Full-waveform inversion, Part 2: Adjoint modeling

Mathias Louboutin[1], Philipp Witte[1], Michael Lange[2], Navjot Kukreja[2], Fabio Luporini[2], Gerard Gorman[2], and Felix J. Herrmann[1,3]

Introduction

This is the second part of a three-part tutorial series on full-waveform inversion (FWI) in which we provide a step-by-step walk through of setting up forward and adjoint wave equation solvers and an optimization framework for inversion. In Part 1 (Louboutin et al., 2017), we showed how to use Devito (http://www.opesci.org/devito-public) to set up and solve acoustic wave equations with (impulsive) seismic sources and sample wavefields at the receiver locations to forward model shot records. Here in Part 2, we will discuss how to set up and solve adjoint wave equations with Devito and, from that, how we can calculate gradients and function values of the FWI objective function.

The gradient of FWI is most commonly computed via the adjoint-state method, by crosscorrelating forward and adjoint wavefields and summing the contributions over all time steps (Plessix, 2006). Calculating the gradient for one source location consists of three steps:

1) Solve the forward wave equation to create a shot record. The time varying wavefield must be stored for use in step 3; techniques such as subsampling can be used to reduce the storage requirements.
2) Compute the data residual (or misfit) between the predicted and observed data.
3) Solve the corresponding discrete adjoint model using the data residual as the source. Within the adjoint (reverse) time loop, crosscorrelate the second time derivative of the adjoint wavefield with the forward wavefield. These crosscorrelations are summed to form the gradient.

We start with the definition and derivation of the adjoint wave equation and its Devito stencil and then show how to compute the gradient of the conventional least-squares FWI misfit function. As usual, this tutorial is accompanied by all the code you need to reproduce the figures. Go to github.com/seg/tutorials-2018 and follow the links.

A simple experiment

To demonstrate the gradient computation in the simplest possible way, we perform a small seismic transmission experiment with a circular imaging phantom, i.e., a constant velocity model with a circular high-velocity inclusion in its center, as shown in Figure 1. For a transmission experiment, we place 21 seismic sources on the left-hand side of the model and 101 receivers on the right-hand side.

We will use the forward propagator from Part 1 to independently model the 21 "observed" shot records using the true model. As the initial model for our gradient calculation, we use a constant velocity model with the same velocity as the true model, but without the circular velocity perturbation. We will then model the 21 predicted shot records for the initial model, calculate the data residual and gradient for each shot, and sum them to obtain the full gradient.

The adjoint wave equation

Adjoint wave equations are a main component in seismic inversion algorithms and are required for computing gradients of both linear and nonlinear objective functions. To ensure stability of the adjoint modeling scheme and the expected convergence of inversion algorithms, it is important that the adjoint wave equation is in fact the adjoint (transpose) of the forward wave equation. The derivation of the adjoint wave equation in the acoustic case is simple, as it is self-adjoint if we ignore the absorbing boundaries

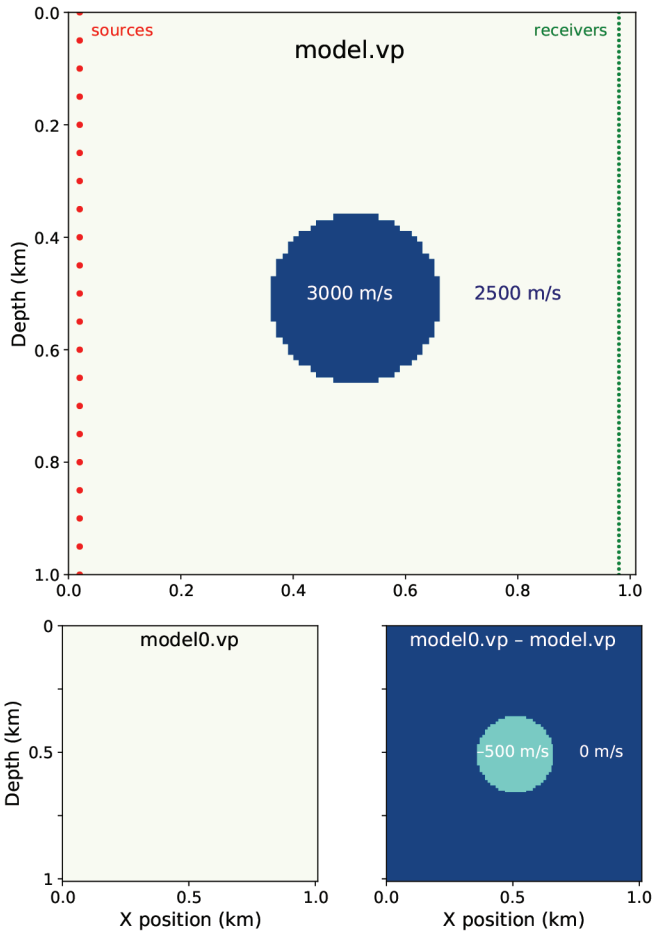

Figure 1. (a) The velocity model, with sources and receivers arranged vertically. (b) The initial estimate. (c) The difference between the model and the initial estimate.

[1]The University of British Columbia, Seismic Laboratory for Imaging and Modeling (SLIM).
[2]Imperial College London.
[3]Georgia Institute of Technology.

https://doi.org/10.1190/tle37010069.1.

for the moment. However, in the general case, discrete wave equations do not have this property (such as the coupled anisotropic TTI wave equation [Zhang et al., 2011]) and require correct derivations of their adjoints. We concentrate here, as in Part 1, on the acoustic case and follow an optimize-discretize approach, which means we write out the adjoint wave equation for the continuous case first and then discretize it, using finite-difference operators of the same order as for the forward equation. With the variables defined as in Part 1 and the data residual $\delta d(x, y, t; x_r, y_r)$, located at x_r, y_r (receiver locations) as the adjoint source, the continuous adjoint wave equation is given by:

$$m(x,y)\frac{\mathrm{d}^2 v(t,x,y)}{\mathrm{d}t^2} - \Delta v(t,x,y) - H(t,x,y) = \delta d(t,x,y;x_r,y_r). \quad (1)$$

The adjoint acoustic wave equation is equivalent to the forward equation with the exception of the damping term $H(t, x, y) = \eta(x, y)\,\mathrm{d}v(t, x, y)/\mathrm{d}t$, which contains a first time derivative and therefore has a change of sign in its adjoint. (A second derivative matrix is the same as its transpose, whereas a first derivative matrix is equal to its negative transpose and vice versa.)

Following the pattern of Part 1, we first define the discrete adjoint wavefield **v** as a Devito `TimeFunction` object. For reasons we will explain later, we do not need to save the adjoint wavefield:

```
v = TimeFunction(name="v", grid=model.grid,
                 time_order=2, space_order=4,
                 save=False)
```

Now symbolically set up the PDE:

```
pde = model.m * v.dt2 - v.laplace - model.damp * v.dt
```

As before, we then define a stencil:

```
stencil_v = Eq(v.backward, solve(pde, v.backward)[0])
```

Just as for the forward wave equation, `stencil_v` defines the update for the adjoint wavefield of a single time step. The only difference is that, while the forward-modeling propagator goes forward in time, the adjoint propagator goes backward in time, since the initial time conditions for the forward propagator turn into final time conditions for the adjoint propagator. As for the forward stencil, we can write out the corresponding discrete expression for the update of the adjoint wavefield:

$$\mathbf{v}[\mathrm{time}-\mathrm{dt}] = 2\mathbf{v}[\mathrm{time}] - \mathbf{v}[\mathrm{time}+\mathrm{dt}] + \frac{\mathrm{dt}^2}{\mathbf{m}}\Delta\mathbf{v}[\mathrm{time}], \quad (2)$$

with dt being the time-stepping interval. Once again, this expression does not contain any (adjoint) source terms so far, which will be defined as a separate `SparseFunction` object. Since the source term for the adjoint wave equation is the difference between an observed and modeled shot record, we first define an (empty) shot record `residual` with 101 receivers and coordinates defined in `rec_coords`. We then set the data field `rec.data` of our shot record to be the data residual between the observed data `d_obs` and the predicted data `d_pred`. The symbolic residual source expression `res_term` for our adjoint wave equation is then obtained by *injecting* the data residual into the modeling scheme (`residual.inject`). Since we solve the time-stepping loop backward in time, the `res_term` is used to update the previous adjoint wavefield `v.backward`, rather than the next wavefield. As in the forward-modeling example, the source is scaled by $\mathrm{dt}^2/\mathbf{m}$. In Python, we have:

```
residual = PointSource(name='residual', ntime=nt,
                       grid=model.grid, coordinates=rec_coords)

res_term = residual.inject(field=v.backward,
                           expr=residual * dt**2 / model.m,
                           offset=model.nbpml)
```

In this demonstration, there is no real data. Instead we will generate the "observed" data via forward modeling with the true model `model`. The synthetic data is generated from the initial model `model0`. The resulting data, and their difference, are shown in Figure 2.

Finally, we create the full propagator by adding the residual source expression to our previously defined stencil and set the flag `time_axis=Backward`, to specify that the propagator runs backward in time:

```
op_adj = Operator([stencil_v] + res_term, time_axis=Backward)
```

In contrast to forward modeling, we do not record any measurements at the surface since we are only interested in the adjoint wavefield itself. The full script for setting up the adjoint wave equation, including an animation of the adjoint wavefield is available in `adjoint_modeling.ipynb`.

Computing the FWI gradient

The goal of FWI is to estimate a discrete parametrization of the subsurface by minimizing the misfit between the observed shot records of a seismic survey and numerically modeled shot records. The predicted shot records are obtained by solving an individual wave equation per shot location and depend on the parametrization **m** of our wave propagator. The most common function for measuring the data misfit between the observed and modeled data is the ℓ_2 norm, which leads to the following objective function (Lions, 1971; Tarantola, 1984):

$$\underset{\mathbf{m}}{\text{minimize}}\, f(\mathbf{m}) = \sum_{i=1}^{n_s}\frac{1}{2}\left\|\mathbf{d}_i^{\text{pred}}(\mathbf{m},\mathbf{q}_i) - \mathbf{d}_i^{\text{obs}}\right\|_2^2, \quad (3)$$

where the index i runs over the total number of shots n_s, and the model parameters are the squared slowness. Optimization problems of this form are called nonlinear least-squares problems, since the predicted data modeled with the forward-modeling propagator (`op_fwd()` in Part 1) depends nonlinearly on the unknown parameters **m**. The full derivation of the FWI gradient using the adjoint-state method is outside the scope of this tutorial, but conceptually we obtain the gradient by applying the chain

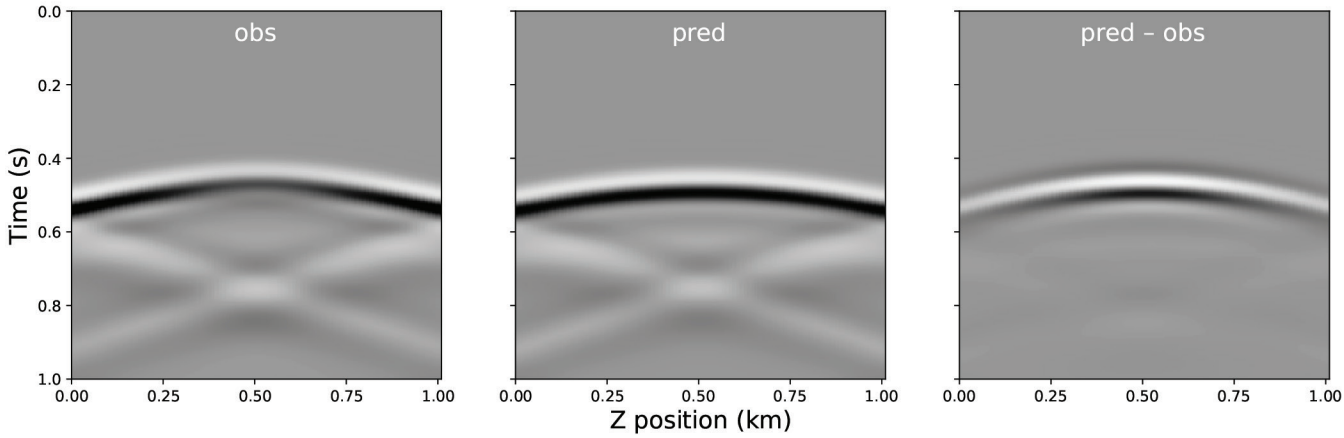

Figure 2. Shot records for a shot at a Z position of 0.5 km. (a) The observed data, using the known model with the high-velocity disc, contains a perturbation not present in (b) the "predicted" data, using the initial estimate of the model, which contains no disc. (c) The residual.

rule and taking the partial derivative of the inverse wave equation $\mathbf{A}(\mathbf{m})^{-1}$ with respect to $\mathbf{m}$, which yields the following expression (Plessix, 2006; Virieux and Operto, 2009):

$$f(\mathrm{m}) = -\sum_{i=1}^{n_s}\sum_{\mathrm{time}=1}^{n_t} \mathrm{u}[\mathrm{time}] \odot \ddot{\mathrm{v}}[\mathrm{time}]. \quad (4)$$

The inner sum time = 1,…, n_t runs over the number of computational time steps n_t and $\ddot{\mathrm{v}}$ denotes the second temporal derivative of the adjoint wavefield $\mathbf{v}$. Computing the gradient of equation 3, therefore corresponds to performing the point-wise multiplication (denoted by the symbol $\odot$) of the forward wavefields with the second time derivative of the adjoint wavefield and summing over all time steps.

To avoid the need to store the adjoint wavefield, the FWI gradient is calculated in the reverse time loop while solving the adjoint wave equation. To compute the gradient $\mathbf{g}$ for the current time step $\mathbf{v}$[time]:

$$\mathbf{g} = \mathbf{g} - \frac{\mathbf{v}[\mathrm{time\text{-}dt}] - 2\mathbf{v}[\mathrm{time}] + \mathbf{v}[\mathrm{time+dt}]}{\mathrm{dt}^2} \odot \mathbf{u}[\mathrm{time}]. \quad (5)$$

The second time derivative of the adjoint wavefield is computed with a second-order finite-difference stencil and uses the three adjoint wavefields that are kept in memory during the adjoint time loop (equation 2).

In Devito, we define the gradient as a `Function` since the gradient is computed as the sum over all time steps and therefore has no time dependence:

```
grad = Function(name="grad", grid=model.grid)
```

The update for the gradient as defined in equations 4 and 5 is then:

```
grad_update = Eq(grad, grad - u * v.dt2)
```

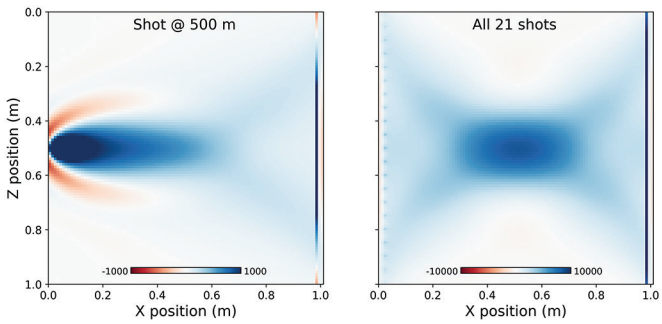

Figure 3. Gradient plots for (a) a single shot at 0.5 km and (b) the sum of all shots.

Now we must add the gradient update expression to the adjoint propagator `op_grad`. This yields a single symbolic expression with update instructions for both the adjoint wavefield and the gradient:

```
op_grad = Operator([stencil_v] + res_term + [grad_update],
                   time_axis=Backward)
```

Solving the adjoint wave equation by running the following now computes the FWI gradient for a single source. Its value is stored in `grad.data`.

```
op_grad(u=u0, v=v, m=model0.m,
        residual=pred.data-obs.data,
        time=nt, dt=dt)
```

Now we can iterate over all the shot locations, running the same sequence of commands each time.

This gradient can then be used for a simple gradient descent optimization loop, as illustrated at the end of the notebook `adjoint_modeling.ipynb`. After each update, a new gradient is computed for the new velocity model until sufficient decrease of the objective or chosen number of iteration is reached. A detailed treatment of optimization and more advanced algorithms will be described in the third and final part of this tutorial series.

Conclusions

We need the gradient of the FWI objective function in order to find the optimal solution. It is computed by solving adjoint wave equations and summing the point-wise product of forward and adjoint wavefields over all time steps. Using Devito, the adjoint wave equation is set up in a similar fashion as the forward wave equation, with the main difference being the (adjoint) source, which is the residual between the observed and predicted shot records.

With the ability to model shot records and compute gradients of the FWI objective function, we are ready to demonstrate how to set up more gradient-based algorithms for FWI in Part 3 next month. TLE

Acknowledgments

This research was carried out as part of the SINBAD II project with the support of the member organizations of the SINBAD Consortium. This work was financially supported in part by EPSRC grant EP/L000407/1 and the Imperial College London Intel Parallel Computing Centre.

Corresponding author: mathias.louboutin@gmail.com

References

Lions, J. L., 1971, Optimal control of systems governed by partial differential equations: Springer-Verlag Berlin Heidelberg.

Louboutin, M., P. A. Witte, M. Lange, N. Kukreja, F. Luporini, G. Gorman, and F. J. Herrmann, 2017, Full-waveform inversion, part 1: Forward modeling: The Leading Edge, **36**, no. 12, 1033–1036, https://doi.org/10.1190/tle36121033.1.

Plessix, R.-E., 2006, A review of the adjoint-state method for computing the gradient of a functional with geophysical applications: Geophysical Journal International, **167**, no. 2, 495–503, https://doi.org/10.1111/j.1365-246X.2006.02978.x.

Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, **49**, no. 8: 1259–1266, https://doi.org/10.1190/1.1441754.

Virieux, J., and S. Operto, 2009, An overview of full-waveform inversion in exploration geophysics: Geophysics, **74**, no. 6, WCC1–WCC26, https://doi.org/10.1190/1.3238367.

Zhang, Y., H. Zhang, and G. Zhang, 2011, A stable TTI reverse time migration and its implementation: Geophysics, **76**, no. 3, WA3–WA11, https://doi.org/10.1190/1.3554411.