



Unconstrained Optimization

Projet sur les réseaux de distribution d'eau

Professor: Kengy Barty

Student: 1. Mohamed Ali
2. Truong Hai Le

Paris, December 2021

REPORT FOR THE PROJECT

1. Introduction

This report aims at summarizing what we have done for the water network problem. The main task is to determine the optimal flow pressure in different arcs. To deal with this task, we first compute the result theoretically. To achieve the result, we will build up a minimization problem to find the balance of the flow pressure in section 2. As any other optimization problem, we can hardly find the solution by solving equations. In our case, we have used iteration methods to reach an acceptable solution. Those methods include (i) gradient descent method, (ii) Newton method, (iii) gradient conjugate method and (iv) BFGS method. With each method, we will apply fixed step size and variable step size for the line search. We will not dig deeply in the mathematical meaning for each method but in section 3, we will represent the result of each method in different cases of step size and explain which is the best for this method before citing out which is the best method amongst all. Conclusion comes in section 4.

2. Mathematical Expression

In the first I would like to define some notations:

r : reservoir

d : demand

A : characteristic matrix

f : flow at the nodes (f_d : flow at reservoir nodes, f_d : flow at demand nodes)

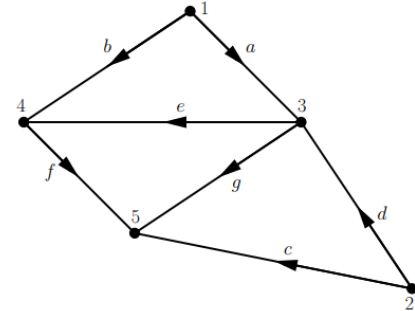
p : pressure at the nodes (p_r : pressure at reservoir nodes, p_d : pressure at demand nodes)

q : flow in the arcs

m : number of nodes (m_r : nodes related to reservoirs, m_d : nodes related to consumption)

r : resistance of the arcs

n : number of arcs



A system must satisfy the rule: the total flow at one node must equal the consumption at that node. We have the following equation system:

$$Aq = f$$

(where: $q \in R^n$, $A \in R^m \times R^n$, $f \in R^m$)

According to the above system, the number of known parameters are $m_d + m_r$, where the number of unknown parameters are $m_r + m_d + n$.

To find the network equilibrium, we have to solve the following problem:

$$\min_{(q \in R^n, f_r \in R^{m_r})} \frac{1}{3} \langle q, r \cdot q \cdot |q| \rangle + \langle p_r, f_r \rangle$$

with the constraint: $Aq = f$

If we expand the matrix A and substitute to the objective function to make it unconstrained optimization, finally we have:

$$\min_{(q_c \in R^{n-m_d})} \frac{1}{3} \langle q_0 + Bq_c, r \cdot q_0 + Bq_c \cdot |q_0 + Bq_c| \rangle + \langle p_r, A_r(q_0 + Bq_c) \rangle$$

the value of the function is

$$\psi(q_c) = \frac{1}{3} \langle q_0 + Bq_c, r \cdot q_0 + Bq_c \cdot |q_0 + Bq_c| \rangle + \langle p_r, A_r(q_0 + Bq_c) \rangle$$

the gradient of the function is

$$\nabla \psi = B^T r \cdot q_0 + Bq_c \cdot |q_0 + Bq_c| + B^T A_r^T p_r$$

the hessian of the function is

$$\nabla^2 \psi = B^T \text{diag}(2r \cdot |q_0 + Bq_c|) B$$

General descent method

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}$$

The outline of a general descent method is as follows, it alternates between two steps: determine a descent direction Δx and the selection of the step size t .

The Algorithm:

Given: a starting point $x \in \text{dom } f$

Repeat: 1. Determine a descent direction Δx .
2. Line search. Choose a step size $t > 0$.
3. update $x := x + t\Delta x$

Until: stopping criterion is satisfied.

The stopping criterion is often of the form $\|\nabla f(x)\| \leq \eta$, where η is the threshold (small and positive).

In order to do the algorithm, we will focus on two criteria: direction of the step and step size.

Direction of the step or search direction Δx

We will compute the direction of the step using the four methods:

- a) **Gradient Descent method** ($\Delta x = -\nabla f(x)$)
- b) **Newton method** ($\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x)$)
- c) **Conjugate Gradient method**

$$\Delta x^{(k)} = \begin{cases} -\nabla f(x)^{(1)} & , k = 1 \\ -\nabla f(x)^{(k)} + \beta^{(k)} \Delta x^{(k-1)} & , otherwise \end{cases}$$

Polak-Ribière coefficient $\beta^{(k)}$

$$\beta^{(k)} = \frac{\nabla f(x)^{(k)T} (\nabla f(x)^{(k)} - \nabla f(x)^{(k-1)})}{\|\nabla f(x)^{(k-1)}\|^2}$$

- d) **Quasi-Newton Method(BFGS).**

$$\Delta x^{(k)} = -\omega^{(k)} \nabla f(x)^{(k)}$$

ω^1 is identity matrix (I)

$$\delta_x^{(k)} = x^{(k)} - x^{(k-1)}$$

$$\delta_g^{(k)} = \nabla f(x)^{(k)} - \nabla f(x)^{(k-1)}$$

$$\omega^{(k)} = \left(I - \frac{\delta_x^{(k)} \delta_g^{(k)T}}{\delta_g^{(k)T} \delta_x^{(k)}} \right) \omega^{(k-1)} \left(I - \frac{\delta_g^{(k)} \delta_x^{(k)T}}{\delta_g^{(k)T} \delta_x^{(k)}} \right) + \frac{\delta_x^{(k)} \delta_x^{(k)T}}{\delta_g^{(k)T} \delta_x^{(k)}}$$

Step size or step length

We will compute the optimal value for our problem “function” by each method to compute the direction of descent at 4 cases of step size:

- a) fixed step size (t)
- b) variable step size which satisfy the Wolfe conditions,
first condition

$$f(x^{(k)} + \alpha^{(k)} d^{(k)}) \leq f(x^{(k)}) + \omega_1 \alpha^{(k)} g^{(k)T} d^{(k)}$$

Second condition

$$\left(\nabla f(x^{(k)} + \alpha^{(k)} d^{(k)}) \right)^T d^{(k)} \geq \omega_2 g^{(k)T} d^{(k)}$$

where $0 < \omega_1 < \omega_2 < 1$

initial put $\underline{\alpha} = 0$, $\bar{\alpha} = +\infty$; $\alpha^{(k,l)} \in]\underline{\alpha}, \bar{\alpha}[$

- if $\alpha^{(k,l)}$ not verify first condition

$$\bar{\alpha} = \alpha^{(k,l)}$$

$$\alpha^{(k,l+1)} = \frac{1}{2} (\underline{\alpha} + \bar{\alpha})$$

- if $\alpha^{(k,l)}$ not verify second condition

$$\underline{\alpha} = \alpha^{(k,l)}$$

If $\bar{\alpha} = +\infty$:

$$\alpha^{(k,l+1)} = 2 \underline{\alpha}$$

else:

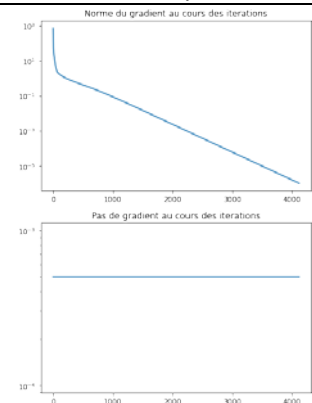
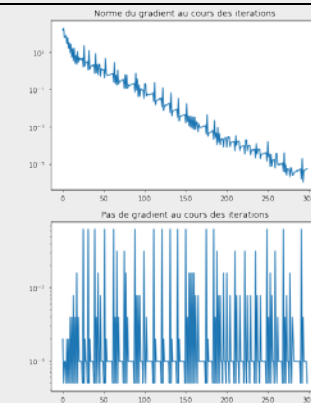
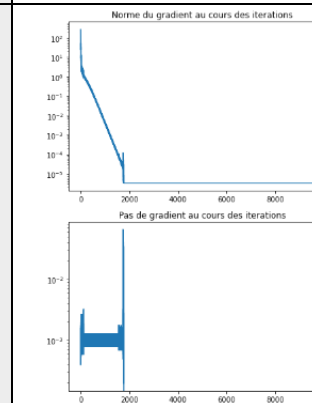
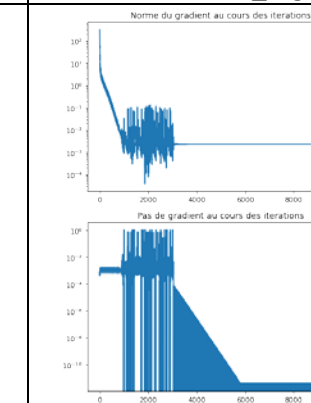
$$\alpha^{(k,l+1)} = \frac{1}{2}(\underline{\alpha} + \bar{\alpha})$$

- c) variable step size which computed using two implemented function in python from Scipy called "fmin", "fmin_bfgs" to compute the step length make minimum value of the function.

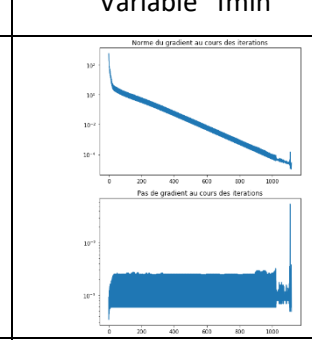
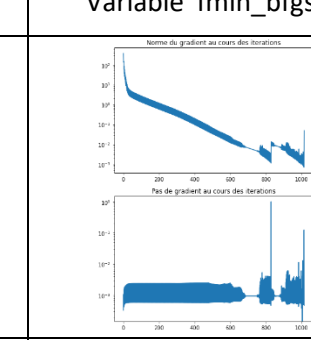
$$\varepsilon^{(k)} = \arg \min_{\varepsilon > 0} f(x^{(k)} - \varepsilon \nabla f(x^{(k)}))$$

3. Python results

i) Gradient Descent Method

case	Fixed step size	Variable Wolfe	Variable “fmin”	Variable “fmin_bfgs”
Graph				
Threshold	0.000001	0.000001	0.000001	0.000001
Initial step	0.0005	1	1	1
No. Iteration	4122	299	9999	9999
Time	1.11 s	1.1 s	42.2 s	32.2 s
Optimal Value	-3.73400704804348753	-3.734007048043514	-3.734007048043029	-3.7340069504641455
Gradient norm	$9.96640748269163 \times 10^{-7}$	$8.361641381121785 \times 10^{-7}$	$3.295716995848965 \times 10^{-6}$	0.002300888840238408

From the above results we find at the threshold 0.000001, fmin and fmin_bfgs fail to force the norm of the gradient goes below the threshold after 10000 iterations and they cannot converge even when we increase the number of iterations. However, if we increase the threshold to 0.00001, these methods converge. Finally, we find the best way to compute optimal value using gradient descent method with variable step size using wolfe conditions because it get best optimal value with the least iterations (299) and consuming time (1.1 s).

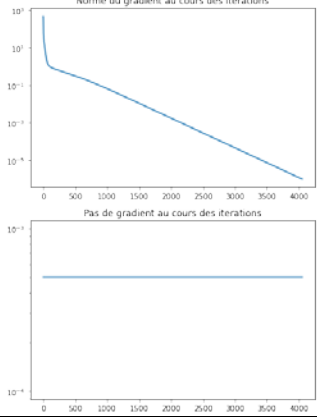
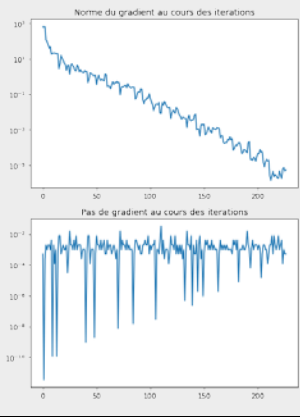
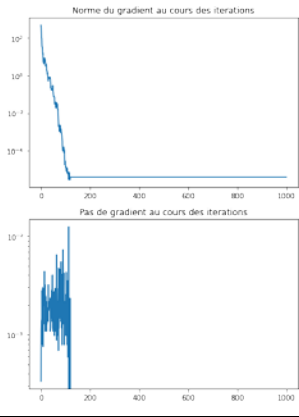
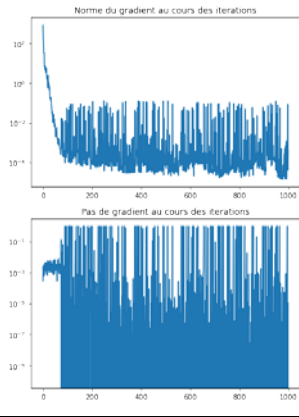
case	Variable “fmin”	Variable “fmin_bfgs”
Graph		
Threshold	0.00001	0.0001
No. Iteration	1119	1015
Time	5.24 s	4.62 s
Optimal Value	-3.734007048037581	-3.734007047795144
Gradient norm	$9.3796757440198 \times 10^{-6}$	$6.5814299593796 \times 10^{-5}$

ii) Newton Method

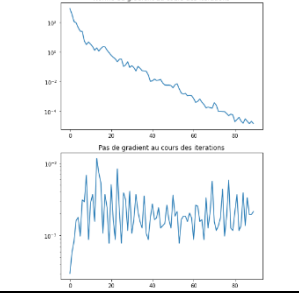
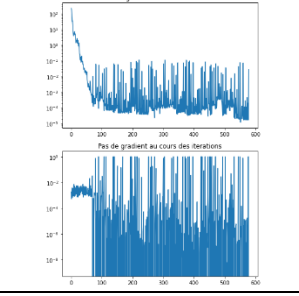
case	Fixed step size	Variable Wolfe	Variable “fmin”	Variable “fmin_bfgs”
Graph				
Threshold	0.000001	0.000001	0.000001	0.000001
Initial step	1	1	1	1
No. Iteration	6	5	5	5
Time	1.31 s	7.31 s	1.42 s	1.46 s
Optimal Value	-3.734007048043564	-3.734007048043555	-3.7340070480435656	-3.7340070480435497
Gradient norm	$8.1533000875842 \times 10^{-10}$	$1.823886636318556 \times 10^{-9}$	$3.376295514822295 \times 10^{-7}$	$9.932958451189666 \times 10^{-7}$

The table suggests that we cannot say for sure which is the best method among four candidates. Fixed step size will be the fastest method (1.31 s) where it needs more iteration and can't to be the best for every function than the other methods. In order to be concrete, we decided that the best optimal solution can get by applying Newton method with variable step size computed using implemented optimization function in python is “fmin” because it not take too much time.

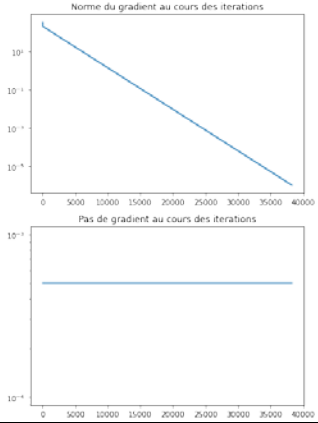
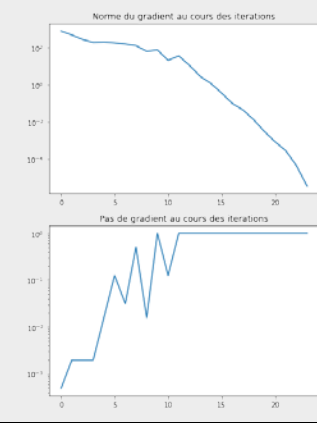
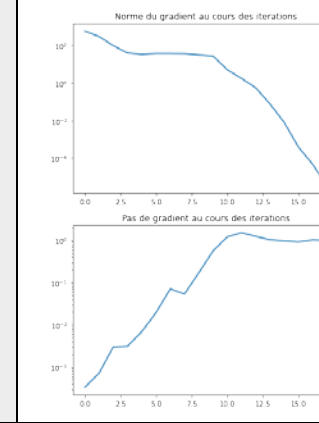
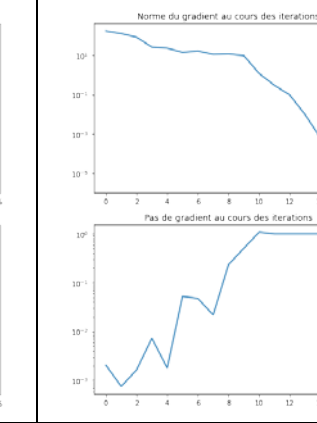
iii) Conjugate Gradient method

case	Fixed step size	Variable Wolfe	Variable “fmin”	Variable“fmin_bfgs”
Graph				
Threshold	0.000001	0.000001	0.000001	0.000001
Initial step	0.0005	1	1	1
No. Iteration	4053	227	999	999
Time	1.17 s	1.14 s	4.33 s	4.25 s
Optimal Value	-3.7340070480434964	-3.734007048043555	-3.734007048043459	-3.734007047989195
Gradient norm	$9.98823490332182 \times 10^{-7}$	$9.142682497963075 \times 10^{-7}$	$3.912653206027933 \times 10^{-6}$	0.0004094469383642634

From the above results we find at the threshold 0.000001, fmin and fmin_bfgs fail to force the norm of the gradient goes below the threshold after 1000 iterations and they cannot converge even when we increase the number of iterations. However, if we increase the threshold to 0.00001, these methods converge. Finally, we find the best way to compute optimal value using Conjugate Gradient method with variable step size using wolfe conditions because it gets best optimal value with the least iterations (227) and consuming time (1.14 s).

case	Variable “fmin”	Variable“fmin_bfgs”
Graph		
Threshold	0.00001	0.00001
No. Iteration	90	579
Time	1.16 s	3.4 s
Optimal Value	-3.734007048043434	-3.7340070480422476
Gradient norm	$9.09940715417606 \times 10^{-6}$	$9.92508124676572 \times 10^{-6}$

iv) Quasi-Newton Method(BFGS)

case	Fixed step size	Variable Wolfe	Variable “fmin”	Variable“fmin_bfgs”
Graph				
Threshold	0.000001	0.000001	0.000001	0.000001
Initial step	0.0005	1	1	1
No. Iteration	38278	24	18	17
Time	5.22 s	879 ms	1.07 s	1.19 s
Optimal Value	-3.734007048043562	-3.734007048043568	-3.734007048043029	-3.734007048043562
Gradient norm	$9.99605087423072 \times 10^{-7}$	$4.968121168884151 \times 10^{-7}$	$3.603363259354432 \times 10^{-7}$	$8.515806868641927 \times 10^{-8}$

As shown in the above table, fmin and fmin_bfgs seem to dominate the other two candidates in the number of iteration needed. Fmin_bfgs needs only 17 iterations to achieve the optimal value while it takes a little bit more time than fmin, 1.19 s in comparing with 1.07s. For this method, we prefer fmin_bfgs to fmin, but we think the best method will be with variable step size using wolfe conditions because it's have the best consuming time unless it's take more iterations than fmin and fmin_bfgs.

4. Conclusion

From the above results, we can say that we can find the solution for the optimal flow pressure problem. With the same objective function, four different methods with four different types of step sizes gave us a range of results. While some of them cannot give a solution, namely gradient descent method and gradient conjugate descent method both with fmin and fmin_bfgs step size, some other methods proved to be very efficient in dealing with the problem. In the global perspective, Newton and Quasi-Newton method are faster and calculating-inexpensive than the other method. They are also better because they can give a convergence with any given step size. In the local perspective, we choose Newton method with fmin as our best method for its super iteration saving while it does not need too much more time in calculating.