. (eds.) (1991): The Marmousi experience. Proc. EAGE workshop on Practical Aspects of Seismic Data Inversion (Copenhagen, 1990), Eur. Assoc. Explor. Geophysicists, Zeist.*