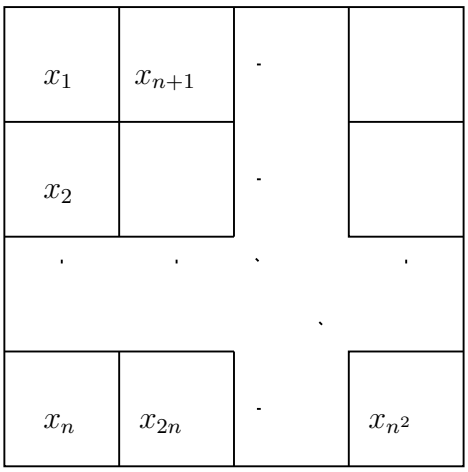
Homework 4

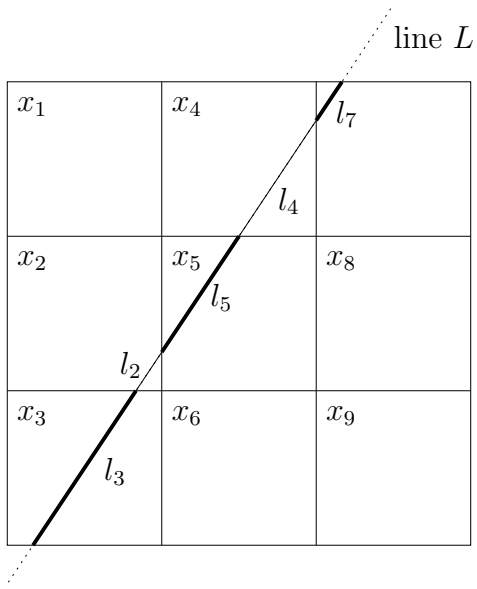
1. *Tomography.* In this problem we explore a simple version of a tomography problem. We consider a square region, which we divide into an $n \times n$ array of square pixels, as shown below.



The pixels are indexed column first, by a single index i ranging from 1 to n^2, as shown above. We are interested in some physical property such as density (say) which varies over the region. To simplify things, we'll assume that the density is constant inside each pixel, and we denote by x_i the density in pixel i, $i = 1, \ldots, n^2$. Thus, $x \in \mathbf{R}^{n^2}$ is a vector that describes the density across the rectangular array of pixels. The problem is to estimate the vector of densities x, from a set of sensor measurements that we now describe. Each sensor measurement is a *line integral* of the density over a line L. In addition, each measurement is corrupted by a (small) noise term. In other words, the sensor measurement for line L is given by

$$\sum_{i=1}^{n^2} l_i x_i + v,$$

where l_i is the length of the intersection of line L with pixel i (or zero if they don't intersect), and v is a (small) measurement noise. This is illustrated below for a problem with $n = 3$.



In this example, we have $l_1 = l_6 = l_8 = l_9 = 0$, and your sensor measurement will be something like

$$y = 0.4x_2 + 1.3x_3 + 0.9x_4 + 0.9x_5 + 0.4x_7 + v$$

Now suppose we have N line integral measurements, associated with lines $L_1, \ldots, L_N$. From these measurements, we want to estimate the vector of densities x. The lines are characterized by the intersection lengths

$$l_{ij}, \quad i = 1, \ldots, n^2, \quad j = 1, \ldots, N,$$

where l_{ij} gives the length of the intersection of line L_j with pixel i. Then, the whole set of measurements forms a vector $y \in \mathbf{R}^N$ whose elements are given by

$$y_j = \sum_{i=1}^{n^2} l_{ij} x_i + v_j, \quad j = 1, \ldots, N.$$

And now the problem: you will reconstruct the pixel densities x from the line integral measurements y. The class webpage contains `tomodata_fullysampled.json` and `tomodata_undersampled.json`, each of which contains the following variables:

- `N`, the number of measurements (N),
- `n_pixels`, the side length in pixels of the square region (n),
- `y`, a vector with the line integrals $y_j,\ j = 1, \ldots, N$,
- `lines_d`, a vector containing the displacement $d_j,\ j = 1, \ldots, N$, (distance from the center of the region in pixels lengths) of each line, and
- `lines_theta`, a vector containing the angles $\theta_j,\ j = 1, \ldots, N$, of each line.

We also provide the function `line_pixel_length.jl` on the webpage, which you do need to use in order to solve the problem. This function computes the pixel intersection lengths for a given line. That is, given d_j and θ_j (and the side length n), `line_pixel_length.jl` returns a $n \times n$ matrix, whose i, jth element corresponds to the intersection length for pixel i, j on the image. Use this information to find x, and display it as an image (of n by n pixels). You may refer to `TV_inpainting.ipynb` from the course webpage, which contains several utility functions that can be used to display your reconstructed images.

(a) Use `tomodata_fullysampled.json` to reconstruct the pixel densities. Note that he file contains the measurement data obtained from 5184 line integrals on a 60×60 image, therefore providing more measurement data than the unknown variables. So you can simply do this job by setting up a problem like

$$\text{minimize}_x \quad ||Ax - y||_2^2$$

where your feature matrix A can be constructed by using `line_pixel_length.jl`.

(b) Use `tomodata_undersampled.json` to reconstruct the pixel densities. Note that he file contains the measurement data obtained from 1296 line integrals on a 60×60 image, therefore providing way less measurement data than the unknown variables. In order to solve this problem, try the following regularizer defined by the 2-normed total variation (TV) function. You are welcome to use the `Dx` and `Dy` provided in `TV_inpainting.ipynb`, to express the TV function.

$$\text{TV}_2(X) = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \left(|X_{ij} - X_{i+1,j}|^2 + |X_{ij} - X_{i,j+1}|^2 \right) = \left\| \begin{bmatrix} D_x \\ D_y \end{bmatrix} \text{vec}(X) \right\|_2^2$$

In other words, you will have to set up and solve the following. Let us simply choose our weighting parameter as $\lambda = 0.1$, however you are strongly encouraged to explore different weighting parameters.

$$\text{minimize}_x \quad ||Ax - y||_2^2 + \lambda \left\| \begin{bmatrix} D_x \\ D_y \end{bmatrix} x \right\|_2^2$$

(c) Repeat the above problem with the 1-normed total variation (TV) function.

$$\text{TV}_1(X) = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \left(|X_{ij} - X_{i+1,j}| + |X_{ij} - X_{i,j+1}| \right) = \left\| \begin{bmatrix} D_x \\ D_y \end{bmatrix} \text{vec}(X) \right\|_1$$

In other words, you will have to set up and solve the following. Similarly, let us simply choose our weighting parameter as $\lambda = 0.1$, however you are strongly encouraged to explore different weighting parameters.

$$\text{minimize}_x \quad ||Ax - y||_2^2 + \lambda \left\| \begin{bmatrix} D_x \\ D_y \end{bmatrix} x \right\|_1$$

For problem (b) and (c), you may use Julia 0.6 version with the Convex.jl package and ECOS solver.

Note: While irrelevant to your solution, this is actually a simple version of *tomography*, best known for its application in medical imaging as the CAT scan. If an *x-ray* gets attenuated at rate x_i in pixel i (a little piece of a cross-section of your body), the j-th measurement is

$$z_j = \prod_{i=1}^{n^2} e^{-x_i l_{ij}},$$

with the l_{ij} as before. Now define $y_j = -\log z_j$, and we get

$$y_j = \sum_{i=1}^{n^2} x_i l_{ij}.$$