

1 Présentation du problème

Un réseau de distribution d'eau potable se présente sous la forme d'un **graphe orienté** \mathcal{G} dont l'ensemble des arcs, de cardinal n , est noté \mathcal{A} et l'ensemble des nœuds, de cardinal m , est noté \mathcal{N} :

$$\mathcal{G} = (\mathcal{A}, \mathcal{N}) \quad , \quad \text{card}(\mathcal{A}) = n \quad , \quad \text{card}(\mathcal{N}) = m . \quad (1)$$

L'ensemble \mathcal{N} se partitionne en un sous-ensemble \mathcal{N}_r , de cardinal m_r , correspondant aux nœuds où sont localisés les réservoirs du réseau, et un sous-ensemble \mathcal{N}_d , de cardinal m_d , correspondant aux nœuds où sont localisées les demandes (consommations) du réseau :

$$\mathcal{N} = \mathcal{N}_r \cup \mathcal{N}_d \quad , \quad \text{card}(\mathcal{N}_r) = m_r \quad , \quad \text{card}(\mathcal{N}_d) = m_d . \quad (2)$$

On supposera que les arcs (resp. nœuds) du graphe sont numérotés de 1 à n (resp. m), et que les nœuds où sont localisés les réservoirs sont les premiers dans la numérotation de \mathcal{N} . On supposera de plus que le graphe est **connexe**, ce qui signifie qu'il existe au moins un chemin (non orienté) reliant deux nœuds quelconques du graphe. Cette condition de connexité implique la relation :

$$n \geq m - 1 . \quad (3)$$

La topologie du réseau peut être décrite par sa **matrice d'incidence nœuds-arcs**, que l'on note A . Cette matrice a autant de lignes (resp. colonnes) qu'il y a de nœuds (resp. arcs) dans le graphe associé. Étant donné un arc α et un nœud i du graphe, l'élément $a_{i,\alpha}$ de la matrice A a pour valeur :¹

$$a_{i,\alpha} = \begin{cases} -1 & \text{si } i \text{ est le nœud initial de l'arc } \alpha, \\ +1 & \text{si } i \text{ est le nœud final de l'arc } \alpha, \\ 0 & \text{sinon.} \end{cases} \quad (4)$$

Remarque 1. Chaque colonne de la matrice A contient **exactement** un -1 et un $+1$. La somme des lignes de la matrice est donc identiquement nulle, ce qui prouve (au moins dans le cas $n \geq m$) que la matrice A n'est pas de plein rang. On peut montrer que la matrice obtenue en supprimant une ligne² quelconque de la matrice A est de plein rang. On déduit alors de la relation (3) que :

$$\text{rang}(A) = m - 1 . \quad (5)$$

Remarque 2. La i -ème ligne de la matrice A décrit la connectivité du nœud i dans le graphe : le nombre d'éléments non nuls de cette ligne est égal au nombre d'arcs reliés au nœud i , et le signe d'un élément indique si l'arc correspondant est entrant ou sortant en ce nœud.

On note alors :

- f **le vecteur des flux aux nœuds du graphe** ; les valeurs du flux aux nœuds de \mathcal{N}_d sont supposées connues : elles sont égales aux demandes (exprimées en mètre-cube par seconde) des consommateurs, et sont comptées positivement dans le cas du soutirage et négativement dans le cas de l'injection ; les valeurs du flux aux nœuds de \mathcal{N}_r doivent être calculées, et correspondent aux débits entrant ou sortant des réservoirs ;
- p **le vecteur des pressions aux nœuds du graphe** ; les valeurs de la pression aux nœuds de \mathcal{N}_r sont supposées connues : elles sont égales aux hauteurs des colonnes d'eau contenues dans les réservoirs (exprimées en mètre) ; les valeurs des pressions aux nœuds de \mathcal{N}_d doivent être calculées ;
- r **le vecteur des résistances des arcs du graphe** ; les valeurs des résistances sont supposées connues ; elles dépendent du diamètre, de la longueur et de la rugosité des canalisations ;
- q **le vecteur des débits dans les arcs du graphe** ; ces valeurs doivent toutes être calculées.

1. Le graphe étant *orienté*, il n'y a pas d'ambiguïté sur les notions de nœud initial et de nœud final d'un arc.
2. une seule ligne car on a supposé que le graphe \mathcal{G} est connexe

L'état d'équilibre du réseau se traduit par des équations de deux types.

1. La première série d'équations exprime le fait qu'il ne peut pas y avoir accumulation d'eau aux nœuds du graphe (**première loi de Kirchhoff**). Compte tenu de la remarque 2, cette loi a pour expression matricielle :

$$Aq - f = 0 . \quad (6)$$

2. La seconde série d'équations combine le fait que la perte de charge le long d'un arc dérive d'un potentiel (**seconde loi de Kirchhoff**) et que cette perte de charge est elle-même une fonction du débit de l'arc (**loi d'Ohm non linéaire**) : pour un arc α de nœud initial i et de nœud final j , la perte de charge z_α , égale à la différence $p_i - p_j$ des pressions aux extrémités de l'arc, est une fonction φ_α dont la forme est, par exemple, donnée par la formule de Colebrooks :

$$\varphi_\alpha(q_\alpha) = r_\alpha q_\alpha |q_\alpha| . \quad (7)$$

Notant $r \bullet q \bullet |q|$ le vecteur ($\in \mathbb{R}^n$) de composantes $r_\alpha q_\alpha |q_\alpha|$, on obtient les relations :³

$$A^\top p + r \bullet q \bullet |q| = 0 . \quad (8)$$

La partition de l'ensemble des nœuds \mathcal{N} en \mathcal{N}_r et \mathcal{N}_d induit une partition de la matrice A et des vecteurs f et p en :

$$A = \begin{pmatrix} A_r \\ A_d \end{pmatrix} , \quad f = \begin{pmatrix} f_r \\ f_d \end{pmatrix} , \quad p = \begin{pmatrix} p_r \\ p_d \end{pmatrix} . \quad (9)$$

Le but de l'étude est de calculer les vecteurs q , f_r et p_d vérifiant les relations (6) et (8), en supposant connus la matrice A et les vecteurs r , f_d et p_r . C'est le problème de la détermination du point d'équilibre d'un réseau d'eau soumis aux deux lois de Kirchhoff et à la loi d'Ohm, ici non linéaire.

2 Mise sous forme d'un problème d'optimisation

On peut montrer⁴ que le problème d'équilibre décrit au §1 est équivalent à un problème de minimisation sous contraintes. Notant Φ_α la primitive de la fonction φ_α définie en (7) :

$$\Phi_\alpha(q_\alpha) = \frac{1}{3} r_\alpha q_\alpha^2 |q_\alpha| , \quad (10)$$

on définit la fonction \bar{J} (qui peut s'interpréter comme l'énergie du réseau) par :

$$\bar{J}(q, f_r) = \sum_{\alpha \in \mathcal{A}} \Phi_\alpha(q_\alpha) + \sum_{i \in \mathcal{N}_r} p_i f_i . \quad (11)$$

Notant $\langle \cdot, \cdot \rangle$ le produit scalaire usuel (dans \mathbb{R}^n ou \mathbb{R}^{m_r}), cette fonction se met sous la forme compacte :

$$\bar{J}(q, f_r) = \frac{1}{3} \langle q, r \bullet q \bullet |q| \rangle + \langle p_r, f_r \rangle . \quad (12)$$

La détermination de l'équilibre du réseau revient alors à minimiser l'énergie du réseau tout en respectant la première loi de Kirchhoff, ce qui revient à résoudre le problème d'optimisation suivant :

$$\min_{(q \in \mathbb{R}^n, f_r \in \mathbb{R}^{m_r})} \frac{1}{3} \langle q, r \bullet q \bullet |q| \rangle + \langle p_r, f_r \rangle , \quad (13a)$$

sous la contrainte :

$$Aq - f = 0 . \quad (13b)$$

3. L'opération " \bullet " correspondant au produit terme à terme de deux vecteurs s'appelle le **produit d'Hadamard**.

4. et on le montrera lors de la quatrième séance de TP, après le cours sur la dualité et la relaxation lagrangienne

Sous cette forme, il est (un peu) délicat de prouver l'existence et l'unicité de la solution, essentiellement à cause de la présence du terme linéaire en f_r dans le critère, qui fait que la fonction \bar{J} n'est ni coercive, ni strictement convexe. Pour surmonter cette difficulté, on utilise la partition (9) de la matrice A pour exprimer les flux f_r en fonction des débits q . Le problème (13) se met alors sous la forme équivalente :

$$\min_{(q \in \mathbb{R}^n)} \frac{1}{3} \langle q, r \bullet q \bullet |q| \rangle + \langle p_r, A_r q \rangle , \quad (14a)$$

sous la contrainte :

$$A_d q - f_d = 0 . \quad (14b)$$

Ce problème sous contrainte peut lui-même être mis sous la forme d'une minimisation libre : d'après la remarque 1, la matrice A_d est de plein rang m_d , et on peut donc en extraire une sous-matrice carrée inversible de dimension m_d . En supposant que cette sous-matrice est constituée des m_d premières colonnes de la matrice A_d , cette dernière se partitionne en :

$$A_d = (A_{d,T} \ A_{d,C}) . \quad (15)$$

Cette partition des colonnes de A_d correspond à une partition de l'ensemble des arcs du graphe suivant laquelle le vecteur des débits s'écrit :⁵

$$q = \begin{pmatrix} q_T \\ q_C \end{pmatrix} . \quad (16)$$

Alors, la contrainte du problème (14) permet d'exprimer q_T en fonction de q_C :

$$q_T = A_{d,T}^{-1} (f_d - A_{d,C} q_C) , \quad (17)$$

d'où l'on déduit l'expression du vecteur q lui-même en fonction de q_C :

$$q = q^{(0)} + B q_C \quad \text{avec} \quad q^{(0)} = \begin{pmatrix} A_{d,T}^{-1} f_d \\ 0_{n-m_d} \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} -A_{d,T}^{-1} A_{d,C} \\ I_{n-m_d} \end{pmatrix} . \quad (18)$$

Le problème (14) se réécrit finalement sous la forme sans contrainte :

$$\min_{(q_C \in \mathbb{R}^{n-m_d})} \frac{1}{3} \langle q^{(0)} + B q_C, r \bullet (q^{(0)} + B q_C) \bullet |q^{(0)} + B q_C| \rangle + \langle p_r, A_r (q^{(0)} + B q_C) \rangle . \quad (19)$$

Remarque 3. La matrice B a une interprétation intéressante en terme de graphe car chacune de ses colonnes représente un cycle du réseau, c'est-à-dire soit un chemin reliant deux réservoirs du réseau, soit un chemin dont le nœud initial et le nœud final sont identiques. L'ensemble des colonnes de B forme même une base de cycles, en ce sens que tout cycle du réseau est engendré par une combinaison des colonnes de B . La matrice B est appelée **matrice d'incidence arcs-cycles** du réseau.

5. À l'aide de la théorie des graphes, on montre que cette partition correspond à une décomposition **arbre/co-arbre** de l'ensemble des arcs du réseau, d'où les indices T ("Tree") et C ("Cotree") utilisés dans le partitionnement des matrices et des vecteurs.

3 Aspects informatiques

3.1 Langages

Ce TP sera réalisé en **PYTHON** ou en **SCILAB**, de préférence à d'autres langages. Néanmoins, les fichiers de données et la plupart des fonctions permettant de réaliser le TP sont aussi disponibles en **JULIA** et en **MATLAB**.

Pour les langages **PYTHON** et **JULIA**, un **notebook** prototype est fourni, qui permet de guider l'utilisateur dans la réalisation du TP et de simplifier l'exécution des codes à développer. On peut aussi y insérer les calculs, réponses aux questions théoriques et commentaires associés au TP. À défaut de notebook, un **moniteur d'enchaînement** est disponible pour tous les langages, qui permet lui aussi d'être guidé dans la réalisation du TP.

3.2 Variables

Pour mémoire, on donne ci dessous la table de correspondance entre les symboles mathématiques utilisés dans la description du problème et les variables informatiques utilisées dans les codes. La dernière partie du tableau (q , q_C , z , f et p) correspond aux variables qu'il s'agit de calculer lors de la résolution du problème, alors que les variables dans les premières parties du tableau correspondent à des données fournies aux utilisateurs (voir les fichiers **Probleme_R** et **Structures_N**).

Description physique de la variable	Nom math.	Variable info.	Ensemble d'appartenance
Nombre total d'arcs	n	n	\mathbb{N}
Nombre total de nœuds	m	m	\mathbb{N}
Nombre de nœuds de demande	m_d	md	\mathbb{N}
Nombre de nœuds réservoir	m_r	mr	\mathbb{N}
Demandes aux nœuds de demande	f_d	fd	$\mathcal{M}(m_d, 1)$
Pressions aux nœuds réservoir	p_r	pr	$\mathcal{M}(m_r, 1)$
Résistances des arcs	r	r	$\mathcal{M}(n, 1)$
Vecteur initial des débits	$q^{(0)}$	q0	$\mathcal{M}(n, 1)$
Matrice d'incidence nœuds-arcs	A	A	$\mathcal{M}(m, n)$
Sous-matrice "demande" de A	A_d	Ad	$\mathcal{M}(m_d, n)$
Sous-matrice "réservoir" de A	A_r	Ar	$\mathcal{M}(m_r, n)$
Sous-matrice "arbre" de A_d	$A_{d,T}$	AdT	$\mathcal{M}(m_d, m_d)$
Sous matrice "coarbre" de A_d	$A_{d,C}$	AdC	$\mathcal{M}(m_d, n - m_d)$
Matrice inverse de $A_{d,T}$		AdI	$\mathcal{M}(m_d, m_d)$
Matrice d'incidence arcs-cycles	B	B	$\mathcal{M}(n, n - m_d)$
Débits des arcs	q	q	$\mathcal{M}(n, 1)$
Débits réduits des arcs	q_C	qc	$\mathcal{M}(n - m_d, 1)$
Pertes de charge des arcs	z	z	$\mathcal{M}(n, 1)$
Flux des nœuds	f	f	$\mathcal{M}(m, 1)$
Pressions des nœuds	p	p	$\mathcal{M}(m, 1)$

TABLE 1 – Correspondance entre noms mathématiques et variables informatiques

On fera attention au fait que les noms des variables figurant dans la première partie de la table ci-dessus sont **réservés** : il ne faut donc pas les redéfinir sous peine de perdre, dans certains langages l'accès aux données correspondantes.

4 Séance de travaux pratiques No 1

1. On s'intéresse au **problème primal** d'optimisation sans contrainte (19).
 - À partir des données et des notations des fichiers **Probleme_R** et **Structures_N**, écrire un premier oracle associé au problème qui fournit en un point q_C les valeurs du critère et du gradient du problème (19) :
$$[F, G] = \text{OraclePG}(q_C),$$
avec :
 - q_C : vecteur des débits réduits,
 - F : valeur du critère évalué au point q_C ,
 - G : vecteur des dérivées du critère par rapport à q_C .
 - Écrire un second oracle incorporant le calcul du Hessien :
$$[F, G, H] = \text{OraclePH}(q_C),$$
avec :
 - q_C : vecteur des débits réduits,
 - F : valeur du critère évalué au point q_C ,
 - G : vecteur des dérivées premières du critère par rapport à q_C ,
 - H : matrice des dérivées secondes du critère par rapport à q_C .
2. Tester ces deux oracles en les interfaçant respectivement avec l'algorithme de gradient à pas fixe (fonction **Gradient_F**) et l'algorithme de Newton à pas unité (fonction **Newton_F**) tous deux fournis dans le cadre du TP. On notera dans ces algorithmes la mémorisation des valeurs du critère, de la norme du gradient et de la longueur du pas de gradient (en fait le \log_{10} de ces deux dernières quantités), permettant de visualiser le comportement de l'algorithme au cours des itérations (fonction **Visualg**).
3. Tester le premier oracle en l'interfaçant avec une fonction d'optimisation disponible dans le langage utilisé : fonction **minimize** de PYTHON (voir le fichier **Scipy_Optim.py**), ou fonction **optim** de SCILAB (voir le fichier **Scilab_Optim.sci**), ...

4. Question facultative.

Dans l'algorithme de la question 3, remplacer le pas fixe par le pas de Cauchy, à savoir le pas qui minimise le coût selon la direction de descente : si l'on minimise la fonction J à partir d'un point donné x dans une direction donnée d , le pas de Cauchy est le résultat de la minimisation suivante (en dimension 1) : $\min_{\alpha \in \mathbb{R}^+} J(x + \alpha d)$.

Conseil : pour calculer le pas de Cauchy, on utilisera la fonction d'optimisation du langage utilisé, que l'on appellera avec l'oracle (à écrire) associé au problème : $\alpha \mapsto J(x + \alpha d)$.

5. Question théorique.

Montrer que les problèmes (13), (14) et (19) sont équivalents.

Montrer que les problèmes (14) et (19) admettent chacun une solution unique, et que les critères de ces 2 problèmes sont *strictement* (mais pas *fortement*) convexes.

1. Il est fortement conseillé de respecter les *noms* des fonctions (`OraclePG`, `OraclePH`), ainsi que les *arguments* en entrée et en sortie de ces fonctions.
2. Dans le nom `OraclePG`, la lettre P indique que l'oracle correspond au problème primal, et la lettre G indique que l'oracle effectue le calcul du gradient, mais pas celui du Hessian.
3. La syntaxe d'appel de l'oracle est imposée. Les tableaux et les variables définis dans les scripts `Probleme.R` et `Structures.N` sont disponibles dans l'oracle. On notera en particulier que, la lettre *f* (minuscule) étant utilisée pour désigner le vecteur des flux aux nœuds du réseau, la valeur du critère est notée F (majuscule).
4. On encapsulera l'appel à toute méthode d'optimisation dans une fonction de la forme :

$$[F^\sharp, x^\sharp, G^\sharp] = \text{Minimise}(\text{Oracle}, x_0, \alpha_0)$$

avec :

x_0 : valeur des variables avant optimisation,
 α_0 : longueur initiale du pas de gradient,
 F^\sharp : valeur du critère après optimisation,
 x^\sharp : valeur des variables après optimisation,
 G^\sharp : valeur du gradient après optimisation,

et où `Oracle` est la fonction oracle qui sera utilisée par la méthode d'optimisation (voir à titre d'exemple les algorithmes de gradient à pas fixe `Gradient_F` et de Newton à pas unité `Newton_F`). Comme pour l'oracle, la syntaxe d'appel de la fonction `Minimize` est imposée. Ainsi, toutes les fonctions d'optimisation que l'on sera amené à écrire respecteront les mêmes conventions, et il sera donc plus facile de les tester.

5. Il est fourni un prototype de script d'enchaînement des tâches, appelé *moniteur*, de nom `TP_Reseau_Moniteur`, définissant un cadre général pour :
 - l'acquisition et la mise en forme des données du problème,
 - la définition, l'initialisation et la résolution du problème de minimisation,
 - le calcul des variables hydrauliques du réseau à partir de la solution obtenue,
 - la vérification et la visualisation des résultats.

Si le langage utilisé permet de définir un *notebook*, il est fortement conseillé d'utiliser celui fourni dans le cadre du TP. Dans le cas contraire, le moniteur permet de mettre en œuvre les méthodes d'optimisation écrites durant le TP, en y apportant les modifications nécessaires.

L'acquisition et la mise en forme du problème sont effectuées par l'intermédiaire des 2 scripts `Probleme.R` et `Structures.N`. Une fois ces scripts exécutés, les variables qui y sont définies peuvent être utilisées par toutes les procédures utilisées dans le TP.

La résolution du problème de minimisation est la partie centrale du TP (écriture des oracles, de la méthode de recherche linéaire, des différents algorithmes d'optimisation).

Le calcul des variables hydrauliques (débits q , pertes de charge z , flux f et pressions p) à partir de la solution q_C^\sharp du problème d'optimisation se fait à l'aide de la fonction de nom `HydrauliqueP` (voir `HydrauliqueP`) qui est fournie.

La vérification des résultats se fait à l'aide de la fonction de nom `Verification` (voir `Verification`) qui calcule les écarts maximaux sur les deux lois de Kirchhoff).

La visualisation du comportement de l'algorithme se fait à l'aide de la fonction de nom `Visualg` (voir `Visualg`); les résultats de l'algorithme (valeur du critère, logarithme en base 10 de la norme du gradient et longueur du pas de gradient à chaque itération) devront être mémorisés dans la fonction d'optimisation `Minimise`.

5 Séances de travaux pratiques No 2 et 3

Le but de ces séances est de résoudre le problème (19) par 4 algorithmes d'optimisation différents.

1. Programmer une recherche linéaire vérifiant les conditions de Wolfe et coder l'algorithme de gradient à pas variable utilisant cette recherche linéaire. La recherche linéaire mettra en œuvre l'algorithme de Fletcher–Lemaréchal (voir la note [Methodes.pdf](#) disponible sur le site du TP). On prendra pour base la fonction [Wolfe](#) pour coder cet algorithme.
2. Écrire les méthodes de gradient conjugué et de quasi-Newton avec recherche de Wolfe ;
 - coder l'algorithme de Polak-Ribière (gradient conjugué non linéaire),
 - coder l'algorithme de BFGS (utilisant une approximation de l'inverse du Hessien),
 - résoudre le problème (19) par ces deux méthodes en utilisant l'oracle `OraclePG`.
3. Écrire la méthode de Newton ; on devra :
 - écrire l'algorithme de Newton, en y incorporant la recherche linéaire de Wolfe,
 - résoudre le problème (19) en utilisant l'oracle `OraclePH`.
4. Comparer les résultats (valeurs, nombre d'itérations, temps CPU, vitesse de convergence) de ces quatre méthodes d'optimisation et de la méthode initiale de gradient à pas fixe.

Question facultative.

5. Modifier les programmes pour pouvoir traiter des réseaux de **très grande taille** (plusieurs milliers d'arcs) en utilisant le fait que les matrices du problème ont une structure creuse. On se limitera à la méthode de Newton, afin d'exploiter efficacement le creux du Hessien.

Pour disposer de réseaux de grande taille (en fait de taille paramétrable), on utilisera les scripts [Probleme_P](#) et [Structures_S](#), qui remplaceront donc dans le moniteur les scripts `Probleme_R` et `Structures_N`. Le paramètre T que l'on définit dans le script [Probleme_P](#) détermine la taille du réseau :

$T = 7$: 500 arcs,
 $T = 10$: 4000 arcs,
 $T = 12$: 16000 arcs.

Pour l'utilisation des matrices creuses, on se reportera à la documentation du langage utilisé pour :

- la définition des matrices creuses (matrices décrivant le réseau et le Hessien),
- la suppression des éléments d'ordre ϵ dans de telles matrices,
- la résolution des systèmes linéaires creux (pour l'algorithme de Newton).

On modifiera l'oracle `OraclePH` et l'algorithme de Newton de façon à ce que les matrices qui y sont utilisées soient toutes définies comme des matrices creuses et utilisées en tant que telles (aucun passage par des matrices « pleines »).

1. La programmation de la méthode de recherche linéaire et des algorithmes d'optimisation doit être effectuée indépendamment du problème traité. On rappelle (voir la séance précédente) que l'on encapsule chaque méthode d'optimisation dans une fonction de la forme :
 $[F^\#, x^\#, G^\#] = \text{Minimise}(\text{Oracle}, x_0, \alpha_0)$, en utilisant de préférence les noms suivants :
 - `Gradient_V` : algorithme de gradient à pas variable,
 - `Polak_Ribiere` : algorithme de Polak-Ribière,
 - `BFGS` : algorithme de quasi-Newton BFGS,
 - `Newton_V` : algorithme de Newton à pas variable.
 Chaque méthode fait appel, d'une part à une fonction oracle, et d'autre part à la fonction de recherche linéaire (voir [Wolfe](#)).
2. Il faut être attentif à la manière de mettre en œuvre et d'arrêter l'algorithme de Fletcher–Lemaréchal. Ce point est abordé dans la note [Methodes.pdf](#). Par contre, pour le choix du pas initial dans l'algorithme de Fletcher–Lemaréchal, on se contentera de procéder par essais et erreurs et on ne cherchera pas forcément à trouver le pas initial de Fletcher.
3. Dans toutes les méthodes d'optimisation, on vérifiera que la direction suivant laquelle se fait la recherche linéaire est bien une direction de descente. Dans la méthode de Newton, on contrôlera que la matrice hessienne est bien inversible.
4. Pour comparer les vitesses de convergence, on tracera pour chaque méthode la variation du logarithme base 10 de la norme du gradient en fonction de l'indice d'itérations ; on s'inspirera de ce qui est fait dans la fonction `Gradient_F` (voir [Gradient_F](#)).
5. On notera que la recherche linéaire fait appel à l'oracle uniquement pour des calculs de critère et de gradient. La syntaxe des deux oracles écrits lors de la séance précédente est telle qu'ils peuvent être utilisés indifféremment dans l'algorithme de recherche linéaire.

6 Séance de travaux pratiques No 4

Pour cette dernière séance, on revient à la formulation sous contraintes (14) du problème de l'équilibre du réseau, que l'on va alors traiter par **dualité**.

1. Écrire le Lagrangien associé au problème sous contraintes (14) et ses conditions d'optimalité ; vérifier que ces conditions correspondent aux équations (6) et (8) de l'équilibre du réseau.
2. Constater que la minimisation en q du Lagrangien se fait de manière explicite, donner l'expression de l'argmin q^\sharp en fonction du multiplicateur dual λ et calculer les expressions de la fonction duale Φ , de son gradient et de son Hessien en fonction de λ .
3. Écrire les deux oracles `OracleDG` et `OracleDH` (D comme dualité) associés à la minimisation de l'opposé de Φ (minimiser $-\Phi$ est équivalent à maximiser Φ).
4. Résoudre le problème dual à l'aide des algorithmes développés lors des séances 2 et 3. Comparer les résultats obtenus par les différentes méthodes sur ce problème, et comparer avec les résultats obtenus sur le problème primal.

Remarques concernant la programmation

1. Les oracles `OracleDG` et `OracleDH` associés à la fonction duale Φ devront avoir la même syntaxe d'appel que celle des oracles écrits lors de la résolution du problème (19).
2. Le calcul des variables hydrauliques (débits q , pertes de charge z , flux f et pressions p) est différent de celui mis en œuvre pour le problème primal puisqu'il est effectué à partir de la solution p_d^\sharp du problème dual. Il faut donc utiliser une fonction de calcul différente, de nom `HydrauliqueD` (fichier `HydrauliqueD`).

7 Ordres de grandeur des résultats

On indique ici l'ordre de grandeur du nombre d'itérations nécessaires pour la convergence des différentes méthodes d'optimisation (pour une tolérance de convergence de 10^{-6} sur la norme du gradient) pour le réseau de distribution décrit dans les fichiers `Probleme_R.sce` et `Structures_N.sce`.

Méthode	Convergence
Gradient à pas constant	~ 4000 itérations
Gradient à pas variable	~ 300 itérations
Gradient conjugué	~ 200 itérations
Quasi-Newton (BFGS)	~ 20 itérations
Newton globalisé	~ 5 itérations

TABLE 2 – Résolution du problème primal

Méthode	Convergence
Gradient à pas constant	~ 6000 itérations
Gradient à pas variable	~ 2500 itérations
Gradient conjugué	~ 300 itérations
Quasi-Newton (BFGS)	~ 50 itérations
Newton globalisé	~ 10 itérations

TABLE 3 – Résolution du problème dual

Ces nombres d'itérations sont donnés à titre indicatif et peuvent légèrement varier en fonction du point initial des algorithmes. Pour les atteindre, il faut bien sûr jouer avec les paramètres de l'optimisation, et en particulier avec la longueur du pas avec laquelle on initialise l'algorithme de Fletcher-Lemaréchal (recherche linéaire vérifiant les conditions de Wolfe).