Project 2: Solving a linearizable inverse problem

Technical Computation in the Earth Sciences

28th September, 2020

Introduction.0: The general problem.

Last week we dealt with some problems of the form

$$y = a * F(x) + b * G(x) + ...$$

where $a, b, etc.$ are constant and $F(x), G(x), etc.$ are different (linearly independent) functions of `x`. In particular, we learned to *invert* for the constants a, b, ..., given x and y.

But what if we want to invert for one or more constants *internal to* the nonlinear function(s) of x?

$$y = F(x, a, b, ...) + G(x, e, f, ...) + ...$$

This is no longer a linear inverse problem, but is quite possibly* still a *linearizable* inverse problem.

Today we will address the linearizable inverse problem of earthquake source (time and location) estimation from P-wave arrival times. The classic approach for linearizing this nonlinear inverse problem is attributable to Geiger (1910), but is effectively just an application of Newton's method. In brief, this means that rather than solving for the true solution right away, we will instead use an iterative method where at each step we get a little bit closer to the true solution.

The key to this approach is that for any differentiable function, no matter how nonlinear the function, the first derivative tells us how to construct a straight line *tangent to* our nonlinear function at any given point. Given the first derivative of our nonlinear function with respect to each of our parameters (remember ϕ from Project 1?), we effectively use these tangent lines as linear approximations for the function itself, and invert for the parameters that would solve the problem *if the function were linear* (i.e., equal to the tangent line) over the region between our initial guess for ϕ and the true solution. Since it's not, we won't get the right answer right away, but we'll get closer each time we try.

This sort of iterative linear approximation (using tangent lines) is the core of Newton's method, which you may have seen before in other classes. The only difference is that we'll be doing it in several dimensions at once, using linear algebra (the left inverse or "pseudoinverse" from Project 1) to solve for the adjustment to ϕ at each step.

* *Terms and conditions may apply. As long as our nonlinear function is somewhat smooth and isn't full of misleading local minima, we might have a chance*

Introduction.1: The specific problem. (*Or: the equations we'll need for this project*)

Consider an earthquake that occurrs at a position (x_0, y_0, z_0) at time t_0. If the seismic wave propagates with a seismic phase velocity v, the **predicted** arrival time $\hat{t_i}$ at some seismic station i at location x_i, y_i, z_i will be

$$\hat{t_i} = \frac{\sqrt{(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2}}{v} + t_0 \tag{1}$$

We wish to find the values of x_0, y_0, z_0, and t_0 that minimize the sum squared *misfit* (or *residual*) between the *observed* (t_i) and *predicted* $(\hat{t_i})$ data over all i

$$\sum_{i=1}^{N} (\hat{t_i} - t_i)^2$$

or in other words, the *least squares solution* for `ϕ = [x₀, y₀, z₀, t₀]`.

Whereas in Project 1 we had a forward equation of the form

$$\hat{y} = \boldsymbol{A}\ \phi$$

and an inverse equation of the form

$$\phi = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T y$$

now, we will have some very analagous equations that let us solve for increasingly better iterative *adjustments* to ϕ, denoted $\delta\phi$:

$$\hat{y} = \boldsymbol{A}\ \delta\phi \tag{2}$$

for the forward equation and

$$\delta\phi = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T y \tag{3}$$

for the inverse, where $\boldsymbol{A}$ is now a matrix of *derivatives*, $\boldsymbol{y}$ is now a column vector of *residuals*, and $\boldsymbol{\delta\phi}$ is a column vector of *adjustments* to our last proposed source location:

$$\boldsymbol{A} = \begin{bmatrix} \frac{\partial t_1}{\partial x_0} & \frac{\partial t_1}{\partial y_0} & \frac{\partial t_1}{\partial z_0} & \frac{\partial t_1}{\partial t_0} \\ \frac{\partial t_2}{\partial x_0} & \frac{\partial t_2}{\partial y_0} & \frac{\partial t_2}{\partial z_0} & \frac{\partial t_2}{\partial t_0} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial t_N}{\partial x_0} & \frac{\partial t_N}{\partial y_0} & \frac{\partial t_N}{\partial z_0} & \frac{\partial t_N}{\partial t_0} \end{bmatrix} \qquad y = \begin{bmatrix} t_1 - \hat{t_1} \\ t_2 - \hat{t_2} \\ \vdots \\ t_N - \hat{t_N} \end{bmatrix} \qquad \delta\phi = \begin{bmatrix} \delta x_0 \\ \delta y_0 \\ \delta z_0 \\ \delta t_0 \end{bmatrix}$$

When implementing this in Julia, it may be convenient to write a function, let's call it `G(x₀, y₀, z₀, t₀)` that calculates the vector of predicted arrival times $\hat{t}$, and another `∂G(x₀, y₀, z₀, t₀)` that calculates the matrix $\boldsymbol{A}$. Keep in mind also that since we're working now with an iterative method, you'll have to start with some initial guess for `ϕ`.

1

Part 1

The file `project2.nc` contains station locations and observed arrival times for an earthquake that occurred at a hypocenter x_0, y_0, z_0 at time t_0. Write a function to estimate the source (x_0, y_0, z_0, t_0) of the earthquake recorded in `project2.nc` using Geiger's method, where the user may specify as an input to the function (among other things) the number of iterations of Geiger's method to run. You may also wish to keep the goals of Part 2 in mind when designing your function. If you don't want to calculate the derivatives of Equation 1 by hand, feel free to use either your favorite symbolic math program (perhaps Mathematica or Wolfram Alpha) or even one of Julia's many *automatic differentiation* packages like Zygote. To read the input file `project2.nc` you'll need to add the NetCDF package with

```
] add NetCDF
```

and then read the file with something along the lines of

```julia
using NetCDF
ncinfo("project2.nc") # To find out what variables we have
somevarname = ncread("project2.nc", "somevarname") # Using the variable names you just discovered
```

2

Part 2

Assess the performance of your function from part 1. How do you know when you have run enough iterations? You may wish to make some number of plots or graphs, considering such variables as the residual (the misfit between model and data) and the magnitude of change in model parameters between iterations.

3

Report

In addition to your code, please describe your results and their significance in the format of a brief lab report with one or more figures. While you may collaborate with others while coding (as long as this is clearly noted in comments within your code), the report should be purely your own work. Some questions you might think about involve the strenghts and weaknesses of this arrival-time-based method, including:

- Are we making any specific assumptions about the medium through which our seismic waves are propagating?
- If so, *where* / *over what scale* might these be good or bad assumptions?
- Are there any conditions where this P-wave-arrival-time-only method might have particular advantages relative to the P-versus-S-wave arrival time lag method?