ECP
LSD
2015 - 2016

Image reconstruction in X-ray tomography

1 X-ray tomography

X-ray tomography reconstructs dense volumes of objects from a set of projections measured at different angles. The measurements $y \in \mathbb{R}^M$ and the sought absorption image $\overline{x} \in \mathbb{R}^N$ obey the linear relation

$$y = H\overline{x} + w, \tag{1}$$

where $w \in \mathbb{R}^M$ is the measurement noise, that we assume i.i.d. Gaussian with variance σ^2. The tomography matrix $H \in \mathbb{R}^{M \times N}$ is sparse and encodes the geometry of the measurements. Here, we will focus on the case when H models parallel projections of a 2-D object $\overline{x}$. Tomography measures are acquired at fixed and regularly sampled rotational positions between the sample and the detector so that H_{mn} models the intersection length between the mth light-ray and the nth pixel. If N_θ is the number of different angular positions of the detector in Fig. 1 and L the linear size of the detector, the number of measurements $M = L \times N_\theta$. In practice, the angular positions are regularly distributed on $[0, \pi)$.

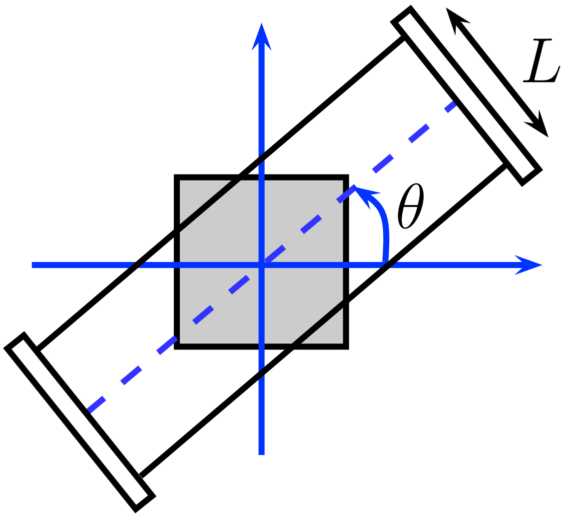

FIGURE 1 – Considered tomographic acquisition model

Traditional reconstruction methods such as the Filtered Back-Projection require the linear system (1) to be sufficiently determined for good results, i.e., $N_\theta \sim L$.

However, several applications could benefit from a smaller number of projections, either in order to reduce the total dose for medical applications, or to reduce the total acquisition time for in-situ experiments where the sample is evolving. Therefore, more sophisticated reconstruction approaches must be developed in order to overcome the under-determinacy of the problem and to make it robust to the presence of noise in the measurements.

1. Download the projection matrix H and the image $\overline{x}$ available on the website. Use `loadmat` from `scipy.io` in Python to load the arrays, note that H is stored as a sparse matrix.
2. Construct y, according to model (1), using $\sigma = 1$.
3. Here, $N = 90 \times 90$ pixels and $M = 90 \times 180$ measurements. Display a 2D version of x and a 2D version of y, also known as sinogram. To do so, in Matlab, use the `reshape` function ; in Python, use the `reshape` method with option `order='F'`.

2 Optimization problem

An efficient strategy to address the reconstruction problem is to define x as a minimizer of an appropriate cost function f. More specifically, we focus on the following penalized least-squares criterion :

$$(\forall x \in \mathbb{R}^N) \quad f(x) = \frac{1}{2}\|Hx - y\|^2 + \lambda r(x), \tag{2}$$

where r is a regularization function incorporating a priori assumptions to guarantee the robustness of the solution with respect to noise. In order to promote images formed by smooth regions separated by sharp edges, we set

$$(\forall x \in \mathbb{R}^N) \quad r(x) = \sum_{n=1}^{2N} \psi([Gx]^{(n)}), \tag{3}$$

where $G \in \mathbb{R}^{2N \times N}$ is a sparse matrix such that $Gx \in \mathbb{R}^{2N}$ is the concatenation of the horizontal and vertical gradients of the image, and ψ is a potential function defined as :

$$(\forall u \in \mathbb{R}) \quad \psi(u) = \sqrt{1 + u^2/\delta^2}, \tag{4}$$

with some parameter $\delta > 0$ aiming at guaranteeing the differentiability of r. In the following, we will set $(\lambda, \delta) = (0.13, 0.02)$.

1. Download the gradient operator G available in the website.
2. Give the expression of the gradient ∇f at some point $x \in \mathbb{R}^N$. Create a function which gives as an output the gradient of f at some input vector x.
3. Show that a Lipschitz constant of ∇f is

   $$L = \|H\|^2 + (\lambda/\delta^2)\|G\|^2.$$

   Calculate it for the (λ, δ) values given above. Note that, in Matlab, one can use `normest` to evaluate the norm of a sparse matrix ; in Python, the function `scipy.sparse.linalg.svds` gives the singular values of a sparse matrix, the maximal singular value being the norm of the matrix.

3 Optimization algorithms

3.1 Gradient descent algorithm

1. Create $x_0 \in \mathbb{R}^N$ a vector with all entries equal to 0. This will be our initialization for all tested algorithms.
2. Implement a gradient descent algorithm to minimize f.

3.2 MM quadratic algorithm

1. Construct, for all $x \in \mathbb{R}^N$, a quadratic majorant function of f at x. Create a function which gives, as an output, the curvature $A(x)$ of the majorant function at an input vector x.
   Hint : in Matlab, use `spdiags` to create a sparse diagonal matrix ; in Python, use `scipy.sparse.diags(d[:,0]).tocsc()` to create a sparse matrix from a diagonal vector $d \in \mathbb{R}^{n\times 1}$ using the compressed sparse column format. In addition, in Python, use the class `LinearOperator` from `scipy.sparse.linalg` to create the curvature operator.
2. Deduce a MM quadratic algorithm to minimize f. Implement it.
   Hint : in Matlab use `pcg` to invert the majorant matrix at each iteration ; in Python, use `bicg` from `scipy.sparse.linalg`.

3.3 3MG algorithm

The MM quadratic algorithm can be accelerated by using a subspace strategy. Here, we will focus on the so-called 3MG (MM Memory Gradient) approach which consists in defining the iterate x_{k+1} as the minimizer of the quadratic majorant function at x_k within a subspace spanned by the following directions :

$$(\forall k \in \mathbb{N}) \quad D_k = [-\nabla f(x_k) \quad | \quad x_k - x_{k-1}] \tag{5}$$

(with the convention $D_0 = -\nabla f(x_0)$). Thus, an iterate of 3MG reads :

$$(\forall k \in \mathbb{N}) \quad x_{k+1} = x_k + D_k u_k, \tag{6}$$

with

$$(\forall k \in \mathbb{N}) \quad u_k = -(D_k^\top A(x_k) D_k)^\dagger (D_k^\top \nabla f(x_k)), \tag{7}$$

where $A(x_k) \in \mathbb{R}^{N\times N}$ is the curvature of the majorant matrix at x_k and $\dagger$ denotes the pseudo-inverse operation.

1. Implement the 3MG algorithm.
   Hint : use `pinv` in Matlab and `scipy.linalg.pinv` in Python to compute the pseudo-inverse. Given the size of the matrices, mind the order of matrix multiplications, *e.g.* when computing $D_k^\top H^\top H D_k$ do not compute $D_k^\top (H^\top H) D_k$ but $(HD_k)^\top (HD_k)$.

3.4 Block-coordinate MM quadratic algorithm

Another acceleration strategy consists in applying a block alternation technique. The vector x is divided into $J \geq 1$ blocks, with size $1 \leq N_j \leq N$. At each iteration $k \in \mathbb{N}$, a block index $j \subset \{1, \ldots, J\}$ is chosen, and the corresponding components of x, denoted $x^{(j)}$, are updated, according to a MM quadratic rule. Here, we will assume that the blocks are selected in a cyclic manner, that is,

$$(\forall k \in \mathbb{N}) \quad j = \text{mod}(k-1, J) + 1. \tag{8}$$

For a given block index j, the corresponding pixel indexes are updated in the image :

$$n \in \mathbb{J}_j = \{N_j(j-1)+1, \ldots, jN_j\}. \tag{9}$$

1. Create a function which gives, as an output, matrix $A_j(x) \in \mathbb{R}^{N_j \times N_j}$ containing only the lines and rows of $A(x)$ with indexes $\mathbb{J}_j$.
2. Deduce an implementation of a block coordinate MM quadratic algorithm for minimizing f. Test it for $N_j = N/K$ with $K \in \{1, 2, 3, 5, 6, 9\}$.

3.5 Parallel MM quadratic algorithm

In order to benefit from the multicore structure of modern computer architecture, a parallel form of the MM quadratic algorithm is desirable. However, the quadratic majorizing function defined in Section 3.2 is not separable with respect to the entries of vector x so that its minimization cannot be performed efficiently in a parallel manner. Here, we propose an alternative construction, that possesses a better potential for parallelization.

1. For every $x \in \mathbb{R}^N$, let $B(x) \in \mathbb{R}^{N \times N}$ be a diagonal matrix with elements

   $$(\forall i \in \{1, \ldots, N\}) \;\; b^{(i)}(x) = \mathcal{H}^\top \mathbf{1} + \lambda \mathcal{G}^\top \left(\frac{\dot{\phi}(Gx)}{Gx} \right) \tag{10}$$

   with $\mathcal{H} \in \mathbb{R}^M$, $\mathcal{G} \in \mathbb{R}^{2N}$, and

   $$\mathcal{H}^{(m)} = |H^{(m,i)}| \sum_{p=1}^{N} |H^{(m,p)}|, \quad \text{and} \quad \mathcal{G}^{(n)} = |G^{(n,i)}| \sum_{p=1}^{N} |G^{(n,p)}|. \tag{11}$$

   Prove that, for every $x \in \mathbb{R}^N$, $A(x) \preceq B(x)$ where $A(\cdot)$ was defined in Section 3.2. Hint : use Jensen's inequality.
2. Deduce an implementation of a parallel MM quadratic algorithm or minimizing f.

3.6 Comparison of the methods

1. Create a function that computes the value of the criterion f along the iterations of the algorithm.

2. We will consider that the convergence is reached when the following stopping criterion is fulfilled :

$$\|\nabla f(x_k)\| \leq \sqrt{N} \times 10^{-4}. \tag{12}$$

What is the required time for each method to achieve this condition ? For each method, plot the decrease of $(f(x_k))_{k \in \mathbb{N}}$ versus time until the stopping criterion is satisfied.

3. The Signal to Noise Ration (SNR) of a restored image $\widehat{x}$ is defined as

$$\mathrm{SNR} = 10 \log_{10} \left(\|\overline{x}\|^2 / \|\overline{x} - \widehat{x}\|^2 \right) \tag{13}$$

Using the fastest method, search for parameters (λ, δ) that optimize the SNR.