École des Ponts ParisTech — Département IMI

Enseignement de deuxième année "Optimisation et Contrôle"

Projet sur les réseaux de distribution d'eau

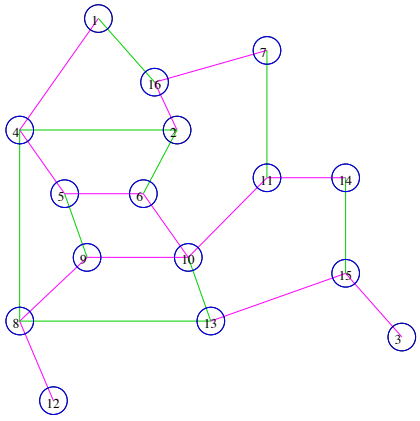
Pierre Carpentier
Unité de Mathématiques Appliquées
ENSTA ParisTech
828, boulevard des Maréchaux, 91762 Palaiseau Cedex

pierre.carpentier@ensta-paristech.fr

Année 2014-2015

Résumé

On se propose au cours de ce projet d'étudier le problème de la résolution des équations décrivant l'état d'équilibre d'un réseau de distribution d'eau potable. Ce problème peut se mettre sous la forme de la minimisation d'une fonction (représentant en quelque sorte l'énergie du réseau) sous des contraintes linéaires d'égalité (traduisant la première loi de Kirchhoff). Une fois le problème bien posé, on lui appliquera les principaux algorithmes d'optimisation du cours, tant dans le cadre sans contraintes (on explicitera alors une partie des variables à l'aide des contraintes) que dans le cadre de la dualité lagrangienne.



La version PDF de ce document ainsi que les procédures Scilab fournies dans le cadre du TP sont disponibles à l'adresse suivante : http://www.ensta.fr/~pcarpent/TP_Reseau/ENPC/

Table des Matières

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 $\mathcal{N}$:

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

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 :

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

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 :

$$n \geq m - 1 \; . \tag{3}$$

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]

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

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 ligne[2] quelconque de la matrice A est de plein rang. On déduit alors de la relation (3) que :

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

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 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 :
$$Aq - f = 0 \,. \tag{6}$$
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 :
$$\varphi_\alpha(q_\alpha) = r_\alpha q_\alpha |q_\alpha| \,. \tag{7}$$
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]
$$A^\top p + r \bullet q \bullet |q| = 0 \,. \tag{8}$$

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 :
$$A = \begin{pmatrix} A_r \\ A_d \end{pmatrix} \;,\; f = \begin{pmatrix} f_r \\ f_d \end{pmatrix} \;,\; p = \begin{pmatrix} p_r \\ p_d \end{pmatrix} \,. \tag{9}$$

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 montrer[4] 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) :
$$\Phi_\alpha(q_\alpha) = \frac{1}{3} r_\alpha q_\alpha^2 |q_\alpha| \,, \tag{10}$$
on définit la fonction $\bar{J}$ (qui peut s'interpréter comme l'*énergie* du réseau) par :
$$\bar{J}(q, f_r) = \sum_{\alpha \in \mathcal{A}} \Phi_\alpha(q_\alpha) + \sum_{i \in \mathcal{N}_r} p_i f_i \,. \tag{11}$$
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 :
$$\bar{J}(q, f_r) = \frac{1}{3} \langle q, r \bullet q \bullet |q| \rangle + \langle p_r, f_r \rangle \,. \tag{12}$$

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 :

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

sous la contrainte :

$$Aq - f = 0 \,. \tag{13b}$$

[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 :

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

sous la contrainte :

$$A_d q - f_d = 0 \ . \tag{14b}$$

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 :

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

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]

$$q = \begin{pmatrix} q_T \\ q_C \end{pmatrix} \ . \tag{16}$$

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

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

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

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

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

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

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.

3 Outils informatiques

Ce TP sera réalisé en ScicosLab (de préférence à Matlab), sur station de type Unix-Linux ou Windows. Scilab est un logiciel de calcul très semblable à Matlab, mais disponible gratuitement.

[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.

Pour mémoire, on donne ci dessous la table de correspondance entre les symboles mathématiques utilisés dans la description du problème et les variables informatiques définies dans SCILAB. 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 précédentes correspondent à des données fournies aux utilisateurs (voir les scripts SCILAB dans les fichiers `Probleme_R.sce` et `Structures_R.sce`).

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`	$\mathcal{M}(m, 1)$

Tableau 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 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 scripts `Probleme_R.sce` et `Structures_R.sce`, écrire en SCILAB un premier oracle associé à ce problème ; cet oracle fournit les valeurs du critère et du gradient du problème (19), et est codé dans une fonction de la forme :

   $[F, G, ind] = \texttt{OraclePG}\,(q_C, ind)$, 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,
   ind : indicateur du type de calcul à effectuer :
   $= 2$: calcul de F uniquement,
   $= 3$: calcul de G uniquement,
   $= 4$: calcul de F et G.
2. Tester ce premier oracle en l'interfaçant avec un algorithme de gradient à pas fixe (fichier `Gradient_F.sci`, fourni dans le cadre du TP). On pourra aussi tester l'oracle en l'interfaçant avec la fonction `optim` de Scilab (fichier `Optim_Scilab.sci`).
3. Écrire un *nouvel* oracle incorporant le calcul du hessien, i.e. une fonction de la forme :

   $[F, G, H, ind] = \texttt{OraclePH}\,(q_C, ind)$, 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,
   H : matrice des dérivées secondes du critère par rapport à q_C,
   ind : indicateur du type de calcul à effectuer :
   $= 2$: calcul de F uniquement,
   $= 3$: calcul de G uniquement,
   $= 4$: calcul de F et G,
   $= 5$: calcul de H uniquement,
   $= 6$: calcul de G et H,
   $= 7$: calcul de F, G et H.

4. **Questions théoriques à faire hors séance de TP.**
   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.

Remarques concernant la programmation

1. 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 hessien.

2. La syntaxe d'appel de l'oracle est imposée. Les tableaux et les variables définis dans les scripts `Probleme_R.sce` et `Structures_R.sce` 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).

3. On encapsulera l'appel à toute méthode d'optimisation dans une fonction de la forme :
   $\left[F^\sharp, x^\sharp, G^\sharp\right] = \texttt{Minimise}\,(\texttt{Oracle}, x_0)$, avec :

   x_0 : valeur des variables avant optimisation,

   $F^\sharp$: 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. Comme pour l'oracle, la syntaxe d'appel est imposée. Par la suite, toutes les fonctions d'optimisation que l'on sera amené à écrire respecteront les mêmes conventions, et il sera donc facile de les insérer dans un script d'enchaînement (voir remarque suivante).

4. Il est fourni un prototype de script d'enchaînement des tâches, de nom `Moniteur_Skel.sce`, 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.

   L'acquisition et la mise en forme du problème sont effectuées par l'intermédiaire des 2 scripts `Probleme_R.sce` et `Structures_R.sce`. 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 constitue le cœur 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 Scilab de nom `HydrauliqueP` (fichier `HydrauliqueP.sci`) qui est fournie.

   La vérification des résultats se fait à l'aide de la fonction Scilab de nom `Verification` (fichier `Verification.sci`) 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 Scilab de nom `Visualg` (fichier `Visualg.sci`) ; 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 Flechter–Lemaréchal (voir la note `Methodes.pdf` disponible sur le site du TP). On prendra pour base le fichier `Wolfe_Skel.sci` pour coder cet algorithme.
2. Écrire les méthodes de gradient conjugué et de quasi-Newton avec recherche de Wolfe ; plus précisément, on devra :
   - 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` (oracle avec calcul du critère et du gradient, développé lors de la séance précédente).
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` (voir séance précédente).
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.

Remarques concernant la programmation

1. La programmation de la méthode de recherche linéaire et des algorithmes d'optimisation doit être effectuée indépendamment du problème traité. On rappelle (voir séance précédente) que l'on encapsule chaque méthode d'optimisation dans une fonction de la forme :

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

   Chaque méthode fait appel, d'une part à une fonction oracle, et d'autre part à la fonction de recherche linéaire, que l'on pourra appelée `Wolfe` et dont on définira soigneusement les entrées/sorties (voir le fichier `Wolfe_Skel.sci`).

2. Il faut être attentif à la manière de mettre en œuvre et d'arrêter l'algorithme de Flechter–Lemaréchal. Ces points sont abordés dans la note `Methodes.pdf`.

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 représentera graphiquement 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` (fichier `Gradient_F.sci`).

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 car on n'utilise alors que les valeurs 2, 3 et 4 de l'argument d'entrée *ind* de l'oracle.

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.

Question facultative.

5. Modifier les programmes afin de pouvoir traiter des réseaux de **très grande taille** (plusieurs milliers d'arcs) en utilisant la structure creuse des matrices du problème.

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.sci).

3. Pour l'utilisation des structures creuses, on se reportera à la documentation Scilab pour :

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

   La description d'un réseau de grande taille (en fait un réseau de taille paramétrable) est fournie par les scripts `Probleme_S.sce` et `Structures_S.sce` (analogues des scripts `Probleme_R.sce` et `Structures_R.sce` dans le cas "creux").

A Moniteur d'enchaînement des tâches : Moniteur.sce

```
////////////////////////////////////////////////////////////////////////////
//                                                                        //
//  MONITEUR D'ENCHAINEMENT POUR LE CALCUL DE L'EQUILIBRE D'UN RESEAU D'EAU  //
//                                                                        //
////////////////////////////////////////////////////////////////////////////

// ------------------------------------
// Dimensionnement de l'espace de travail
// ------------------------------------

   stacksize(10000000);

// ----------------------------------------
// Fonctions fournies dans le cadre du projet
// ----------------------------------------

   // Donnees du problemes

   exec('Probleme_R.sce');
   exec('Structures_R.sce');

   // Affichage des resultats

   exec('Visualg.sci');

   // Verification  des resultats

   exec('HydrauliqueP.sci');
   exec('HydrauliqueD.sci');
   exec('Verification.sci');

// ----------------------------------------
// Fonctions a ecrire dans le cadre du projet
// ----------------------------------------

   // ---> Charger les fonctions  associees a l'oracle du probleme,
   //      aux algorithmes d'optimisation et de recherche lineaire.
   //
   // Exemple : la fonction "optim" de Scilab
   //
   //exec('OraclePG.sci');
   //exec('Optim_Scilab.sci');
   //titrgr = "Fonction optim de Scilab sur le probleme primal";

   // -----> A completer...
   // -----> A completer...
   // -----> A completer...

// ------------------------------
// Initialisation de l'algorithme
// ------------------------------

   // La dimension (n-md) est celle du probleme primal

   xini = 0.1 * rand(n-md,1);

// ---------------------------
// Minimisation proprement dite
// ---------------------------
```

```
   // Exemple : la fonction "optim" de Scilab
   //
   //[fopt,xopt,gopt] = Optim_Scilab(OraclePG,xini);

   // -----> A completer...

// --------------------------
// Verification des resultats
// --------------------------

   [q,z,f,p] = HydrauliqueP(xopt);

   Verification(q,z,f,p);

//
```

B Algorithme du gradient : `Gradient_F.sce`

```
function [fopt,xopt,gopt]=Gradient_F(Oracle,xini)


////////////////////////////////////////////////////////////////////////////
//                                                                        //
//         RESOLUTION D'UN PROBLEME D'OPTIMISATION SANS CONTRAINTES       //
//                                                                        //
//         Methode de gradient a pas fixe                                 //
//                                                                        //
////////////////////////////////////////////////////////////////////////////


// ------------------------
// Parametres de la methode
// ------------------------

   titre = "Parametres du gradient a pas fixe";
   labels = ["Nombre maximal d''iterations";...
             "Valeur du pas de gradient";...
             "Seuil de convergence sur ||G||"];
   typ = list("vec",1,"vec",1,"vec",1);
   default = ["5000";"0.0005";"0.000001"];
   [ok,iter,alphai,tol] = getvalue(titre,labels,typ,default);

// ----------------------------
// Initialisation des variables
// ----------------------------

   logG = [];
   logP = [];
   Cout = [];

   timer();

// -------------------------
// Boucle sur les iterations
// -------------------------

   x = xini;

   kstar = iter;
   for k = 1:iter

//    - valeur du critere et du gradient

      ind = 4;
      [F,G] = Oracle(x,ind);

//    - test de convergence

      if norm(G) <= tol then
         kstar = k;
```

```
         break
      end

//    - calcul de la direction de descente

      D = -G;

//    - calcul de la longueur du pas de gradient

      alpha = alphai;

//    - mise a jour des variables

      x = x + (alpha*D);

//    - evolution du gradient, du pas et du critere

      logG = [ logG ; log10(norm(G)) ];
      logP = [ logP ; log10(alpha) ];
      Cout = [ Cout ; F ];

   end

// ----------------------------
// Resultats de l'optimisation
// ----------------------------

   fopt = F;
   xopt = x;
   gopt = G;

   tcpu = timer();

   cvge = ['Iteration         : ' string(kstar);...
           'Temps CPU         : ' string(tcpu);...
           'Critere optimal   : ' string(fopt);...
           'Norme du gradient : ' string(norm(gopt))];
   disp('Fin de la methode de gradient a pas fixe')
   disp(cvge)

// - visualisation de la convergence

   Visualg(logG,logP,Cout);

endfunction
```

C Données du problème : Probleme_R.sce

```
////////////////////////////////////////////////////////////////////////////
//                                                                        //
//        DONNEES ASSOCIEES A LA RESOLUTION DES EQUATIONS D'UN RESEAU     //
//                                                                        //
//        Probleme_R : reseau representant un cas relativement realiste   //
//                                                                        //
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
// On donne les dimensions du reseau (nombres d'arcs, de noeuds, de reservoirs
// le constituant). On donne aussi deux vecteurs, contenant respectivement les
// numeros des noeuds initiaux et finaux des arcs du reseau. On suppose que le
// reseau comporte au moins un reservoir, que les reservoirs sont associes aux
// premiers  noeuds du graphe, et que la numerotation des noeuds du graphe est
// contigue (de 1 a m). On donne pour finir le vecteur de resistances des arcs
// du reseau, le vecteur des pressions des reservoirs ainsi que le vecteur des
// demandes aux noeuds autres que ceux correspondant aux reservoirs.
//
// On peut des a present noter que la numerotation implicite des arcs que l'on
// obtient ne doit pas etre  quelconque : on suppose en effet que les premiers
// arcs dans cette numerotation forment un arbre, ce qui fournit facilement le
// plus grand bloc carre inversible de la matrice d'incidence noeuds-arcs.
//
// On donne (de maniere facultative) les coordonnees des noeuds du reseau afin
// de pouvoir representer graphiquement le reseau. En Scilab, cette etape sera
// effectuee a l'aide de Metanet.
//
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
// Variables du probleme
// ---------------------
//
// nom  : nom du reseau
//
// n    : nombre total d'arcs
// m    : nombre total de noeuds
// mr   : nombre de noeuds de type reservoir
// md   : nombre de noeuds de type demande (= m-mr)
//
// orig : vecteur des numeros des noeuds initiaux des arcs : M(1,n)
// dest : vecteur des numeros des noeuds finaux   des arcs : M(1,n)
// absn : vecteur des abscisses des noeuds                 : M(1,m)
// ordn : vecteur des ordonnees des noeuds                 : M(1,m)
//
// r    : vecteur des resistances des arcs                 : M(n,1)
// pr   : vecteur des pressions des noeuds reservoirs      : M(mr,1)
// fd   : vecteur des flux des noeuds de demande           : M(md,1)
//
////////////////////////////////////////////////////////////////////////////
```

```
// -------------
// Nom du reseau
// -------------

   nom = 'Realiste';

// --------------------
// Dimensions du reseau
// --------------------

   n  = 22;
   m  = 16;
   mr =  3;

   md = m - mr;

// ---------------------------------------------
// Numeros des noeuds initiaux et finaux des arcs
// ---------------------------------------------

   orig = [ 1  2  3  4  5  6  7  8  8  9 10 11 13  1  2  4  5  7  8 14  2 10];
   dest = [ 4 16 15  5  6 10 16  9 12 10 11 14 15 16  6  8  9 11 13 15  4 13];

// ----------------------
// Coordonnees des noeuds
// ----------------------

   absn = [11 18 38  4  8 15 26  4 10 19 26  7 21 33 33 16];
   ordn = [28 21  8 21 17 17 26  9 13 13 18  4  9 18 12 24];

// --------------------
// Resistances des arcs
// --------------------

   r = [ 100   10 1000  100  100   10 1000  100 1000  100...
        1000 1000 1000   10   10  100  100 1000  100 1000...
         100   10]';

// --------------------------------------
// Pressions au pied des reservoirs (en m)
// --------------------------------------

   pr = [105 104 110]';

// ----------------------------------
// Flux aux noeuds de demande (en m3/s)
// ----------------------------------

   fd = [+0.08 -1.30 +0.13 +0.09 +0.16 +0.14 +0.12 +0.07 +0.17 +0.11...
         +0.25 +0.01 +0.13]';
```

D Calcul des structures de données : Structures_R.sce

```
////////////////////////////////////////////////////////////////////////////
//                                                                        //
// STRUCTURES DE DONNEES NECESSAIRES A LA RESOLUTION DES EQUATIONS DU RESEAU //
//                                                                        //
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
// A partir des deux vecteurs contenant  respectivement les noeuds initiaux et
// finaux des arcs, on construit la matrice d'incidence de graphe noeuds-arcs.
//
// De la sous-matrice extraite de la matrice d'incidence noeuds-arcs du graphe
// en supprimant les lignes  correspondant aux noeuds reservoirs (mr premieres
// lignes de cette matrice), on extrait le plus grand bloc carre et inversible
// (on suppose que les colonnes qui forment ce bloc correspondent aux premiers
//  arcs dans la numerotation du graphe).
//
// On calcule le vecteur des debits admissibles du reseau (c.a.d. satisfaisant
// la 1-iere loi de Kirchhoff) associe au vecteur des debits nuls du co-arbre.
//
// Les tableaux et les variables utilises pour modeliser le reseau proviennent
// du script  Scilab Probleme.sce. On en rappelle ci-dessous la signification.
// Ils sont disponibles dans l'environnement  Scilab, et peuvent etre utilises
// par les fonctions necessaires a la resolution du probleme.
//
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////
//
// Dimensions du reseau
// --------------------
//
// n    : nombre d'arcs
// m    : nombre de noeuds
// mr   : nombre de noeuds de type reservoir
// md   : nombre de noeuds de type demande (= m-mr)
//
// Topologie du reseau
// -------------------
//
// orig : vecteur des numeros des noeuds initiaux des arcs : M(1,n)
// dest : vecteur des numeros des noeuds finaux   des arcs : M(1,n)
// absn : vecteur des abscisses des noeuds                 : M(1,m)
// ordn : vecteur des ordonnees des noeuds                 : M(1,m)
//
// Donnees hydrauliques
// --------------------
//
// r    : vecteur des resistances des arcs                 : M(n,1)
// pr   : vecteur des pressions des noeuds reservoirs      : M(mr,1)
// fd   : vecteur des flux des noeuds de demande           : M(md,1)
//
```

```
// Matrices d'incidence
// --------------------
//
// A    : matrice d'incidence noeuds-arcs du graphe        : M(m,n)
// Ar   : sous-matrice de A correspondant aux reservoirs   : M(mr,n)
// Ad   : sous-matrice complementaire de Ar pour A         : M(md,n)
// AdT  : plus grande sous-matrice carree inversible de Ad : M(md,md)
// AdI  : matrice inverse de AdT                           : M(md,md)
// AdC  : sous-matrice complementaire de AdT pour Ad       : M(md,n-md)
// B    : matrice d'incidence arcs-cycles du graphe        : M(n,n-md)
//
// Debit admissible
// ----------------
//
// q0   : vecteur des debits admissibles des arcs          : M(n,1)
//
// Variables hydrauliques
// ----------------------
//
// q    : vecteur des debits des arcs                      : M(n,1)
// z    : vecteur des pertes de charge des arcs            : M(n,1)
// f    : vecteur des flux des noeuds                      : M(m,1)
// p    : vecteur des pressions des noeuds                 : M(m,1)
//
//
////////////////////////////////////////////////////////////////////////////

// ----------------------------------------
// Matrice d'incidence noeuds-arcs du graphe
// ----------------------------------------

   A = zeros(m,n);
   for l = 1:n
      A(orig(l),l) = -1;
      A(dest(l),l) = +1;
   end

// ----------------------------------------
// Partition de A suivant le type des noeuds
// ----------------------------------------

   Ar = A(1:mr,:);
   Ad = A(mr+1:m,:);

// ------------------------------------------------
// Sous-matrice de Ad associee a un arbre et inverse
// ------------------------------------------------

   AdT = Ad(:,1:md);
   AdI = inv(AdT);

// ---------------------------------------
// Sous matrice de Ad associee a un coarbre
// ---------------------------------------
```

```
   AdC = Ad(:,md+1:n);

// -------------------------------
// Matrice d'incidence arcs-cycles
// -------------------------------

   B = [ -AdI*AdC ; eye(n-md,n-md) ];

// -------------------------------
// Vecteur des debits admissibles
// -------------------------------

   q0 = [ AdI*fd ; zeros(n-md,1) ];
```