Faculté des Sciences et Techniques
Master ACSYON

Practical Work n°2

We consider the numerical resolution of the following Poisson problem over the set $[0,1]$,

$$\Delta u(x) = f(x), \quad \forall x \in]0,1[, \quad u(0) = u(1) = 0.$$

Let $x \in]0,1[$, and $h > 0$, we can use the approximation

$$u'(x) \approx \frac{u(x+h) - u(x)}{h},$$

to obtain an approximation formula for the estimate of $u''(x)$, we get

$$u''(x) \approx \frac{1}{h^2}\left(u(x+2h) - 2u(x+h) + u(x)\right).$$

Finally, it is possible to write a discrete version of the Poisson problem as

$$-\frac{1}{h^2}Ax = b.$$

where

$$A := \begin{pmatrix} 2 & -1 & & 0 \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ 0 & & -1 & 2 \end{pmatrix}, \quad b := \begin{pmatrix} f(h) \\ f(2h) \\ \vdots \\ f(1-h) \end{pmatrix} \quad x := \begin{pmatrix} u(h) \\ u(2h) \\ \vdots \\ u(1-h) \end{pmatrix}$$

To simplify, we forgot about the terms in h in this equation and we consider rather the numerical resolution of the following equation

$$Ax = b.$$

where $A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n$, and $x \in \mathbb{R}^n$, with $n \in \mathbb{N}^*$. In the sequel we will set

$$b = (1, 1, \cdots, 1)^\top \in \mathbb{R}^n.$$

Exercice 1

Gradient method.

In order to solve equation $Ax = b$ we introduce the following cost function $f : \mathbb{R}^n \to \mathbb{R}$,

$$f(x) = \frac{1}{n}\|Ax - b\|^2 = \frac{1}{n}\sum_{k=1}^{n}(a_k x - b_k)^2.$$

where a_k is the kth line of matrix A, and b_k the kth element of vector b.

1. Let $n = 20$, create the matrix A and vector b using Numpy
2. Compute the gradient of f. Write a python function that compute $f(x)$ for a vector x and another function that compute $\nabla f(x)$.
3. The gradient method for solving our problem consist in computing the sequence (x_k) where $x_0 \in \mathbb{R}^n$ and

   $$x_{k+1} = x_k - \alpha \nabla f(x_k).$$

   Implement the gradient method with $x_0 = 0_n$, and with the following stopping condition: stop at iteration k if

   $$f(x_k) < \varepsilon$$

   Try different value of $\alpha > 0$ in order to get a fast convergence of the algorithm, use $\varepsilon = 0.1$ for the stop condition.
4. Draw a plot of the curves $(k, f(x_k)$ and $(k, \|\nabla f(x_k)\|)$ for all iterations k.
5. Evaluate the number of iterations needed to reach the stopping condition with $\varepsilon = 0.5$ and $x_0 = 0_n$. Then use the package `time` of Python in order to evaluate the mean duration of an iteration of the gradient algorithm.

Exercice 2

Stochastic Gradient method.

The expression of the cost function f, that is

$$f(x) = \frac{1}{n} \sum_{k=1}^{n} (a_k x - b_k)^2.$$

can be seen, from a statistical point of view, as a computation of a mean. The principle of the stochastic gradient method is to replace the computation of $\nabla f(x)$ by a random vector build from this statistical point of view. The computation of the gradient can indeed be very costly (for large value of n), since we have to compute a product of matrices. The stochastic gradient doesn't need to perform such computations. At iteration k, the gradient is replaced by a vector g_k which is the gradient of function

$$x \mapsto (a_i x - b_i)^2,$$

where i is a random integer between 1 and n. The algorithm is thus the following

1. Choose $x_0 \in \mathbb{R}^n$, for instance $x_0 = 0_n$, and set the value of $\alpha > 0$.
2. at iteration k:

   (a) draw a random integer i between 1 and n.

   (b) compute g_k, the gradient of $x \mapsto (a_i x - b_i)^2$.

   (c) compute the next iterate

   $$x_{k+1} = x_k - \alpha g_k$$

3. stop the algorithm if $f(x_k) < \varepsilon$

1. implement the stochastic gradient algorithm with $\varepsilon = 0.1$. Try different value for α in order to get a converging algorithm.
2. Draw the plot of the curve $(k, f(x_k)$ for all iterations k and compare with the curves obtained with the gradient method.
3. Evaluate the number of iterations needed to reach the stopping condition with $\varepsilon = 0.5$ and $x_0 = 0_n$. Then use the package `time` of Python in order to evaluate the mean duration of an iteration of the stochastic gradient algorithm. Compare with the gradient method.