Inverse Problems Project 2017

Introduction

For a model made up of discrete blocks, the tomography problem can be written as a linear sum of the forward problem for ray j through k cells

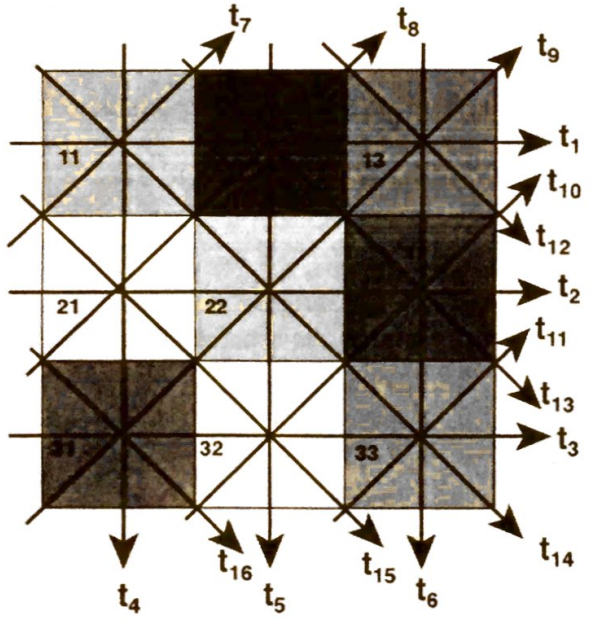
$$t_j = \sum_{i=1}^{K} l_{i,j} s_i$$

with $l_{i,j}$ being the length of the ray path of ray j in block i, which has model parameter s_i.

We then write this as a vector-matrix equation, **d** = **Gm**. Here **d** is the Nx1 data vector of observed travels times (t), **m** is the Kx1 model vector of the slowness values (s), and **G** is a NxK matrix that maps **m** into **d**.

The solution is given as $\mathbf{m} = \mathbf{G}^{-g}\mathbf{d}$ where $\mathbf{G}^{-g}$ is the generalized inverse, calculated using singular value decomposition (SVD).

The aim of this small project is to write a reasonably general code for a 2D linear tomography problem using Singular Value Decomposition (that is, the generalized inverse matrix). We want to consider the general tomographic problem, for which we want to be able to change the dimensions of the 2D model (say **Nx** x **Ny**) and to perform different resolution tests. The matrix **G** depends on the number of ray paths and their geometries. For example, we can assume a general ray path distribution as below (for **Nx** = **Ny** = 3).



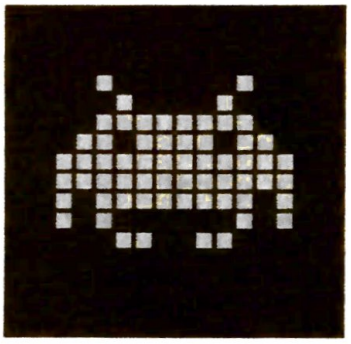
(1) Write a general code to calculate the **G** matrix for a model of size **Nx*Ny** (where **Nx** is the number of rows, **Ny** is the number of columns, and the code should allow for the case when **Nx** is different to **Ny**). Use a block size of width = height = 1. Write an explanation of your algorithm (if you want, you can use a different ray geometry to that shown above).

(2) Using Nx = Ny = 7, test the reconstruction of a spike model (one as in where the centre block is 1 and all others have a value of 0). Compare the reconstructions using perfect data, and data with different levels of noise (sigma = 0.1, 0.5, 1.0). What do you notice ?

(Spike model is something like.... myspike = zeros(Nx, Ny); Image(floor(Nx/2+1), floor(Ny/2)+1) = 1;)

(3) Repeat (2) and (3) with Nx = Ny = 11, 15, 21. Are there differences in the resolution of the spike model as the number of blocks increases and if so how can we explain the differences?

(4) As a test of the general code, we will generate a synthetic image and try to recover it. Generate a simple black and white pixel image (you could take the first letter of your first name (prenom), or use the space invader image below) in a square cell model (you choose the size of the model but make it at least 15x15). In the case of a binary (black and white) image the model parameter will have a value of 1 for a black square and 0 for a white square. We also want to be able to choose a more complex model structure to capture images such as the SuperMario below. More complex images can be built using an integer for each different colour. With Matlab, you can load an image with imread(myimage) and convert it to greyscale with greymyimage = rgb2gray(myimage). Then choose the ray paths as described earlier and generate synthetic data using **d = Gm**. Allow the data to be inverted with or without noise.



We want to be able to save both the model and the data to a file and reload them, so make sure your code can do that. You can choose the format for the output as you like.

Invert the perfect data, and then try this again with noisy data (sigma = 0.1), and plot the reconstructed models.

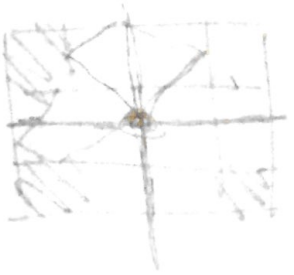
(5) Using the same image, increase the resolution by dividing each original cell into smaller 3x3 cells, so one cell in the original model with a value of 1 with now be represented by 9 cells all with a value of 1.

Repeat the inversion with perfect and noisy data (sigma = 0.1,0.5).

Comment on the differences, if any, for the solutions between (4) and (5), and show how we can assess the resolution of each model, assuming we do not know the true model.

(6) Write a short user guide for your code. You will need to explain how to format and input both a model and data as required by however you have set up the problem).

Send the code (including your test data/image), a short report with the answers to the questions and the short user guide to kerry.gallagher@univ-rennes1.fr in pdf no later than 17.00, 12 January, 2018. Please use the subject **'TOMOGRAPHY'**. I will run your codes with my own images to check the codes work.



HOW TO INPUT IMAGES....either binary but in an obvious way, so we can read in (or create in the code) a matrix (Nx x Ny with integer or real values)... or load an bitmap image...

For Matlab

```
myimage = imread('/Users/kerry/mario.png');
imagesc(myimage)
greymyimage = rgb2gray(myimage);
```

For Scilab we need to install toolboxes for **IPD** (Image Processing Design), or **SIVP**, **SIP** (just for PC). They require **opencv**. For matlab like plot commands we can use the plotlib toolbox.

To install **opencv**, you can follow the videos on youtube...
For Mac
https://www.youtube.com/watch?v=U49CVY8yOxw
For Windows
https://www.youtube.com/watch?v=EcFtefHEEII
or
https://www.youtube.com/watch?v=ScAPinibluA

(SCI gives us the pathname for the working Scilab directory)

To install the toolboxes we can use **Atoms**
(details are given here http://wiki.scilab.org/ATOMS)

Type "Scilab help Atoms"

To launch the user interface (you need to be connected to the internet)

```
atomsGui();
```

To find a toolbox called module

```
atomsSearch("module name")
```

To install a module

```
atomsInstall("plotlib") // Matlab like commands
atomsInstall("IPD");
```

or

```
atomsInstall("IPD","user") ...just for single user (do not need admin)
```

We need to quit Scilab to update the toolboxes. Once this is done, the toolboxes will be loaded automatically. They will also be present in the Scilab help menu.
To check what has been installed, we can type

```
disp( atomsGetInstalled() );
```

To load a toolbox manually, call:

```
atomsLoad("IPD");
```

Example code with the IPD toolbox.

```
myimage = ReadImage("/Users/kerry/Mario.png");
bwimage = RGB2Gray(myimage);
% Note these may be integer so need to convert to double for plotting
SSG = double(SSG);
figure();
ShowColorImage(myimage,'Color Image')   % this did not work on Mac 10.9
figure();
ShowImage(bwimage,'BW Image')  or Matplot(SSG)
```