LSD TP3 Report : Image reconstruction in X-ray tomography

Imen AYADI

December 2020

Abstract

In this lab, we study the computed tomography, an image processing technique based on high-frequency absorption properties commonly used to detect medical anomalies. From a 2D sinogram y, we reconstruct the absorption image $\overline{x}$ by formulating the problem as a minimization of a square cost with a penalization term. To solve this optimization problem, we test several optimization algorithms and we compare their performance in term of computational time.

Notations

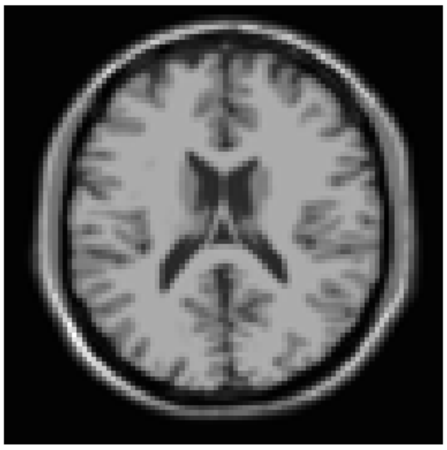
1. If $g : \mathbb{R} \longrightarrow \mathbb{R}$ denotes a scalar function, we can extend it to a vector function: if $x = (x_1, .., x_N)^T \in \mathbb{R}^N$,then, we define $g(x) = (g(x_1), ..., g(x_N)) \in \mathbb{R}^N$. we use the same name of the function.
2. If $x = (x_1, .., x_N)^T \in \mathbb{R}^N$,then,we define $diag(x)$ as the diagonal matrix $\in \mathbb{R}^{N\times N}$ whose diagonal terms are the coefficients of x.
3. Let $(e_1, .., e_{2N})$ be the canonical basis of $\mathbb{R}^{2N}$. So, for $u = (u_1, .., u_{2N})^T \in \mathbb{R}^{2N}$, $\langle u, e_n\rangle_{\mathbb{R}^{2N}} = u_n$ for $1 \leq n \leq 2N$ and $u = \sum_{n=1}^{2N} u_n e_n$

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:

$$y = H\overline{x} + w$$

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.
For $N = 90 \times 90$ pixels and $M = 90 \times 180$ measurements, we display a 2D version of $\overline{x}$ and a 2D version of y, also known as sinogram. The results are plotted in Figure 1.


(a) The real absorption image $\overline{x}$

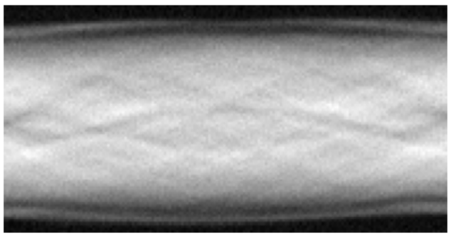

(b) The sinogram y

Figure 1: Displaying a 2D version of $\overline{x}$ and y

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 :

$$f(x) := \frac{1}{2}||Hx - y||_2^2 + \lambda r(x) \tag{1}$$

where the regularization function is given by:

$$r(x) = \sum_{n=1}^{2N} \Psi([Gx]^{(n)})$$

with $\Psi(u) = \sqrt{1 + (\frac{u}{\delta})^2}$, $\delta > 0$ and $G \in \mathbb{R}^{2N \times N}$ the gradient operator matrix.
Let $\rho : x \mapsto ||Hx-y||_2^2$. It is obvious that ρ is twice differentiable: $\forall x \in \mathbb{R}^N$, $\nabla\rho(x) = H^T(Hx-y)$ and $\nabla^2\rho(x) = H^TH$.
Now, we are interested in differentiating r. We notice that $r = \sum_{n=1}^{2N} \Psi \circ p_n(x)$ where $p_n : x \in \mathbb{R}^N \mapsto \langle Gx, e_n\rangle_{\mathbb{R}^{2N}} = x^TG^Te_n$. The function p_n is differentiable and its first derivative is $\nabla p_n(x) = G^Te_n$.
Ψ and p_n are differentiable then, as a sum of compositions of differentiable functions,r is differentiable such that:

$$\nabla r(x) = \sum_{n=1}^{2N} \Psi'(p_n(x))\nabla p_n(x) = G^T \sum_{n=1}^{2N} \Psi'([Gx]^{(n)})e_n = G^T\Psi'(Gx)$$

Since $f = \rho + \lambda r$, then, f is differentiable and $\forall x \in \mathbb{R}^N$

$$\boxed{\nabla f(x) = H^T(Hx - y) + \lambda G^T\Psi'(Gx)} \tag{2}$$

An efficient method to show that ∇f is a Lipschitz function is to derive the hessian of f and demonstrate that it is bounded. For that, we notice that $\nabla r(x)$ is differentiable by composition and $\nabla^2 r(x) = G^T Jac_{\tilde{\Psi}}(x)$ where $\tilde{\Psi} : x \in \mathbb{R}^N \mapsto \Psi'(Gx)$ and $Jac_{\tilde{\Psi}}(x) \in \mathbb{R}^{N\times 2N}$ is its Jacobian at point $x \in \mathbb{R}^N$. For $(m,n) \in [\![1, 2N]\!] \times [\![1, 2N]\!]$, :

$$\begin{aligned}
[Jac_{\tilde{\Psi}}(x)]^{(m,n)} &= \frac{\partial \Psi'([Gx]^{(m)})}{\partial x_n} \\
&= \frac{\partial \Psi'(\sum_{k=1}^{N}[G]^{(m,k)}x_k)}{\partial x_n} \\
&= [G]^{(m,n)}\Psi''([Gx]^{(m)}) \\
&= [diag(\Psi''(Gx))]^{(m,m)}[G]^{(m,n)} \\
&= [diag(\Psi''(Gx))G]^{(m,n)}
\end{aligned} \tag{3}$$

and it follows that $Jac_{\tilde{\Psi}}(x) = diag(\Psi''(Gx))G$ Hence , $\nabla^2 r(x) = G^Tdiag(\Psi''(Gx))G$. Finally, we get:

$$\boxed{\nabla^2 f(x) = H^TH + \lambda G^Tdiag(\Psi''(Gx))G} \tag{4}$$

For $t \in \mathbb{R}$, $\Psi''(t) = \frac{1}{\delta^2}(1 + \frac{t^2}{\delta^2})^{\frac{-3}{2}} \leq \frac{1}{\delta^2}$.Then,

$$||\nabla^2 f(x)u|| \leq ||H^THu|| + \lambda||G^Tdiag(\Psi''(Gx))Gu|| \leq ||H^TH||\ ||u|| + \frac{\lambda}{\delta^2}||G^TG||\ ||u||$$

where we used the fact that $||G^Tdiag(\Psi''(Gx))G|| \leq ||G^T||\ ||diag(\Psi''(Gx))||\ ||G|| \leq \frac{1}{\delta^2}||G^T||\ ||G|| = \frac{1}{\delta^2}||G^TG||$.
Then, $||\nabla^2 f(x)|| \leq ||H^TH|| + \frac{\lambda}{\delta^2}||G^TG||$

Let $\boxed{L := ||H^TH|| + \frac{\lambda}{\delta^2}||G^TG|| = ||H||^2 + \frac{\lambda}{\delta^2}||G||^2}$.
Using the mean value theorem, we deduce that ∇f is an L-Lipschitz function.

- **<u>Remark:</u>** Notice that $\forall u \in \mathbb{R}^N$, $u^T\nabla^2 f(x)u = ||Hu||_2^2 + \lambda||diag(\sqrt{\psi''(Gx)}\ )Gu||_2^2 \geq ||Hu||_2^2 \geq 0$. Consequently, f is convex. Using the notebook, I tested if 0 belongs to the set of the singular values of H : this is not the case, which means that H is invertible [i]. Then, for each $u \neq 0, ||Hu|| > 0$. So, f is strictly-convex. Hence, f admits at most one minimizer.
  Moreover, because r is positive, $f(x) \geq \frac{1}{2}||Hx - y||^2 \geq \frac{1}{2}(||Hx|| - ||y||)^2$. Let $(h_1, .., h_N)$ be an orthonormal basis of $\mathbb{R}^N$ that diagonalizes H^TH and $0 < \nu_1 \leq ... \leq \nu_N$ the eigenvalues of H.
  Since $||Hx||^2 = x^T(H^THx) = \langle x, H^THx\rangle = \langle\sum_{n=1}^{N}\langle x, f_n\rangle f_n, \sum_{n=1}^{N}\langle x, f_n\rangle H^THf_n\rangle = \langle\sum_{n=1}^{N}\langle x, f_n\rangle f_n, \sum_{n=1}^{N}\langle x, f_n\rangle\nu_nf_n\rangle$
  $\sum_{n=1}^{N}\nu_n(\langle x, f_n\rangle)^2 \geq \nu_1||x||^2$. Then, $f(x) \geq \frac{1}{2}(\nu_1||x||^2 - ||y||^2) \underset{||x||\to+\infty}{\longrightarrow} +\infty$. Thus, f is coercive.
  Therefore, we conclude that f has a unique minimizer.

[i] if $0 \leq \sigma_1 \leq ... \leq \sigma_N$ are the singular values of H ,then $\exists U \in R^{M\times N}$, $V \in \mathbb{R}^{N\times N}$ orthogonal matrices and $\Sigma = diag(\sigma_1, .., \sigma_N)$ such that $H = U\Sigma V^T$. Then,$H^TH = V\Sigma^2V^T$ .Thus, the singular of H are the square-root of H^TH.If H non invertible, then, $\exists x \neq 0$ such that $Hx = 0$. Then, $H^THx = 0$ i.e. H^TH is non invertible. So, 0 belongs to the spectrum of H^TH. Hence, 0 is a singular value of H

3 Optimization algorithms

3.1 Gradient descent algorithm

We have proved that $f \in \Gamma_0(\mathbb{R}^N)$, f is differentiable function with a L-Lipschitzian gradient and $Argmin\ f \neq \emptyset$. Therefore, we can use the gradient descent algorithm with a fixed step-size γ such that $0 < \gamma < \frac{2}{L}$.
Each iteration requires the computation of $\nabla f(x_k)$ that needs $O(N^3)$ operations.

Algorithm 1: Gradient descent Algorithm

```
Result: $x_n$
initialization : $x_0 \in \mathbb{R}^N$ , $\gamma \in ]0, \frac{2}{L}[$
while not stopping criterion do
    $x_{k+1} = x_k - \gamma \nabla f(x_k)$
```

- **Convergence Analysis:** An interesting property about f that help study the convergnce rate of the algorithm is its strong convexity: we can notice easily that $\forall x, u \in \mathbb{R}^n$, $u^T \nabla^2 f(x) u \geq ||Hu||^2 \geq m||u||^2$ where m is the smallest eigenvalue of $H^T H$ (which corresponds to the smallest square of singular values of H). Then, $0 \preceq_{S_N^+} \nabla^2 f - m I_N$. Therefore, $j : x \in \mathbb{R}^N \mapsto f(x) - \frac{m}{2}||x||_2^2$ is convex (because its hessian is positive semi-definite) i.e. for $x, y \in \mathbb{R}^N$, $j(y) \geq j(x) + \nabla j(x)^T (y - x)$,then,

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{m}{2}||x - y||_2^2 \tag{5}$$

Such property helps us assess the convergence speed of the gradient descent algorithm. To see that, first, we apply the descent lemma[ii] for $y = x_{k+1}$ and $x = x_k$ to get:

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \nabla f(x_k)^T (x_{k+1} - x_k) + \frac{L}{2}||x_{k+1} - x_k||_2^2 \\ &\leq f(x_k) + \nabla f(x_k)^T (-\gamma \nabla f(x_k)) + \frac{L}{2}||\gamma \nabla f(x_k)||_2^2 \\ &\leq f(x_k) - \gamma \left(1 - \frac{L\gamma}{2}\right) ||\nabla f(x_k)||_2^2 \end{aligned}$$

Therefore,

$$f(x_{k+1}) - f(x^*) \leq f(x_k) - f(x^*) - \gamma \left(1 - \frac{L\gamma}{2}\right) ||\nabla f(x_k)||_2^2 \tag{6}$$

where $x^* = argmin(f)$.
On other hand, we apply Equation (5) to get:

$$\begin{aligned} f(x^*) &= min_{y \in \mathbb{R}^N} f(y) \\ &\geq min_{y \in \mathbb{R}^N} f(x_k) + \nabla f(x)^T (y - x_k) + \frac{m}{2}||y - x_k||_2^2 \\ &= f(x_k) - \frac{1}{2m}||\nabla f(x)||_2^2 \quad \text{(because the min is reached at } y = x_k - \tfrac{1}{m}\nabla f(x_k)) \end{aligned}$$

Then,

$$||\nabla f(x_k)||_2^2 \geq 2m(f(x_k) - f(x^*)) \tag{7}$$

Using Equation (6) and Equation (9), we obtain that $f(x_{k+1}) - f(x^*) \leq c_\gamma (f(x_k) - f(x^*))$ with $c_\gamma := 1 - 2m\gamma \left(1 - \frac{L\gamma}{2}\right)$. Consequently,

$$\forall k \in \mathbb{N}, \quad f(x_k) - f(x^*) \leq c_\gamma^k (f(x_0) - f(x^*)) \tag{8}$$

Since $\gamma < \frac{2}{L}$and $m < L$, then, $0 < 2m\gamma \left(1 - \frac{L\gamma}{2}\right) < 1$ i.e. $0 < c_\gamma < 1$.
If we want the stopping criterion on $||\nabla f(x_k)||_2$, we can use the fact that:

$$||\nabla f(x_k)||_2^2 \leq 2L(f(x_k) - f(x^*) \tag{9}$$

To see that, we proceed as in Equation (9):

$$\begin{aligned} f(x^*) &= min_{y \in \mathbb{R}^N} f(y) \\ &\leq min_{y \in \mathbb{R}^N} f(x_k) + \nabla f(x)^T (y - x_k) + \frac{L}{2}||y - x_k||_2^2 \\ &= f(x_k) - \frac{1}{2L}||\nabla f(x)||_2^2 \quad \text{(because the min is reached at } y = x_k - \tfrac{1}{L}\nabla f(x_k)) \end{aligned}$$

Thus, we finally get :

$$\forall k \in \mathbb{N}, \quad ||\nabla f(x_k)||_2^2 \leq 2L\ c_\gamma^k (f(x_0) - f(x^*)) \tag{10}$$

[ii]for $x, y \in \mathbb{R}^N, f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2}||x - y||_2^2$

- **Complexity analysis**: If we want a ϵ-precision on $||\nabla f(x_k)||$, we need at most $\frac{log\left(\frac{\epsilon^2}{2L(f(x_0)-f(x^*))}\right)}{log(c_\gamma)}$ iterations. To reduce the complexity, we can maximize c_γ by choosing $\gamma_0 = \frac{1.99}{L}$.
  For $\epsilon = \sqrt{N}10^{-4}$, then, $\frac{log\left(\frac{\epsilon^2}{2L(f(x_0)-f(x^*))}\right)}{log(c_\gamma)} = O(log(N))$.Then, the total complexity of the gradient descent algorithm is majorized by $O(N^3 log(N))$.
  To be more concrete, we find for $N = 190 \times 190$, $\frac{log\left(\frac{\epsilon^2}{2L(f(x_0)-f(x^*))}\right)}{log(c_\gamma)} \approx 8000$ and When we run the code for the same stopping criterion, we obtain 2334 iterations, which is of the same order of magnitude of the theoretical estimation.

3.2 MM quadratic algorithm

The first step to build our MM quadratic algorithm is to derive a quadratic majorant function h of f a given point at $x_0 \in \mathbb{R}^N$. Hence, we aim to find a matrix $A(x_0)$ such that $\forall x \in \mathbb{R}^N$:

$$h(x, x_0) = f(x_0) + \nabla f(x_0)^T(x - x_0) + \frac{1}{2}(x - x_0)^T A(x_0)(x - x_0) \tag{11}$$

We notice that the function $\phi : u \mapsto \sqrt{1 + \frac{u}{\delta^2}}$ is concave over $\mathbb{R}_+$ because $\phi''(u) = \frac{-1}{4\delta^4}\left(1 + \frac{u}{\delta^2}\right)^{\frac{3}{2}} < 0 \ \ \forall u \in \mathbb{R}_+$. Then, for each $(x, y) \in \mathbb{R}_+^2$, we have : $\phi(y) \leq \phi(x) + \phi'(x)(y - x)$.
Let $(u, v) \in \mathbb{R}^2$. if we take $y = v^2$ and $x = u^2$, then, we get:

$$\sqrt{1 + \frac{v^2}{\delta^2}} \leq \sqrt{1 + \frac{u^2}{\delta^2}} + (v^2 - u^2)\frac{1}{2\delta^2}\frac{1}{\sqrt{1 + \frac{u^2}{\delta^2}}}$$

Notice that $v^2 - u^2 = (v - u)^2 + 2u(v - u)$. So, we get finally that $\forall (u, v) \in \mathbb{R}^2$:

$$\Psi(v) \leq \Psi(u) + \Psi'(u)(v - u) + \frac{1}{2}w(u)(v - u)^2 \tag{12}$$

where $w(u) = \frac{1}{\delta^2}\frac{1}{\sqrt{1+\frac{u^2}{\delta^2}}}$.
We apply Equation (12) for $u = [Gx_0]^{(n)}$ and $v = [Gx]^{(n)}$:

$$\Psi([Gx]^{(n)}) \leq \Psi([Gx_0]^{(n)}) + \Psi'([Gx_0]^{(n)})([Gx]^{(n)} - [Gx_0]^{(n)}) + \frac{1}{2}w([Gx_0]^{(n)})([Gx]^{(n)} - [Gx_0]^{(n)})^2 \tag{13}$$

Then, summing over n:

$$r(x) \leq r(x_0) + \sum_{n=1}^{2N} \Psi'([Gx_0]^{(n)})[G(x - x_0)]^{(n)} + \frac{1}{2}\sum_{n=1}^{2N} w([Gx_0]^{(n)})([G(x - x_0)]^{(n)})^2 \tag{14}$$

We notice that

$$\Psi'([Gx_0]^{(n)})\ [G(x-x_0)]^{(n)} = \langle \Psi'(Gx_0), G(x-x_0)\rangle_{\mathbb{R}^{2N}} = \Psi'(Gx_0)^T G(x-x_0) = (G^T\Psi'(Gx_0))^T(x-x_0) = \nabla r(x_0)^T(x-x_0)$$

and,

$$\sum_{n=1}^{2N} w([Gx_0]^{(n)})([G(x - x_0)]^{(n)})^2 = \sum_{n=1}^{2N}[diag(w(Gx_0))]^{(n)}([G(x - x_0)]^{(n)})^2 = (G(x - x_0))^T\ diag(w(Gx_0))\ G(x - x_0)$$

Therefore,

$$r(x) \leq r(x_0) + \nabla r(x_0)^T(x - x_0) + \frac{1}{2}(x - x_0)^T G^T diag(w(Gx_0))G(x - x_0) \tag{15}$$

For another part, using the fact that $||u||_2^2 - ||v||_2^2 = \langle u - v, u + v\rangle$, we have :

$$\begin{aligned}
\rho(x) - \rho(x_0) - \nabla\rho(x_0)^T(x - x_0) &= \frac{1}{2}||Hx - y||_2^2 - \frac{1}{2}||Hx_0 - y||_2^2 - ((Hx - y)^T H)(x - x0) \\
&= \frac{1}{2}\langle H(x - x_0), H(x + x_0) - 2y\rangle - \langle H(x - x_0), Hx - y\rangle \\
&= \frac{1}{2}\langle H(x - x_0), H(x - x_0)\rangle \\
&= \frac{1}{2}(x - x_0)^T H^T H(x - x_0)
\end{aligned} \tag{16}$$

since $f = \rho + \lambda r$, then:

$$f(x) \leq f(x_0) + \nabla f(x_0)^T (x - x_0) + \frac{1}{2}(x - x_0)^T \left(H^T H + \lambda G^T diag(w(Gx_0))G\right)(x - x_0) \tag{17}$$

So, we can choose $\boxed{A(x_0) = H^T H + \lambda G^T diag(w(Gx_0))G}$.
Notice that $\forall x_0 \in \mathbb{R}^N,\ A(x_0)$ is invertible. In fact:

$$\begin{aligned}
A(x_0)v = 0 &\implies v^T A(x_0)v = 0 \\
&\implies ||Hv||_2^2 + \lambda ||diag(\sqrt{w(Gx_0)})Gv||_2^2 = 0 \\
&\implies ||Hv|| = 0 \text{ and } ||diag(\sqrt{w(Gx_0)})Gv|| = 0 \\
&\implies Hv = 0 \text{ and } diag(\sqrt{w(Gx_0)})Gv = 0 \\
&\implies v = 0 \quad \text{(because } H \text{ is invertible)}
\end{aligned}$$

Recall finally that $argmin_{x \in \mathbb{R}^N} h(x, x_0) = x_0 - A(x_0)^{-1} \nabla f(x_0)$
We have already all the required assumptions by the MM algorithm:

- f is a coercive, differentiable function.
- $\forall x \in \mathbb{R}^N$, $mI_N \preceq A(x) \preceq LI_N$ where m is the smallest eigenvalue of $H^T H$ and L is the constant defined in Section 2.

Thus, the scheme of the classical MM algorithm is given by:

Algorithm 2: Quadratic MM Algorithm

```
**Result:** $x_k$
initialization : $x_0 \in \mathbb{R}^N$
**while** *not stopping criterion* **do**
    $x_{k+1} = x_k - A(x_k)^{-1} \nabla f(x_k)$
```

At each iteration, we compute the matrix $A(x_n)$ (this requires the calculus of $(w([Gx_n]^{(i)})_{1 \leq i \leq 2N}$ and two matritial multiplication) then we invert it [iii]: in total, each iteration requires $O(N^3)$ operations.
We have shown in the course that$\forall k \in \mathbb{N},\ f(x_k) - f(x_{k+1}) \geq \mu_2 ||\nabla f(x_k)||_2^2$ where $\mu_2 := \frac{m}{2L^2}$.
Using Equation (9), we get $f(x_{k+1}) - f(x^*) \leq (1 - 2m\mu_2) f(x_k) - f(x^*) = (1 - \frac{m^2}{L^2}) f(x_k) - f(x^*)$. Hence,

$$f(x_k) - f(x^*) \leq \left(1 - \frac{m^2}{L^2}\right)^k (f(x_0) - f(x^*)) \tag{18}$$

Finally, we get:

$$||\nabla f(x_k)||_2^2 \leq \frac{2L^2}{m} \left(1 - \frac{m^2}{L^2}\right)^k (f(x_0) - f(x^*)) \tag{19}$$

- <u>**Complexity analysis :**</u> If we want a ϵ-precision on $||\nabla f(x_k)||$ where $\epsilon = \sqrt{N} 10^{-4}$, we need at most $\frac{log\left(\frac{m\epsilon^2}{2L^2(f(x_0) - f(x^*))}\right)}{log\left(1 - \frac{m^2}{L^2}\right)} = O(log(N))$ iterations.

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 :

$$\forall k \in \mathbb{N}, D_k = [-\nabla f(x_k) | x_k - x_{k-1}] \in \mathbb{R}^{N \times 2}$$

(with the convention $D_0 = -\nabla f(x_0)$.
Thus, an iterate of 3MG reads : $\forall k \in \mathbb{N}, x_{k+1} = x_k + D_k u_k$ with $u_k = -(D_k^T A(x_n) D_n)^\dagger (D_n^T \nabla f(x_n))$ where $A(x_n) \in \mathbb{R}^{N \times N}$ is the curvature of the majorant matrix at x_n and $\dagger$ denotes the pseudo-inverse operation.
Such that, the computational cost of the matrix inversion is reduced from N^3 to 2^3.

Algorithm 3: 3MG Algorithm

```
**Result:** $x_k$
initialization : $x_0 \in \mathbb{R}^N$
**while** *not stopping criterion* **do**
    $D_k = [-\nabla f(x_k) | x_k - x_{k-1}]$
    $u_k = -(D_k^T A(x_n) D_n)^\dagger (D_n^T \nabla f(x_n))$
    $x_{k+1} = x_k + D_k u_k$
```

[iii]To reduce the complexity, we exploit sparsity of the matrices

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 $n \in N$, a block index $j \in [\![0, J-1]\!]$ 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 : $\forall n \in \mathbb{N}, j = n \ mod(J)$ .In fact, the arrays are indexed from index 0 in Python, whereas they are indexed from 1 in the original subject. Therefore, I adapted the formulas presented in the lab subject such that the indexation starts from 0 .
For a given block index j, the corresponding pixel indexes are updated in the image : $n \in \mathcal{G}_j := \{jN_j, ..., (j+1)N_j - 1\}$
Here, we consider blocks with the same size i.e. $N_j = N//J$.
We have $A_j(x_k) \in \mathbb{R}^{N_j \times N_j}$. Therefore, computing its inverse requires $O(N_j^3) = O(\frac{N^3}{J^3})$. Thus, this strategy divides the computational cost for an iteration of the basic quadratic MM algorithm by J^3. However, we need much more iterations to reach the stopping criterion when J increases because we update less coefficients of x_k and then we are farther from the correct full update, which worsen the expected performance of the block coordinate strategy. That is why we tested only small values of J. To verify this fact, I tested $J = 100$ and the convergence was dramatically slow.

- **Remark:** Unlike the other mentioned algorithms, Block-coordinate MM algorithm is has not a monotonic update: no guarantee that at each iteration $f(x_{k+1}) \leq f(x_k)$.

Algorithm 4: Block-coordinate MM Algorithm

```
Result: $x_k$
initialization : $x_0 \in \mathbb{R}^N$
while not stopping criterion do
    $H_j = H[:, : \mathcal{G}_j]$
    $G_j = G[:, : \mathcal{G}_j]$
    $A_j(x_k) = H_j^T H_j + \lambda G_j^T diag(w(Gx_k))G_j$
    $x_{k+1}[\mathcal{G}_j] = x_k[\mathcal{G}_j] - A(x_n)^{-1} \nabla f(x_k)[\mathcal{G}_j]$
```

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 majorant function defined in Equation (11) 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. For that, we introduce the diagonal matrix $B(x) = diag(b(x))$ where $b(x) \in \mathbb{R}^N$ is defined as:

$$\forall i \in [\![1, N]\!],\ b(x)^{(i)} = \mathcal{H}_i^T \mathbf{1} + \lambda \mathcal{G}_i^T \left(\frac{\Psi'(Gx)}{Gx} \right) \tag{20}$$

such that $\left(\frac{\Psi'(Gx)}{Gx}\right) = \left(\frac{\Psi'([Gx]^{(n)})}{[Gx]^{(n)}}\right)_{1 \leq n \leq 2N} = \left(w([Gx]^{(n)})\right)_{1 \leq n \leq 2N} \in \mathbb{R}^{2N}$ [iv] and $\mathcal{H} \in \mathbb{R}^M$ and $\mathcal{G} \in \mathbb{R}^{2N}$ are given by:

$$\begin{aligned} \forall m \in [\![1, M]\!],\ \mathcal{H}_i^{(m)} &= |H^{(m,i)}| \sum_{p=1}^{N} |H^{(m,p)}| \\ \forall n \in [\![1, 2N]\!],\ \mathcal{G}_i^{(n)} &= |G^{(n,i)}| \sum_{p=1}^{N} |G^{(n,p)}| \end{aligned} \tag{21}$$

It is obvious that $\forall x \in \mathbb{R}^N,\ \forall i \in [\![1, N]\!],\ b(x)^{(i)} \geq 0$.
Suppose by absurd that there exits $x \in \mathbb{R}^N$ and $j\ in [\![1, N]\!]$ such that $b(x)^{(j)} = 0$. Then:

$$\begin{aligned} b(x)^{(j)} = 0 &\longrightarrow \sum_{m=1}^{M}\sum_{p=1}^{N} |H^{(m,j)}||H^{(m,p)}| + \lambda \sum_{n=1}^{2N}\sum_{p=1}^{N} |G^{(n,j)}||G^{(n,p)}| \left(\frac{\Psi'([Gx]^{(n)})}{[Gx]^{(n)}}\right) = 0 \\ &\longrightarrow \forall (m,n,p) \in [\![1, M]\!] \times [\![1, 2N]\!] \times [\![1, N]\!],\ |H^{(m,j)}||H^{(m,p)}| = 0 \text{ and } |G^{(n,j)}||G^{(n,p)}| = 0 \text{ (in fact, } \tfrac{\Psi'([Gx]^{(n)})}{[Gx]^{(n)}} > 0) \\ &\longrightarrow \forall (m,n) \in [\![1, M]\!] \times [\![1, 2N]\!],\ H^{(m,j)} = 0 \text{ and } G^{(n,j)} = 0 \ \text{ (in fact, we take in particular } p = j) \\ &\longrightarrow \text{the } j^{th}\text{column of } H \text{ and } j^{th}\text{column of } G \text{ are null, which is impossible because } H \text{ and } G \text{ are invertible} \end{aligned}$$

(verified with `Python`)

[iv] for $x = 0$, we take for $\frac{\Psi'([0]^{(n)})}{[0]^{(n)}}$ the limit of $\frac{\Psi'([u]^{(n)})}{[u]^{(n)}}$ at 0 i.e. $\frac{\Psi'([0]^{(n)})}{[0]^{(n)}} = 1$

Therefore, $\forall x \in \mathbb{R}^N,\ \forall i \in [\![1,N]\!],\ b(x)^{(i)} > 0$. In particular, $B(x) \in \mathcal{S}_N^{++}$.
So, if we manage to demonstrate that $A(x) \preceq_{S_N^+} B(x)$, we can consider as quadratic majorant function

$$\tilde{h} : (x,y) \in \mathbb{R}^N \mapsto f(y) + \nabla f(y)^T (x-y) + \frac{1}{2}(x-y)^T B(y)(x-y)$$

since $A(x) \preceq_{S_N^+} B(x)$ implies $\tilde{h}(x,y) \geq h(x,y) \ \ \forall x, y \in \mathbb{R}^N$.
Notice that it is computationally easy to invert $B(x)$ because it is diagonal : in fact, $B(x)^{-1} = diag(1/b(x))$ which requires $O(N)$ operations at each iteration because the vectors $\mathcal{H}_i$ and $\mathcal{G}_i$ do not depend on x and we can compute then before starting the loop and we need only at each iteration to calculate $\frac{\Psi'([Gx]^{(n)})}{[Gx]^{(n)}} = w([Gx]^{(n)})$ for $n \in [\![1, 2N]\!]$ (recall that we needed $O(N^3)$ operations per iteration to invert $A(x)$). Therefore, it is evident that this new quadratic majorant function is advantageous with respect to the one defined in Equation (11).
Now, let us prove that $\forall x \in \mathbb{R}^N,\ A(x) \preceq_{S_N^+} B(x)$ i.e. $\forall u \in \mathbb{R}^N,\ u^T(A(x) - B(x))u \geq 0$
Let $x, u \in \mathbb{R}^N$. We have that :

$$\begin{aligned}
u^T(A(x) - B(x))u &= ||Hu||_2^2 + \lambda ||diag(\sqrt{w(Gx)})Gu||_2^2 - \sum_{i=1}^{N} b(x)^{(i)} u^{(i)2} \\
&= (\sum_{m=1}^{M}([Hu]^{(m)})^2 + \lambda \sum_{n=1}^{2N} w([Gx]^{(n)})([Gu]^{(n)})^2) - \sum_{i=1}^{N}\left(\sum_{m=1}^{M}\sum_{p=1}^{N}|H^{(m,i)}||H^{(m,p)}|(u^{(i)})^2\right. \\
&\qquad \left. + \lambda \sum_{n=1}^{2N}\sum_{p=1}^{N}|G^{(n,i)}||G^{(n,p)}|w([Gx]^{(n)})(u^{(i)})^2\right) \\
&= \sum_{m=1}^{M}\left[([Hu]^{(m)})^2 - \sum_{p,i=1}^{N}|H^{(m,i)}||H^{(m,p)}|(u^{(i)})^2\right] + \lambda \sum_{n=1}^{2N} w([Gx]^{(n)}) \Big[([Gu]^{(n)})^2 \\
&\qquad - \sum_{p,i=1}^{N}|G^{(n,i)}||G^{(n,p)}|(u^{(i)})^2\Big]
\end{aligned} \tag{22}$$

For that, we start by demonstrating the following lemma:

Lemma 3.1. *Let $g : \mathbb{R} \longrightarrow \mathbb{R}$ a convex function and $P \in \mathbb{N}^*$. Then, for all $x^{(1)}, ..., x^{(P)}, c^{(1)}, .., c^{(P)} \in \mathbb{R}$ and $w^{(1)}, ..., w^{(P)} \in [0,1]$ such that $\sum_{i=1}^{P} w^{(p)} = 1$:*

$$g\left(\sum_{i=1}^{P} c^{(i)} x^{(i)}\right) \leq \sum_{i=1}^{P} w^{(i)} g\left(\frac{c^{(i)}x^{(i)}}{w^{(i)}}\right)$$

Proof.

$$\begin{aligned}
g\left(\sum_{i=1}^{P} c^{(i)} x^{(i)}\right) &= g\left(\sum_{i=1}^{P} w^{(i)} \frac{c^{(i)}x^{(i)}}{w^{(i)}}\right) \\
&\leq \sum_{i=1}^{P} w^{(i)} g\left(\frac{c^{(i)}x^{(i)}}{w^{(i)}}\right) \quad \text{(using Jensen's inequality)}
\end{aligned} \tag{23}$$

■

First, we apply the Lemma 3.1 for $g_m : t \mapsto t^2$, $P = N$, $x^{(i)} = u^{(i)}$,$c^{(i)} = H^{(m,i)}$ and $w^{(i)} = \frac{|H^{(m,i)}|}{\sum_{p=1}^{N}|H^{(m,p)}|}$.So, we get:

$$\begin{aligned}
([Hu]^{(m)})^2 &= g_m([Hu]^{(m)}) \\
&= g(\sum_{i=1}^{N} H^{(m,i)} u^{(i)}) \\
&\leq \sum_{i=1}^{N}\left(\frac{|H^{(m,i)}|}{\sum_{p=1}^{N}|H^{(m,p)}|}\right) g\left(\frac{H^{(m,i)}u^{(i)}}{\frac{|H^{(m,i)}|}{\sum_{j=1}^{N}|H^{(m,j)}|}}\right) \\
&= \sum_{p=1}^{N}|H^{(m,p)}| \sum_{i=1}^{N}|H^{(m,i)}|(u^{(i)})^2 \\
&= \sum_{p,i=1}^{N}|H^{(m,i)}||H^{(m,p)}|(u^{(i)})^2
\end{aligned} \tag{24}$$

We proceed similarly by taking $g_n : t \mapsto t^2$, $P = N$, $x^{(i)} = u^{(i)}$,$c^{(i)} = G^{(n,i)}$ and $w^{(i)} = \frac{|G^{(n,i)}|}{\sum_{p=1}^{N} |G^{(n,p)}|}$.So, we get:

$$([Gu]^{(n)})^2 \leq \sum_{p,i=1}^{N} |G^{(n,i)}||G^{(n,p)}|(u^{(i)})^2 \tag{25}$$

Therefore, by summing up Equation (24) for $m := 1$ to M and by multiplying Equation (25) by $w([Gx]^{(n)})$ (which is > 0) then summing up it for $n := 1$ to $2N$, we get using Equation (22) that:

$$\forall x \in \mathbb{R}^N, \quad A(x) \preceq_{S_N^+} B(x) \tag{26}$$

Algorithm 5: Parallel MM Algorithm

```
Result: $x_n$
initialization : $x_0 \in \mathbb{R}^N$
while not stopping criterion do
    $x_{k+1} = x_k - B(x_k)^{-1} \nabla f(x_k)$
```

3.6 Comparison of the methods

I tested all the previously-mentioned algorithms for the same stopping criterion $||\nabla f(x_k)|| \leq \epsilon$ where $\epsilon = \sqrt{N}10^{-4}$. I plotted in the same Figure 2 the decrease of $f(x_k)$ versus time until the stopping criterion is satisfied. To get this plot, I assumed that for a given algorithm, all iterations have the same executive time. To compute it, I divided the running time of the whole algorithm by the number of the required iterations.
It would be more rigorous to run each algorithm for different initializations to test the impact of x_0 (actually the smoothness of f around x_0) on the running time.
It is obvious that the fastest method is **3MG algorithm**.

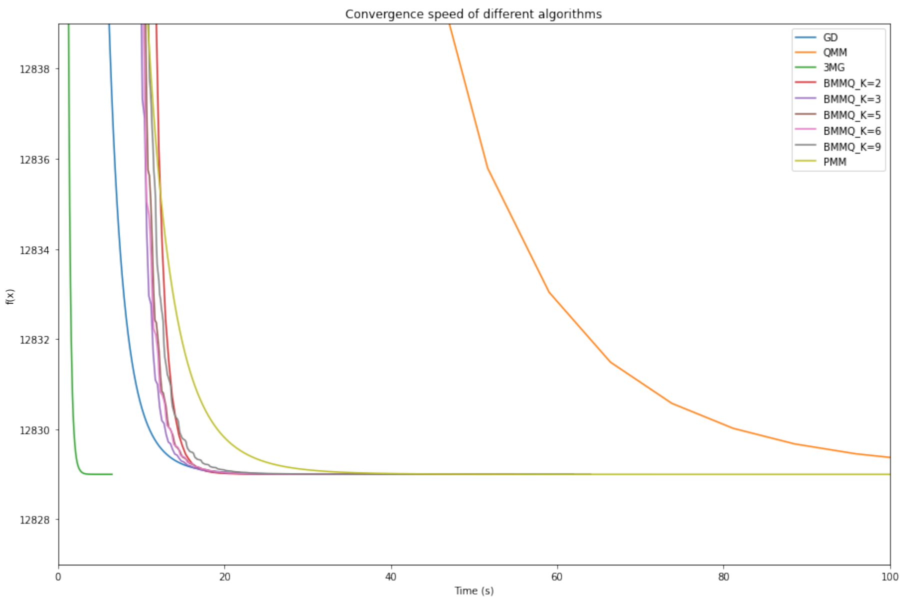


Figure 2: The evolution of $f(x_k)$ as a function of time

Now,using the fastest method, we search for parameters (λ, δ) that optimize the SNR. A grid search yields $\lambda_{opt} = 0,1$ and $\delta_{opt} = 0,01$ with SNR$= 19,4413$.