Tomographic Modeling

Jonathan M. Lees
University of North Carolina, Chapel Hill
Department of Geological Sciences

July 13, 2012

1 Tomographic Inversion: Introduction

Seismic tomography uses the travel time of elastic waves to probe the internal structure of the earth. It differs from traditional medical tomography in four major aspects:

1. acoustic signals travel in highly curved raypaths in media that vary in 3-dimensions,

2. the travel time is a non-linear function of the velocity field ("velocity" field in the seismic sense is the scalar wave speed),

3. when the sources are earthquakes, the distribution of rays covering the target cannot be controlled and is often highly inhomogeneous and

4. uncertainties in the travel time exist because the source location and origin time must be determined from the observations themselves. These differences indicate that special care must be taken when techniques borrowed from the medical field are applied to seismic data. Specifically, due to the non-uniform distribution of sources and receivers, the convolutional techniques of inversion, common in medical tomography, are inapplicable in the seismic case and iterative approaches are used instead.

In this demonstration we show how to set up a synthetic 2D tomographic modeling experiment and perform the inversion via matrix inversion.

2 History

Tomography (literally, 'slice picture') originated in radio astronomy as a method to image aspects of remote regions of the universe. Later physicists and bio-physicists collaborated to create the first methodology and instrumentation that led to the first tomographic analysis of live tissue, especially human bodies. This approach was called 'computer aided tomography' or CAT scans. Researchers who pioneered these methods received the Nobel Prize in physiology and medicine in 1979 (Allan Cormack and Godfrey Hounsfield). At the same time seismologists recognized that similar methodology could be applied to imaging the earth. Early papers on these approaches were not called tomography, but simply 'three-dimensional analysis'. It was not until the early 1980's that data sets large enough to actually mimic an approach similar to medical tomography emerged.

3 Basic Idea

The basic idea is illustrated in cartoon form in Figure 1 . Earthquakes emit seismic energy that travels out to the stations at the surface. At first, we assume an intervening velocity structure, typically one dimensional, and use that to predict traveltimes to each station. If the model is correct the difference between predicted and observed arrivals will be small. If waves pass through anomalous structures, however, travel times will be perturbed and the differences will become significant. Seismic tomography often involves using the travel time residuals to reconstruct anomalies where large numbers of raypaths overlap at varying angles. It can be shown that with complete coverage from all angles, the anomalous body can be reconstructed perfectly. This ideal situation is never achieved in real analyses, of course.

4 Inversion approach

Nearly all seismic tomography is founded on a linearization of the highly non-linear seismic inversion problem. In equation (1) it was noted that the raypath depended on the velocity model which also depends on the raypaths in the inversion. Furthermore, earthquake locations are also derived using the velocity models, so this introduces an additional non-linearity. To circumvent the nonlinearity we usually introduce a linearization of the inversion and iterate a sequence of linear inversion in the hopes that we will converge to the correct non-linear solution. This is accomplished by assuming we have a reasonably close initial guess to the correct solution and then seeking a small perturbation which will drive the model closer to the correct, final model. The sequence of calculations involves making an initial (typically 1D) model, finding the raypaths in that model, perturbing the (3D) model so the travel times are minimized, adding new perturbations to the first model and iterating. Once the new model is derived, new earthquake locations are calculated and the process is repeated. These iterations usually converge in a few to several steps. Convergence is determined when the models change

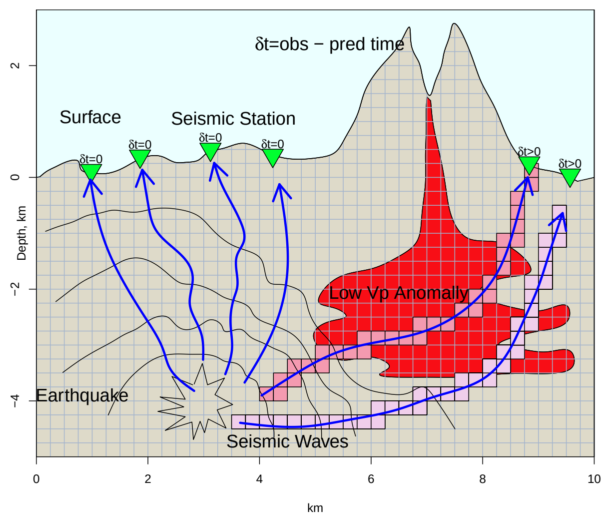

Figure 1: Tomographic inversion of a Magma Chamber. The grid in the subsurface represents the grid used for the tomographic inversion. The two rays on the right traverse through the anomalous magma body. those on the left do not, so they do not contribute changing the model. The blocks highlighted are the ones that are perturbed sequentially in row-action methods of inversion.

by a very small amount between iterations and travel time differentials get small.

The inversion described above is typically discretized (digitized) by assuming the earth is composed of blocks, or nodes of points, where the velocity is defined. The integrals in Equation (1) can be discretized and cast in matrix form. The matrices involved include a data design matrix which describes the way the raypaths intersect the earth model and the data (travel time perturbations), i.e. the differences between the observed travel times and calculated times through the current model. New, upgraded, models are derived by least-squares methods. Since the matrices are very sparse, specialized methods have been developed to store the data and arrive at an inverse solution with great speed.

The inversion described above is typically discretized (digitized) by assuming the earth is composed of blocks, or nodes of points, where the velocity is defined. The integrals in Equation (1) can be discretized and cast in matrix form. The matrices involved include a data design matrix which describes the way the raypaths intersect the earth model and the data (travel time perturbations), i.e. the differences between the observed travel times and calculated times through the current model. New, upgraded, models are derived by least-squares methods. Since the matrices are very sparse, specialized methods have been developed to store the data and arrive at an inverse solution with great speed.

5 Methodology and Inversion

Once all the data are collected and the relationship of the intersecting ray paths with the target region is discretized and digitized, a set of matrices is typically set up to solve the so called inverse problem: what earth model would give rise to the travel time residuals observed. If we represent the earth model as a vector of perturbations (anomalies), x, the interaction of the raypaths with the earth as A, the array of travel time residuals, Δt, one can relate the earth to the residuals by the simple linear relationship,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{5.1}$$

This matrix equation is solved using a variety of methods depending on the structure of the matrices, although the specific approach usually does not have a large impact on the resulting images.

6 Synthetic Example using R

Set up libraries used in the following:

```
library(sp)
library(splancs)
library(RTOMO)
```

Next we create a synthetic target region and grid where the earthquakes and stations will be distributed.

We set the background velocity to 4.5 km/sec and we create 2 irregular anomalies, one with a 10 percent positive perturbation and one with negative 5 percent anomaly. This is totally arbitrary.

```
NX = 100
NY = 100
xo = seq(from=1, to=NX, by=1)
yo = seq(from=1, to=NY, by=1)
### v (or s) model

V = 4.5
v1 = V+0.1*V
v2 = V-.05*V
rad = 20
```

The travel times are calculated by using the slowness in seismology, or 1/velocity.

```
MOD =  matrix(1/V, ncol=NX, nrow=NY)
PHANT =  matrix(0, ncol=NX, nrow=NY)
M = meshgrid(xo, yo)

## image(xo, yo, MOD, col="grey")
```

Two arbitrary, complex anomalous bodies are created. I made these by clicking on the screen a then saving the points:

```
##  A1 = locator(type='o', col='black')
A1=list()
A1$x=c(21.4380148625385,22.9678921197393,25.2038665725713,29.0874011485427,
  31.5587413332518,40.7380048764569,40.7380048764569,42.0325164017807,
  46.1514167096291,50.0349512856005,45.2099537821209,29.2050840144812,
  29.7934983441739,33.6770329201453,28.6166696847886,21.9087463262926,
  11.6703369896408,8.96363107305464,17.083748822813,15.3185058337351,
  7.08070521803821,11.5526541237022,14.2593600402883,15.7892372974892,
  17.6721631525056,19.0843575437679)
A1$y=c(55.6872037914692,57.8199052132701,59.0047393364929,
  59.0047393364929,55.2132701421801,58.5308056872038,
  61.9668246445498,70.6161137440758,74.7630331753555,79.8578199052133,
  83.175355450237,88.2701421800948,83.5308056872038,81.9905213270142,
  77.4881516587678,77.3696682464455,78.909952606635,71.3270142180095,
  70.1421800947867,66.1137440758294,54.5023696682464,49.4075829383886,
  51.6587677725119,57.1090047393365,55.0947867298578,52.1327014218010)
##  A2 = locator(type='o', col='black')

A2=list()
A2$x=c(90.6355400343922,82.750788016511,72.9831101436132,81.2209107593101,
  78.8672534405396,63.9215294663467,55.8014117165883,47.0928796371373,
  65.6867724554246,70.9825014226583,57.9197033034818,45.9160509777521,
  48.2697082965226,64.9806752597934,72.3946958139206,76.6312789877075,
  81.1032278933716,74.8660359986297,86.7520054584209,92.7538316212857)
A2$y=c(39.5734597156398,50.4739336492891,52.60663507109,49.0521327014218,
  46.0900473933649,46.563981042654,45.260663507109,35.9004739336493,
  37.914691943128,33.175355450237,25.5924170616114,24.2890995260664,
  16.8246445497630,15.7582938388626,11.6113744075829,9.12322274881517,
  19.4312796208531,24.7630331753554,30.2132701421801,31.3981042654028)
```

These bodies are saved as geometric outlines and all the points inside the bodies are given the perturbation assigned. We used the program inout to extract which points of the model were inside each body.

```
mypoly = as.points(as.vector(A1$x) , as.vector(A1$y))
mypoints = as.points(as.vector(M$x),as.vector(M$y))
```

```
INTEMP1 = inout(mypoints, mypoly, bound=TRUE )
mypoly = as.points(as.vector(A2$x) , as.vector(A2$y))
INTEMP2 = inout(mypoints, mypoly, bound=TRUE )
INTEMP = INTEMP1 | INTEMP2
MOD[INTEMP1] = 1/v1
MOD[INTEMP2] = 1/v2
ZEE = t(MOD)



image(xo, yo, ZEE, col=terrain.colors(100) )

##############   code
```

7 Station Distribution

Make the stations:

```
##############   code

stax = runif(20, min=1, max=100)
stay = runif(20, min=1, max=100)

image(xo, yo, ZEE, col=terrain.colors(100) )
points(stax, stay, pch=6, col='blue')
```

8 Event (source) Distribution

Make the randomly distributed earthquakes (sources).

```
NEV = 200
evx = runif(NEV, min=1, max=100)
evy = runif(NEV, min=1, max=100)
####   get random radii for earthquake magnitude
rads = rnorm(NEV, m=50, s=10)
```

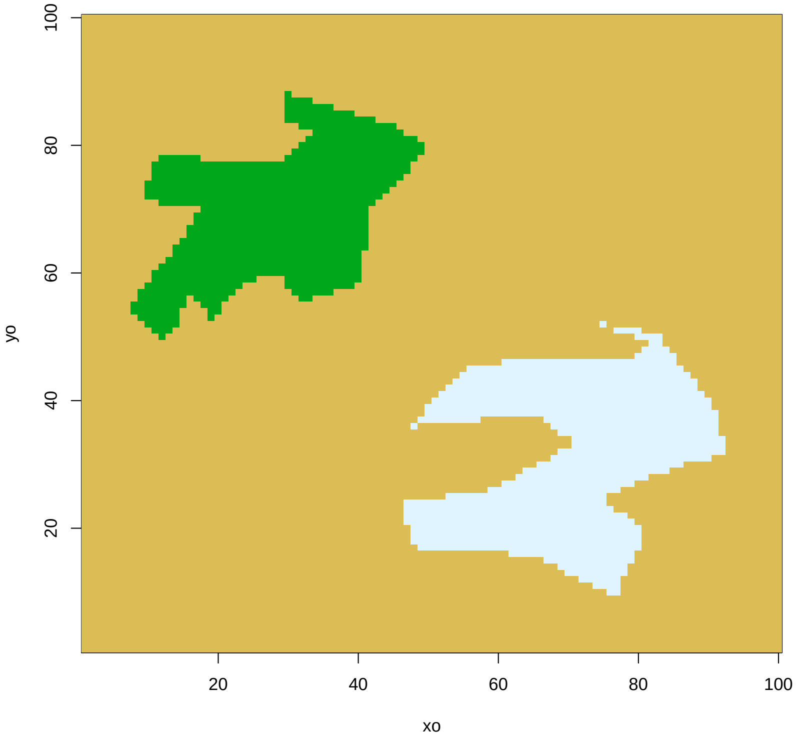

Figure 2: Tomographic Phantom (Synthetic model)

```
image(xo, yo, ZEE, col=terrain.colors(100) )
points(evx, evy, pch=8, col='red')
```

Here we estimate which stations record each event by using the radius (magnitude) of the event:

```
image(xo, yo, ZEE, col=terrain.colors(100) )
points(evx, evy, pch=8, col='red')
for(i in 1:length(evx))
  {
```

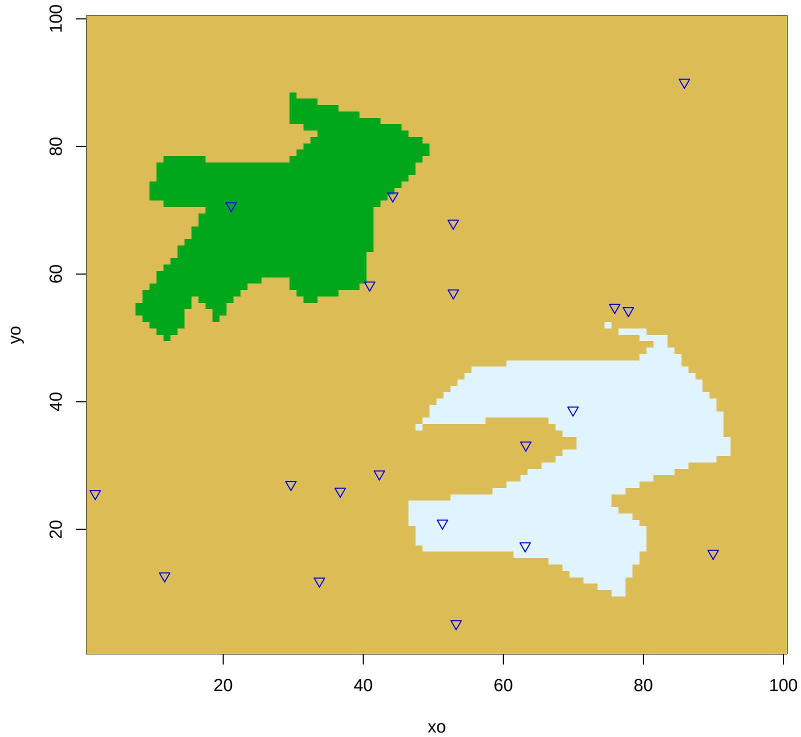

Figure 3: (Random) Station Distribution

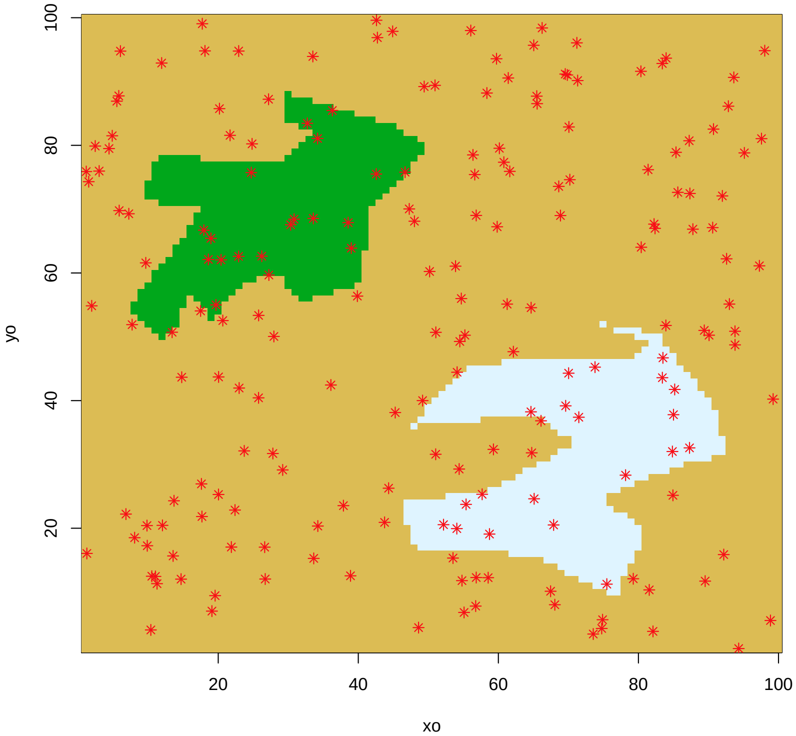

Figure 4: (Random) Event Distribution

```
    print(paste(sep=' ', "working on ", i))
    dis = sqrt( (evx[i]-stax)^2+(evy[i]-stay)^2)
    w = which(dis<rads[i])
    if(length(w)>1)
      {
        segments(evx[i], evy[i], stax[w],  stay[w])
}

  }
```

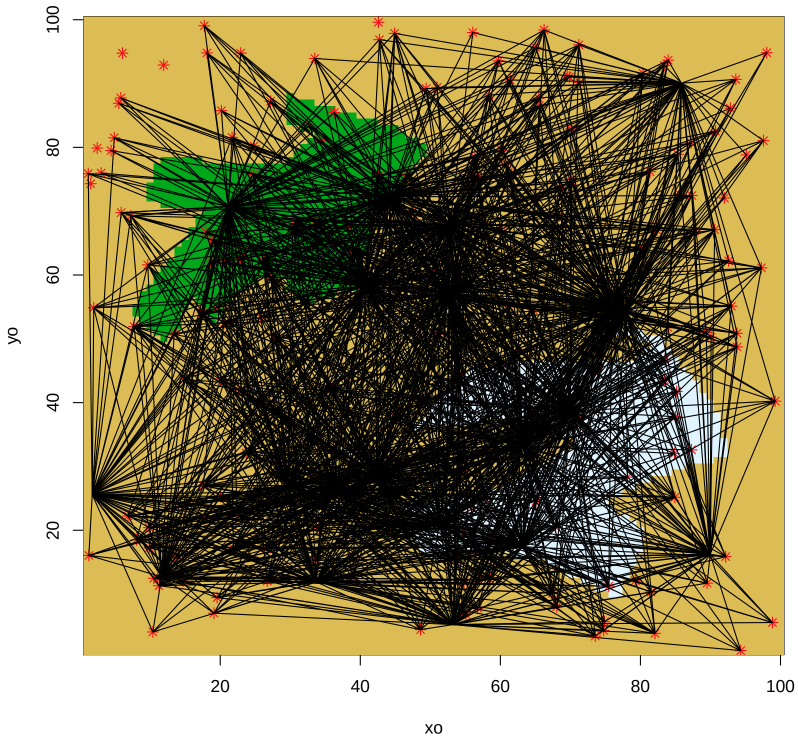

Figure 5: Raypath Distribution

9 Prepare the Matrix

Determine the intersection of the raypaths with the model. In 2D this is just the length of the raypath in the i^{th} block.

```
COV = list()
k = 0
for(i in 1:length(evx))
  {
    print(paste(sep=' ', "working on ", i))
    dis = sqrt( (evx[i]-stax)^2+(evy[i]-stay)^2)
    w = which(dis<rads[i])
    if(length(w)>1)
      {
        segments(evx[i], evy[i], stax[w],  stay[w])
        for(j in 1:length(w))
          {
        RAP = get2Drayblox(evx[i], evy[i],stax[w[j]] ,stay[w[j]] ,
          xo, yo, NODES=TRUE, PLOT=FALSE)
        slns = MOD[cbind(RAP$ix, RAP$iy)]
        tt = slns*(RAP$lengs)
        k = k+1
        COV[[k]] = list(RAP=RAP, tt=tt)
      }

      }

  }
```

Add Noise to the data:

```
####image(xo, yo, ZEE, col=terrain.colors(100) )

#### points(stax, stay, pch=6, col='blue')

###################################################
###################################################
```

```
##################################################

NOISE = 0.001
    for(i in 1:length(COV))
      {

       COV[[i]]$noise = rnorm(n=1, m=0, sd=NOISE)


      }

##################################################
```

Now the synthetic data has been created. The data is stored in memory as lists.

The list named *COV* has the coverage information. That is the travel times and the raypath coverage. The travel times are in a vector called *tt*. these are the 'true' arrival times. The raypaths are stored in a second list called *RAP*. It has the coordinated of each (straight) raypath transecting the model.

10 Inversion by Backprojection

Here we solve the problem,

$$\Delta \boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}$$

Here we solve the tomography problem via iterative backprojection. This is the algorithm that I believe can be made parallel.

10.1 Inversion by backprojection

Do 30 iterations, include noise. At each step use the current model and upgrade the velocity.

```
######
PHANT =  matrix(0, ncol=NX, nrow=NY)
for(j in 1:30)
  {
```

```
    print(paste("Iteration", j))
    for(i in 1:length(COV))
      {
        ###
        oldsln = as.vector(PHANT[cbind(COV[[i]]$RAP$ix, COV[[i]]$RAP$iy)])
        slns = (1/V)*(1+oldsln)
        t1 = slns*COV[[i]]$RAP$lengs
        DT = (COV[[i]]$tt - t1) + COV[[i]]$noise

        PHANT[cbind(COV[[i]]$RAP$ix, COV[[i]]$RAP$iy)] =
          PHANT[cbind(COV[[i]]$RAP$ix, COV[[i]]$RAP$iy)]+
            DT*COV[[i]]$RAP$lengs/(sum(COV[[i]]$RAP$lengs))


      }

  }
```

To illustrate, here is inversion by backprojection using only 1 iteration:

```
###################################################

PHANT1 =  matrix(0, ncol=NX, nrow=NY)
for(i in 1:length(COV))
  {

    t1 =(1/V)*COV[[i]]$RAP$lengs
    DT = (COV[[i]]$tt - t1) + COV[[i]]$noise

    PHANT1[cbind(COV[[i]]$RAP$ix, COV[[i]]$RAP$iy)] =
      PHANT1[cbind(COV[[i]]$RAP$ix, COV[[i]]$RAP$iy)]+
        DT*COV[[i]]$RAP$lengs/(sum(COV[[i]]$RAP$lengs))


  }
```

```
##  screens(2)


##dev.set(dev.next())
```

Display results: This is the original phantom:

```
image(xo, yo, ZEE, col=tomo.colors(100) )
polygon(A1)
polygon(A2)
```

This is the backprojected image with only one iteration:

```
## dev.set(dev.next())

image(xo, yo, t(PHANT1), col=tomo.colors(100) )
polygon(A1)
polygon(A2)
```

This is the backprojected image:

```
## dev.set(dev.next())
image(xo, yo, t(PHANT), col=tomo.colors(100) )
polygon(A1)
polygon(A2)
```

11 Modifications and Improvements

11.1 Damping or Regularization

The solution achieved through solving the normal equations may work, but it may fail since noise in the data corrupts the solution. Some kind of damping

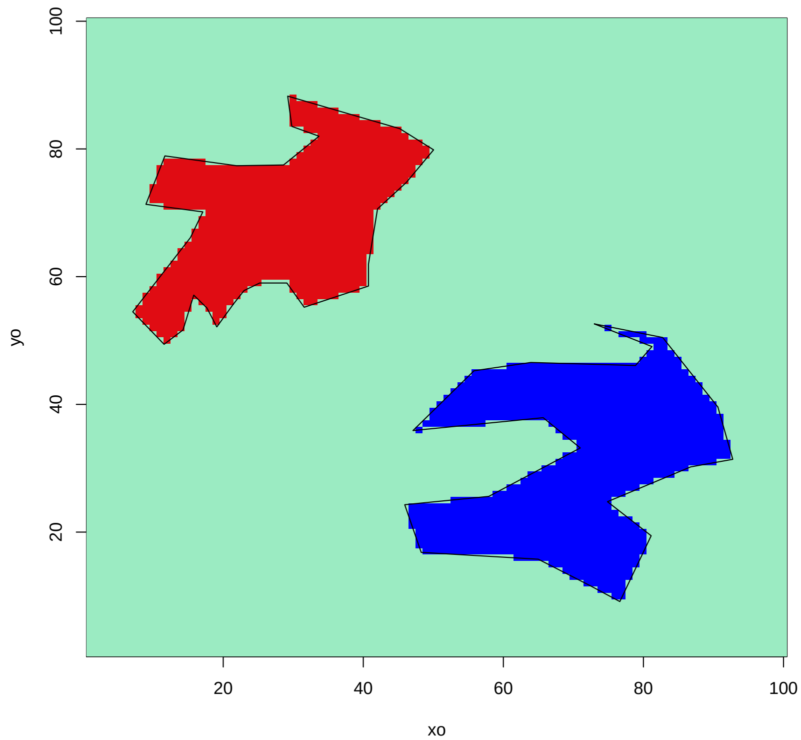

Figure 6: Synthetic phantom - this is the "Truth"

should be applied prior to solving equations in the manner. This is accomplished by adding additional constraint equations to the original matrix inversion problem. The original formula,

$$\boldsymbol{Ax} = \boldsymbol{b}$$

becomes

$$\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{\Theta} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix}$$

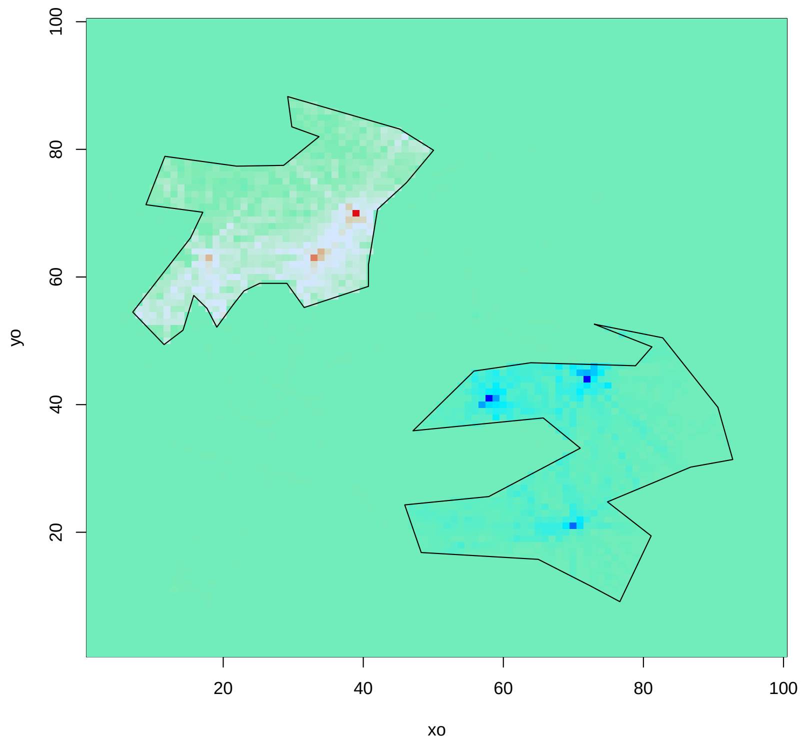

Figure 7: Tomographic inversion result with one backprojected iteration.

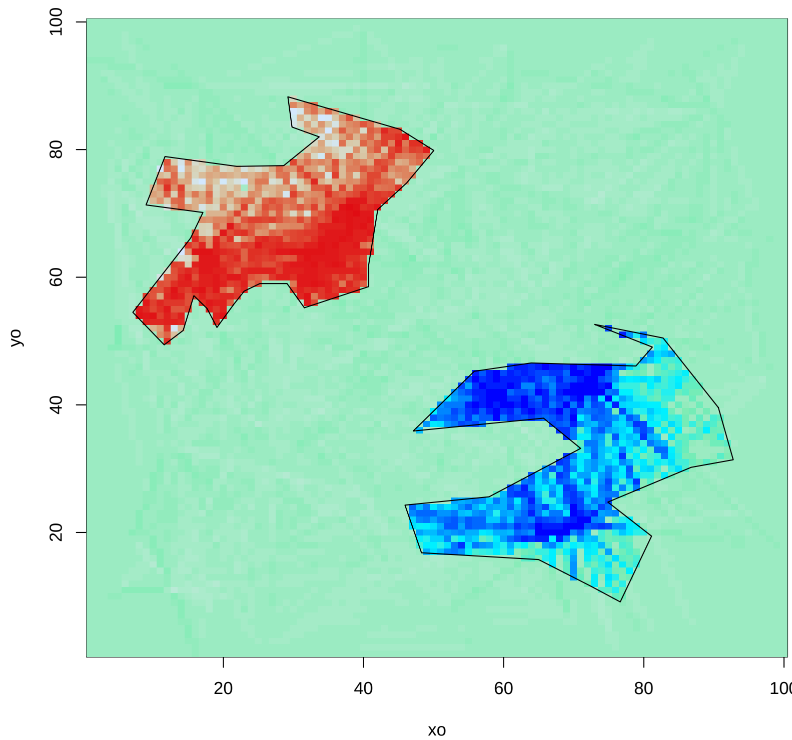

Figure 8: Tomographic inversion result, 30 iterations. This is the image seen through the inversion process.

Where Θ is a matrix of constraints, as described above in the discussion of damping as

$$\boldsymbol{\Theta} = \begin{bmatrix} \theta & 0 & 0 & \cdots \\ 0 & \theta & 0 & \cdots \\ 0 & 0 & \theta & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \theta \boldsymbol{I}$$

There is no analytical way to determine the correct damping θ. This should be done by trial and error, or through experience etc. The damping can be thought of as some kind of *a priori* information constraining the inversion. It is saying, in effect, allow perturbations, but do not allow them to be too large. How large is "too large"? There in lies the problem. Since damping is required because of potentially destabalizing effects of noise, it may be possible to estimate a good damping parameter based on estiamted uncertainty in arrival time picks.

11.2 Weighting

The equations may (should) not have all the same importance. For example, stations much further away will experience more attenuation and arrival time determination may be more prone to uncertainty. Some stations may have known problems or may be located on sites that consistently provide poor arrival time picks. A weighting matrix $\boldsymbol{W}$ is then introduced and mulitplied to account for variable quality of data:

$$\begin{bmatrix} \boldsymbol{WA} \\ \boldsymbol{\Theta} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{Wb} \\ \boldsymbol{0} \end{bmatrix}$$

This set of equations may be solved using a variety of methods including QR-Decomposition, cholesky decomposition, singular value decomposition, etc.... Note that the inversion is repeated for each step in the nonlinear convergence process, i.e. each time the event is shifted a new matrix is formed and a new, perturbed location is derived.