INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE RENNES
Département Génie Mathématique
Promotion 2022

Projet d'Optimisation en Grandes Dimensions

par

Hiba Shaimed, Xuan Vinh Nguyen, Victor Klötzer et Kefan Sun

Rennes, 6 avril 2021

Table des matières

Avant propos

Dans le projet, pour quasiment tous les algorithmes de décomposition, on a fait le choix de conserver et de retourner les solutions primales ($u^{(k)}$) et duales ($p^{(k)}$) obtenues à chaque itération. Ceci nous a permit de débuger, de tracer certains des graphiques et d'essayer d'avoir des fonctions générales. Il faut néanmoins noter qu'en pratique, à itération k, on peut écraser la solution à l'itération $k-1$ pour gagner en temps de calcul (on a d'ailleurs fait cela dans l'exercice 4).

Tous les codes sont disponibles sur Github : https://github.com/Kefan-pauline/OGS-project

Même si certainement tous les résultats de ce projet ne sont pas parfaits, nous y avons consacré de nombreuses heures et avons essayé de faire de notre mieux. Bonne lecture !

1 Exercice 1

Dans cet exercice, on veut comparer les trois algorithmes de décomposition. Les contraintes (1) peuvent être écrites sous forme matricielle : $Cu \leq d$ avec $C = \begin{pmatrix} 1 & 2 & & 0 \\ 0 & \ddots & \ddots & \\ & \ddots & \ddots & 2 \\ 0 & & 0 & 1 \end{pmatrix} \in \mathbb{R}^{N \times N}$ et $d = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^N$.

1.1 Question 1

On représente la fonction objectif et son domaine admissible K pour $N = 2$:

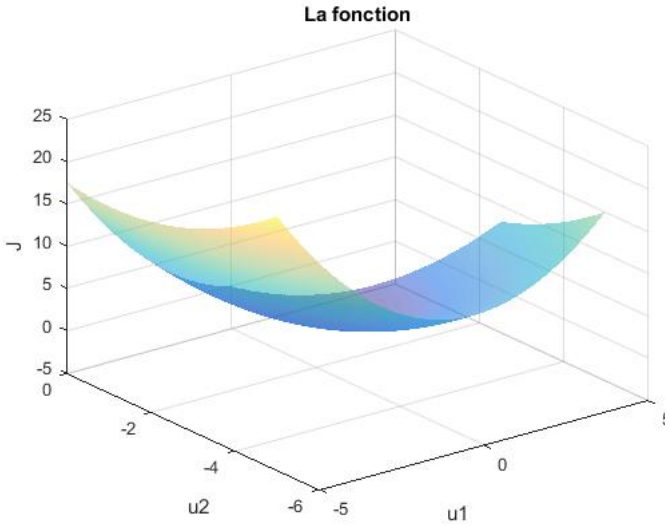

Fig. 1.1 – La fonction objectif

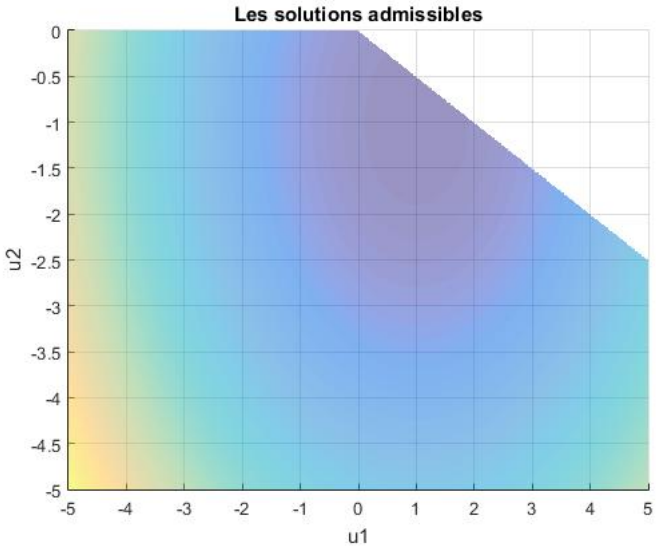

Fig. 1.2 – L'ensemble admissible

On remarque que la fonction objective est quadratique donc continue, convexe, lisse et coercive sur K, et que les contraintes sont linéaires donc le domaine admissible est convexe et fermé de dimension finie. Le problème admet donc une seule solution. D'après le second graphe, la solution optimale se trouve autour $u = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$.

1.2 Question 2

Le problème peut être décomposé comme :

$$J(u) = \sum_{i=1}^{N} J_i(u_i) = \sum_{i=1}^{N} (\frac{1}{2} A(i,i) u_i^2 - b_i u_i)$$

sous les contraintes :

$$\Theta(u) = \sum_{i=1}^{N} \theta_i(u_i) \leq d$$

où $\theta_1(u_1) = \begin{pmatrix} u_1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^N$, $\theta_2 = \begin{pmatrix} 2u_2 \\ u_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^N$, ..., $\theta_N = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 2u_N \\ u_N \end{pmatrix} \in \mathbb{R}^N$.

Le Lagrangien du problème global est donc $L(u,p) = J(u) + \langle p, \Theta(u) \rangle = \sum_{i=1}^{N} (J_i(u_i) + \langle p, \theta_i(u_i) \rangle)$. Le problème vérifie les hypothèses nécessaires pour admettre un point selle, de plus, la fonction objective $J(u)$ et la contrainte $\Theta(u)$ sont différentiables, donc tout point selle vérifie les conditions KKT.

1.2.1 Décomposition par prix

La décomposition par prix consiste à fixer un système de prix (identique pour toutes les unités) et à inciter les unités satisfaire les contraintes globales. Il est équivalent à l'algorithme de Uzawa.

A l'itération k, le prix $p^{(k)}$ est fixé, on résout tout d'abord la phase de minimisation de Lagrangien,

$$\inf_{u_i \in \mathbb{R}} J_i(u_i) + \langle p^{(k)}, \theta_i(u_i) \rangle, \forall i = 1, \ldots, N$$

ce qui nous donne une solution optimale $u_i^{(k+1)}$.

Ensuite, la maximisation par rapport à p revient à chercher le maximum de la fonction duale $\psi(p) = \inf_u L(u,p)$ $= L(u^{(k+1)}, p)$ dont le gradient est $\nabla\psi(p) = \sum_{i=1}^{N} \theta_i(u_i^{(k+1)})$. Par la méthode de gradient, on peut mettre à jour le prix par

$$p^{(k+1)} = \max(p^{(k)} + \rho_k \sum_{i=1}^{N} \theta_i(u_i^{(k+1)}), 0) \; avec \; \rho_k > 0.$$

Notre l'algorithme s'arrête lorsque le nombre maximal d'itération (1000) est épuisé, ou les conditions KKT sont vérifiées à $tol = 10^{-4}$ près, ou la solution u a convergé, i.e. $\|u^{(k+1)} - u^{(k)}\| < tol = 10^{-4}$.

Nous testons avec $N = 4$, l'algorithme converge en 15 itérations, nous obtenons $u = \begin{pmatrix} 1 \\ -1.2 \\ 0.6 \\ -1 \end{pmatrix}$, ce qui coïncide avec la solution trouvée par l'algorithme de Uzawa.

Un avantage de cette méthode est qu'il n'y a pas de contraintes dans chaque sous-problème, c'est donc plus facile de les résoudre. L'inconvénient est que la contrainte ne sera respecté qu'à la convergence de l'algorithme. Un autre inconvénient est que cet algorithme exige l'existence d'un point selle du Lagrangien du problème globale, ce qui ne peut être garantie sous les hypothèses de convexité, de qualification des contraintes etc.

1.2.2 Décomposition par quantité

La décomposition par quantité oblige chaque unité à produire une quantité donnée en respectant la contrainte globale et à trouver les meilleures allocations ensuite.

À l'itération k, les allocations $w_i^{(k)}, \forall i = 1, \ldots, N$ telles que $\sum_{i=1}^{N} w_i^{(k)} = d$ sont fixées, on résout tout d'abord la phase de minimisation du Lagrangien,

$$\inf_{u_i \in \mathbb{R}} J_i(u_i), \forall i = 1, \ldots, N$$

sous la contrainte

$$\theta_i(u_i) \leq w_i^{(k)}$$

ce qui nous donne une solution optimale $u_i^{(k+1)}$ et $p_i^{(k)}$ par l'algorithme de Uzawa.

Ensuite, par la méthode de gradient projeté appliqué au problème équivalent au problème global :

$$\inf_{W=(w_i)_{i=1,\ldots,N}} \sum_{i=1}^{N} G_i(w_i)$$

sous la contrainte

$$\sum_{i=1}^{N} w_i = d$$

avec $G_i(w_i) = J_i(\hat{u}_i(w_i))$ si $\hat{u}_i(w_i)$ est admissible pour le sous-problème i, $G_i(w_i) = \infty$ sinon. On pourrait mettre à jour les allocations par

$$w_i^{(k+1)} = w_i^{(k)} + \rho_k(p_i^{(k)} - \frac{1}{N}\sum_{j=1}^{N} p_j^{(k)}) \text{ avec } \rho_k > 0.$$

Notons qu'à la convergence, $p_i^* = \frac{1}{N}\sum_{j=1}^{N} p_j^*, \ \forall i = 1, \ldots, N$.

Notre algorithme s'arrête lorsque le nombre maximal d'itérations (1000) est épuisé, ou lorsque les conditions KKT sont vérifiées à $tol = 10^{-4}$ près, ou bien lorsque les allocations W ont convergé, c'est-à-dire que $\|W^{(k+1)} - W^{(k)}\| < tol = 10^{-4}$. De plus, on fait attention que $tol_{uzawa} < tol$.

Nous testons avec $N = 4$, l'algorithme converge en 132 itérations, nous obtenons $u = \begin{pmatrix} 1 \\ -1.2 \\ 0.6001 \\ -1 \end{pmatrix}$, ce qui coïncide avec la solution trouvée par l'algorithme de Uzawa.

Un avantage de cet algorithme est que les contraintes sont respectées à chaque itération. Mais l'ensemble des $u_i^{(k)}$ vérifiant les contraintes des sous-problèmes peut être vide si les contraintes des sous-problèmes ne sont pas compatibles avec les contraintes implicites.

1.2.3 Décomposition par prédiction

Cette décomposition est une combinaison des deux précédentes. En effet, on alloue les contraintes aux unités, ce qui correspond à la décomposition de l'espace $\mathbb{R}^m$. On introduit également un système de prix, ce qui permet d'incorporer la contribution de l'unité i aux contraintes allouées aux unités autres que i.

Dans notre exemple, on décide d'allouer la contrainte globale à l'unité i_0. Alors, à l'itération k, soient $w^{(k)}$ et $p^{(k)}$ connus, on résout pour l'unité i_0,

$$\min_{u_{i_0}} J_{i_0}(u_{i_0})$$

sous la contrainte

$$\theta_{i_0}(u_{i_0}) \leq w^{(k)}$$

ce qui nous donne une solution optimale $u_{i_0}^{(k+1)}$ et $\lambda_{i_0}^{(k+1)}$ par l'algorithme de Uzawa.

Pour les unités $i \neq i_0$, on résout

$$\min_{u_i} J_i(u_i) + < p^{(k)}, \theta_i(u_i) >$$

ce qui nous donne les solutions optimales $u_i^{(k+1)}$.

Les conditions KKT des sous-problèmes nous permettent de mettre à jour la prédiction par :

$$p^{(k+1)} = \lambda_{i_0}^{(k+1)}$$

$$w^{(k+1)} = d - \sum_{j \neq i_0} \theta_j(u_j^{(k+1)})$$

Pour accélérer la convergence, on peut sous- ou sur-relaxer la prédiction par :

$$p^{(k+1)} = (1-\beta)p^{(k)} + \beta\lambda_{i_0}^{(k+1)} \text{ avec } 0 < \beta < 1$$

$$w^{(k+1)} = (1-\gamma)w^{(k)} + \gamma(d - \sum_{j \neq i_0} \theta_j(u_j^{(k+1)})) \text{ avec } 0 < \gamma < 1$$

La coordination s'opère par échange directe d'informations entre les sous-problèmes.

Nous implémentons la version parallèle de l'algorithme cité ci-dessus, notre algorithme s'arrête lorsque le nombre maximal d'itérations (1000) est épuisé, ou lorsque les conditions KKT sont vérifiées à $tol = 10^{-4}$ près, ou bien lorsque la solution u ont convergé, i.e. $\|u^{(k+1)} - u^{(k)}\| < tol = 10^{-4}$. De plus, on fait attention que $tol_{uzawa} < tol$.

Nous testons avec $N = 4$, l'algorithme converge en 53 itérations, nous obtenons $u = \begin{pmatrix} 1 \\ -1.2007 \\ 0.5986 \\ -1 \end{pmatrix}$, ce qui est proche de la solution trouvée par l'algorithme de Uzawa.

L'avantage de cette méthode est qu'il y a un seul seul prix par rapport à la décomposition par quantité, il n'y a qu'un seul sous-problème qui a une contrainte explicite. Mais l'inconvénient est que, comme dans la décomposition par prix, les solutions des sous-problèmes ne vérifient la contrainte globale qu'à la convergence. Il se peut également que les sous-problèmes n'admettent pas de solution.

1.3 Question 3

Le Lagrangien est $L(u,p) = \frac{1}{2} < Au, u > - < b, u > + p(Cu - d)$, les conditions de KKT sont :

$$\begin{cases} Au^* - b + C^T p* = 0_m \\ Cu^* - d \leq 0_m \\ p* \in \mathbb{R}_m^+ \\ p^*(Cu^* - d) = 0_m \end{cases}$$

On trace pour les 3 algorithmes, $\|u^{(k)} - u^*\|^2$ pour chaque itération k, le point est en rouge si les conditions de KKT ne sont pas vérifiées à $tol = 10^{-4}$ près, et vert si les conditions sont vérifiées.

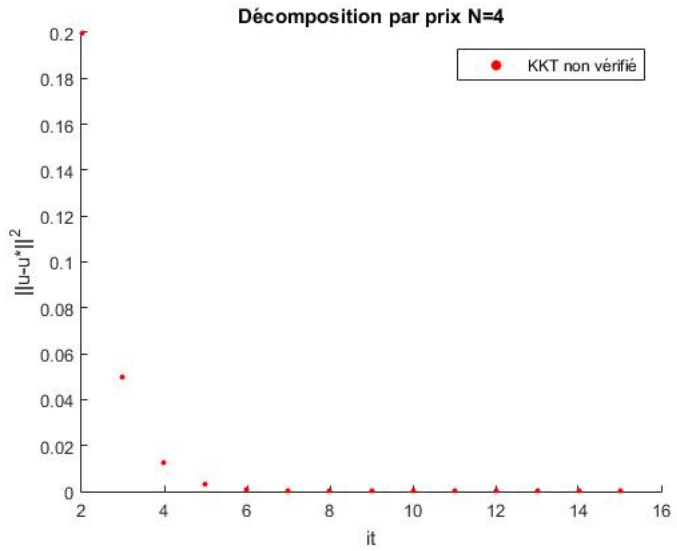

Fig. 1.3 – Résultat par prix

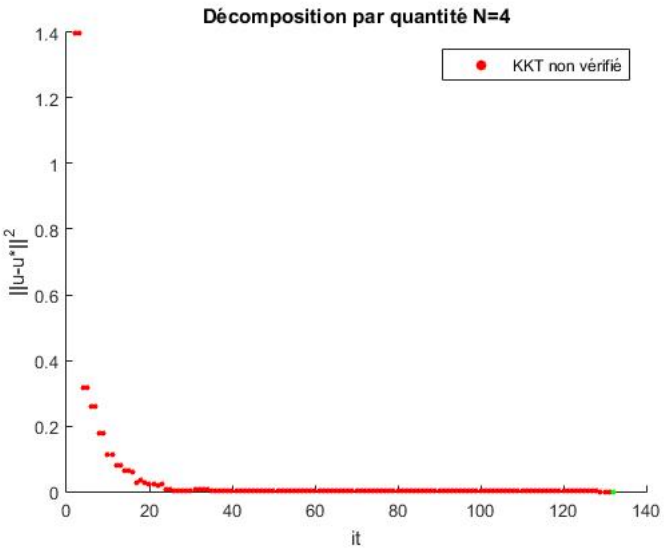

Fig. 1.4 – Résultat par quantité

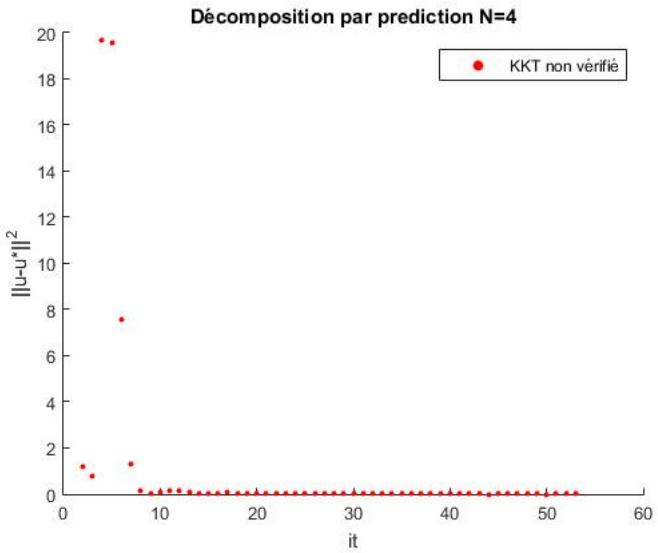

Fig. 1.5 – Résultat par prédiction

On peut voir que l'algorithme le plus performant est la décomposition par prix, il prend le moins d'itération pour converger. La décomposition par quantité prend le plus d'itérations, mais il est le seul algorithme qui

vérifie les conditions KKT à la convergence. La décomposition par prédiction se trouve au milieu probablement car c'est une combinaison des 2 algorithmes.

1.4 Question 4

Nous lançons les 3 algorithmes pour $N = 2, 4, \ldots, 26$, en fixant $\rho = 0.2$ et $\rho_{uzawa} = 0.1$, on récupère ensuite le nombre d'itérations et le temps pour chaque algorithme et on trace en échelle loglog.

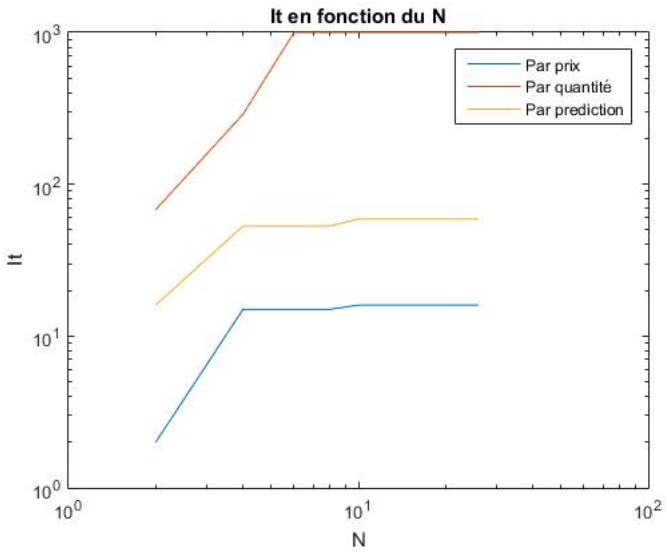

Fig. 1.6 – Complexité en itération

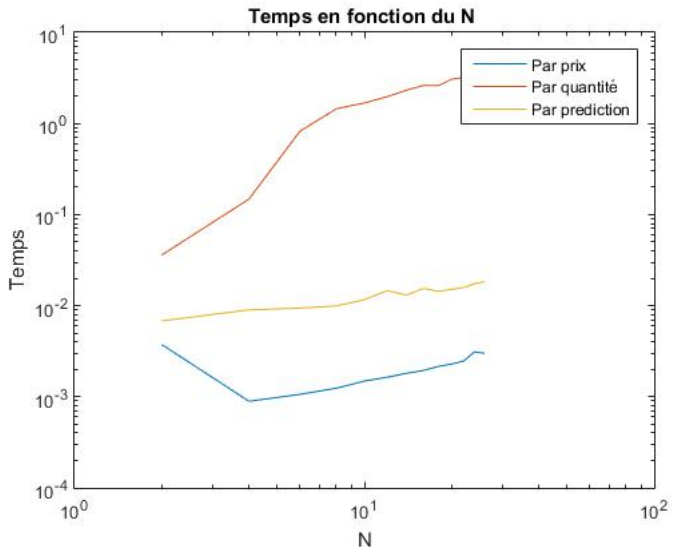

Fig. 1.7 – Complexité en temps

Plus N est grand, plus le problème est difficile à résoudre. L'algorithme le plus efficace est la décomposition par prix, puis la décomposition par prédiction. La décomposition par quantité explose facilement en nombre d'itération et en temps. Cela peut être expliqué par le fait qu'il y a le plus de contraintes dans les sous-problèmes pour la décomposition par quantité. Le nombre de contraintes explose quand N devient grand, cela rend le problème encore plus difficile à résoudre.

1.5 Question 5

Pour $tol = 10^{-4}$ et $\rho = \rho_{uzawa} = 0.1$, on compare le nombre d'itérations et le temps nécessaire à chaque méthode en fixant $N = 200$ puis $N = 251$.

N=200	Temps	It
Prix	0.0209	16
Quantité	12.6370	1000
Prédiction	0.0795	63

N=251	Temps	It
Prix	0.0984	72
Quantité	29.0164	1000
Prédiction	0.1081	71

La décomposition par quantité est la moins performante en temps et en nombre d'itération. Pour N pair, la décomposition par prix est la plus efficace en temps et en itération. Pour N impair, la décomposition par prix et la décomposition par prédiction sont de même ordre de grandeur en temps et en itérations.

Maintenant nous étudions le temps et le nombre nécessaire d'itérations pour chaque méthode en fonction de la tolérance.

Pour $N = 200$, en fixant $\rho = 0.1$ et $\rho_{uzawa} = 0.1$, on obtient en échelle loglog :

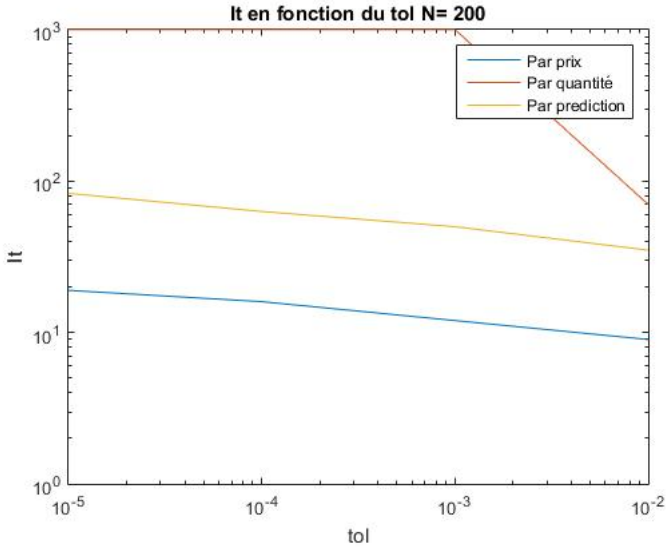

Fig. 1.8 – Itération en fonction de la tolérance

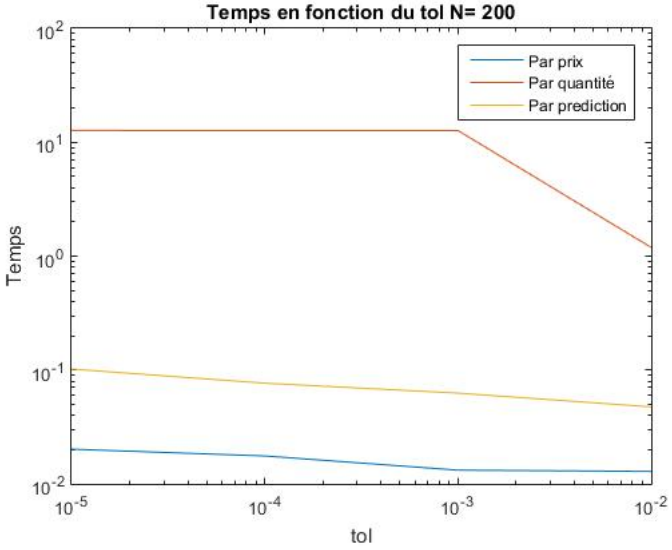

Fig. 1.9 – Temps en fonction de la tolérance

Pour $N = 251$, en fixant $\rho = 0.08$ et $\rho_{uzawa} = 0.1$, on obtient en échellel loglog :

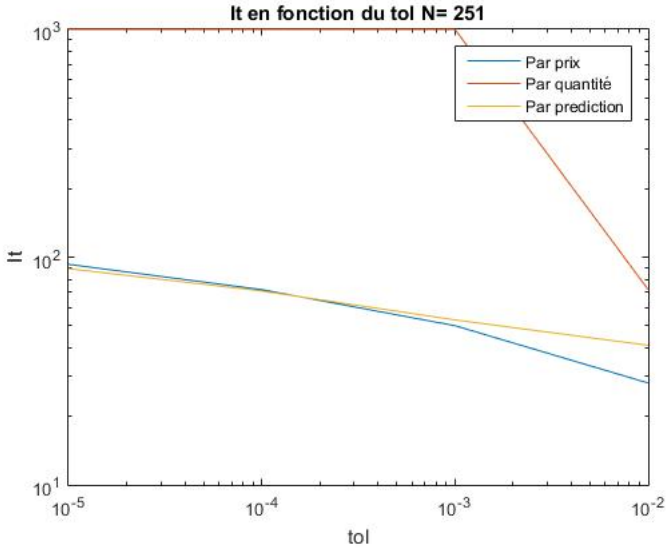

Fig. 1.10 – Itération en fonction de la tolérance

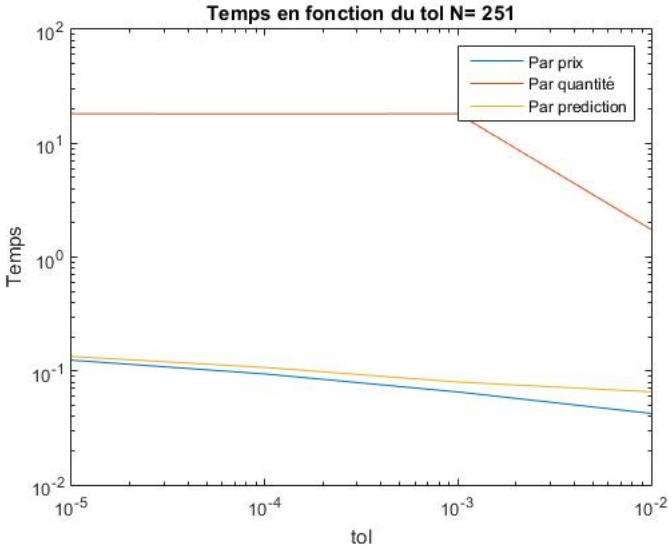

Fig. 1.11 – Temps en fonction de la tolérance

Plus la tolérance est petite, plus il faut d'itérations pour converger. On observe également que la décomposition par quantité est beaucoup plus sensible à la tolérance que les deux autres algorithmes. La méthode la plus performante est la décomposition par prix quand N est pair. Quand N est impair et la tolérance est petite, la décomposition par prix et la décomposition par prédiction sont de même ordre de grandeur.

Ensuite, nous étudions le temps et le nombre nécessaire d'itération pour chaque méthode en fonction des valeurs du pas.

Nous varions tout d'abord la valeur ρ_{uzawa}, qui intervient dans la décomposition par prix pour la mise-à-jour du prix, dans la décomposition par quantité et par prédiction pour la résolution des sous-problèmes.

Pour $N = 8$, en fixant $\rho = 0.1$ et $tol = 10^{-4}$, on obtient en échelle loglog :

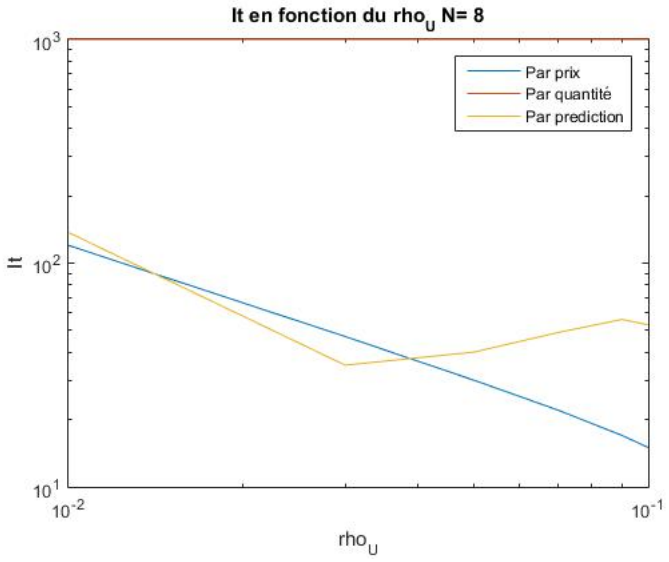

Fig. 1.12 – Itération en fonction de ρ_{uzawa}

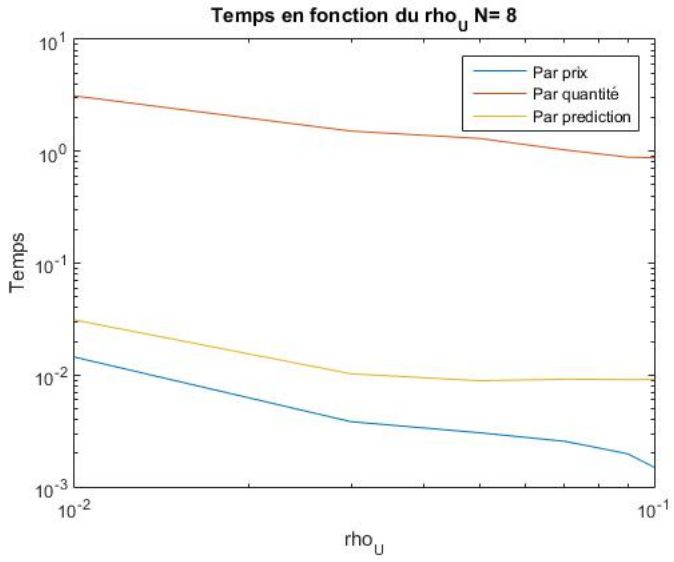

Fig. 1.13 – Temps en fonction de ρ_{uzawa}

Pour $N = 9$, en fixant $\rho = 0.1$ et $tol = 10^{-4}$, on obtient en échelle loglog :

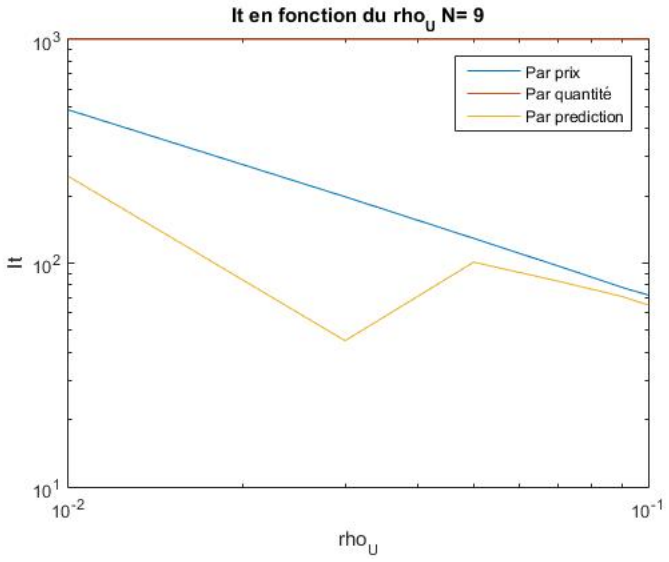

Fig. 1.14 – Itération en fonction de ρ_{uzawa}

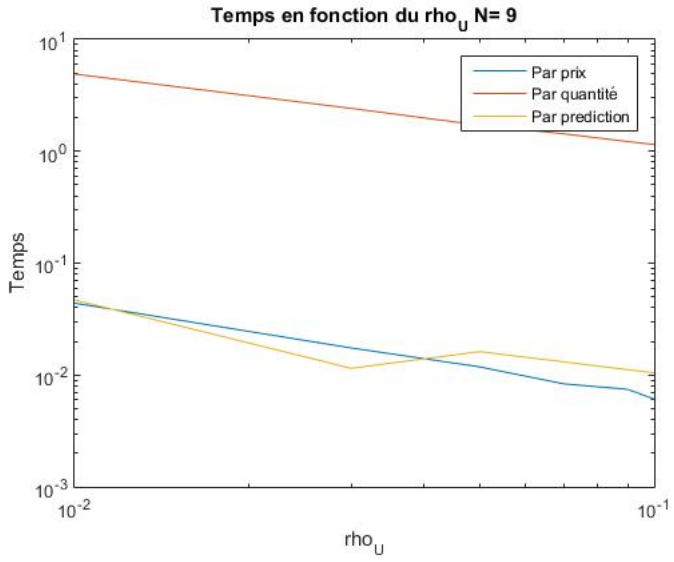

Fig. 1.15 – Temps en fonction de ρ_{uzawa}

Dans tous les 4 graphes, plus le pas est petit, plus qu'il faut d'itérations et du temps pour converger. C'est logique car le pas influence la vitesse avec laquelle on converge vers la solution optimale. Dans le cas où N est impair, la valeur optimale pour la décomposition par prédiction est $\rho_{uzawa} = 0.05$.

Nous varions maintenant la valeur ρ qui intervient dans la décomposition par quantité pour la mise-à-jour des allocations.

Pour $N = 8$, en fixant $\rho_{uzawa} = 0.1$ et $tol = 10^{-4}$, on obtient :

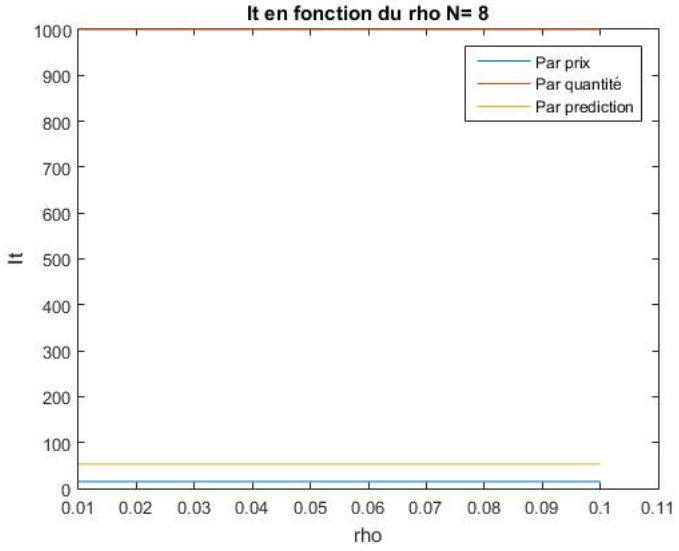

Fig. 1.16 – Itérations en fonction de ρ

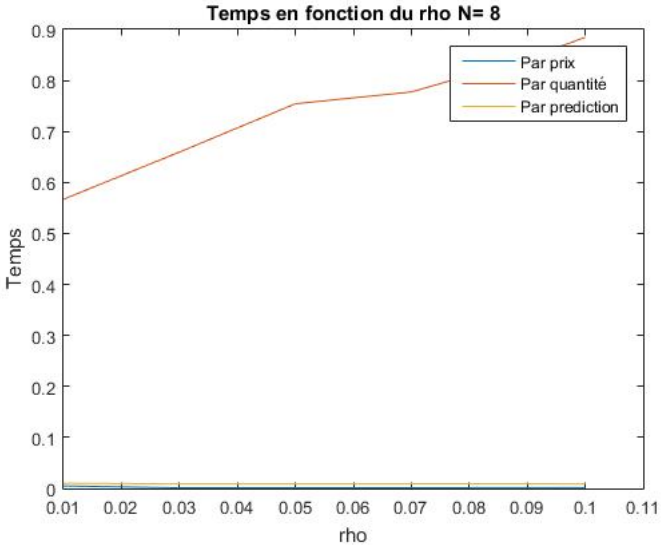

Fig. 1.17 – Temps en fonction de ρ

Pour $N = 9$, en fixant $\rho_{uzawa} = 0.1$ et $tol = 10^{-4}$, on obtient :

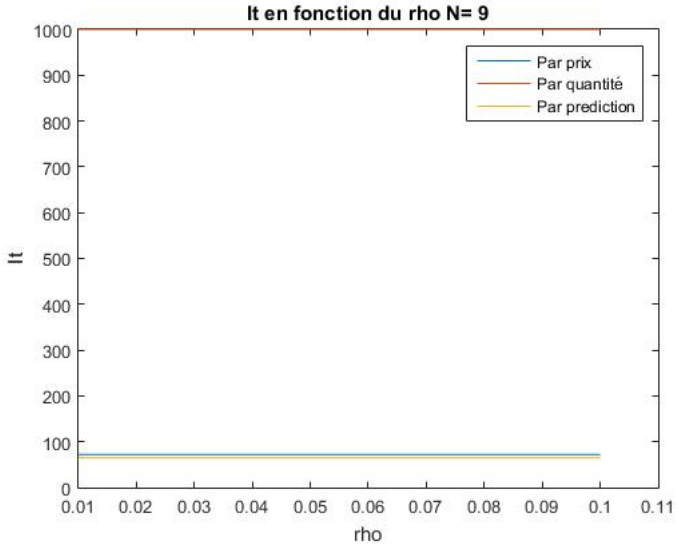

Fig. 1.18 – Itérations en fonction de ρ

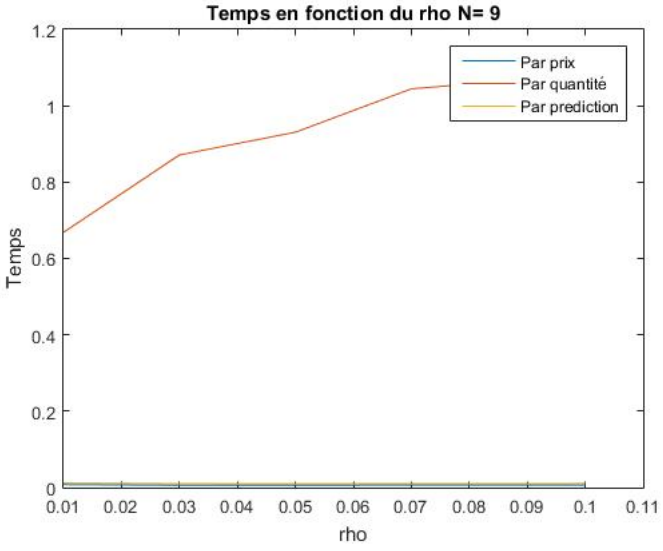

Fig. 1.19 – Temps en fonction de ρ

On a l'impression que plus ρ est petit, plus la décomposition par quantité est performante en terme du temps. En effet, quand ρ est trop petit, la mise-à-jour des allocations a un effet très petit, donc les allocations changent peu entre une itération et une autre, les sous-problèmes changent aussi peu. L'algorithme de Uzawa met moins de temps à résoudre chaque sous-problème, mais la solution ne converge pas encore à l'issue de 1000 itérations.

1.6 Question 6

Si on ajoute une sur-diagonale dans la matrice A, on introduirait un terme supplémentaire $-\alpha(u_1u_2 + u_2u_3 + \cdots + u_{N-1}u_N)$ dans la fonction objectif. Le problème devient donc non-décomposable si $\alpha \neq 0$. On ne peut plus utiliser les trois algorithmes de décomposition, qui ne prennent que les termes diagonaux de la matrice A en compte, ils nous donneront donc les même résultats que pour $\alpha = 0$.

Pour $N = 4$ et $\alpha = 2$, la solution optimale de problème est $u = \begin{pmatrix} -5 \\ -3 \\ -1 \\ -1 \end{pmatrix}$ par Uzawa, pour $\alpha = -2$ la solution est $u = \begin{pmatrix} 7.2222 \\ -3.1111 \\ 0.7778 \\ -1 \end{pmatrix}$ par Uzawa.

On pourrait néanmoins utiliser le principe du problème auxiliaire pour faire la décomposition, cette démarche sera détaillée dans l'exercice 3.

2 Exercice 2

Dans cet exercice, on s'intéresse au problème d'optimisation de N sous-réseaux de distribution d'eau connectés par une usine de refoulement. On peut formuler le problème sous la forme suivante :

$$\begin{cases} \min\limits_{(u_i=(u_i^{(1},u_i^{(2)})_{i=1,...,N+1}} \sum\limits_{i=1}^{N+1} J_i(u_i) \\ \textbf{sous} \qquad \sum_{i=1}^{N} u_i = u_{N+1} \end{cases}$$

avec $J_{N+1}(u_{N+1}) = \frac{1}{2}\langle A_{N+1}u_{N+1}, u_{N+1}\rangle$ et pour $i = 1, ..., N$, $J_i(u_i)$ la fonction définie par la minimisation :

$$J_i(u_i) = \begin{cases} \min\limits_{(v_i=(v_i^{(1)},v_i^{(2)}))} J_i(v_i) = \dfrac{1}{2}\langle A_i v_i, v_i\rangle \\ \textbf{sous} \qquad -v_i^{(1)} - u_i^{(1)} + a_i \leq 0 \\ \qquad\qquad v_i^{(1)} + v_i^{(2)} + u_i^{(1)} + u_i^{(2)} - b_i = 0 \\ \qquad\qquad u_i \in [0, a_i] \times [0, b_i - a_i] \end{cases}$$

Pour résoudre ce problème, nous allons présenter trois méthodes de décomposition : par prix, par quantités et par prédiction et les comparer entre elles.

2.1 Question 1

2.1.1 Décomposition par prix

À l'itération k de l'algorithme, le prix $p^{(k)}$ est fixé , les sous-problèmes s'écrivent sous la forme :

$$\begin{cases} \min\limits_{(u_i\in\mathbb{R}^2)_{i=1,...,N}} J_i(u_i) + \langle p^{(k)}, u_i\rangle \\ \min\limits_{(u_{N+1}\in\mathbb{R}^2} J_{N+1}(u_{N+1}) + \langle p^{(k)}, -u_{N+1}\rangle \end{cases}$$

Ce qui nous donne $\forall i = 1, ..., N + 1$ les solutions $u_i^{(k+1)}$. Ensuite, l'étape de coordination consiste à mettre à jour les prix :

$$p^{(k+1)} = p^{(k)} + \rho\big(\sum_{i=1}^{N} u_i^{k+1} - u_{N+1}^{(k+1)}\big)$$

En introduisant les variables v_i les contraintes locales aux sous-réseaux i, on obtient les problèmes de mini-

misation sous contraintes pour les sous-réseaux $i = 1, ..., N$:

$$\begin{cases} \min\limits_{(u_i,v_i)\in\mathbb{R}^4} \dfrac{1}{2}\langle A_i v_i, v_i\rangle + \langle p^{(k)}, u_i\rangle \\ \quad \textbf{sous} \qquad -v_i^{(1)} - u_i^{(1)} + a_i \leq 0 \\ \qquad\qquad\quad v_i^{(1)} + v_i^{(2)} + u_i^{(1)} + u_i^{(2)} - b_i = 0 \\ \qquad\qquad\quad u_i \in [0, a_i] \times [0, b_i - a_i] \end{cases}$$

En revanche, le sous-problème $N + 1$ associé à l'usine de refoulement n'a pas de contrainte et peut s'écrire :

$$\min_{(u_{N+1})\in\mathbb{R}^2} \frac{1}{2}\langle A_{N+1} u_{N+1}, u_{N+1}\rangle + \langle p^{(k)}, -u_{N+1}\rangle$$

On obtient donc que la solution du sous-problème N + 1 associé à l'usine de refoulement est $u_{N+1,j}^{(k+1)} = \frac{p^{(k)}}{\alpha_{N+1,j}}$ pour $j = 1, 2$.

Avec cette décomposition par prix, on a un problème à 6 contraintes.

2.1.2 Décomposition par quantités

On utilise maintenant une décomposition par allocation pour résoudre le problème. On introduit donc une allocation $w_i^{(0)}$, $i = 1, ..., N$ pour chaque sous-réseaux et $w_{N+1}^{(0)} = -\sum_{i=1}^{N} w_i^{(0)}$ pour l'usine de refoulement qui évoluera au cours des itérations. L'allocation initiale $w^{(0)}$ doit être réalisable pour le problème. D'où les sous problèmes s'écriraient, à l'itération k, pour $i = 1, ..., N$ de la manière suivante :

$$\begin{cases} \min\limits_{v_i\in\mathbb{R}^{\not 4}} \dfrac{1}{2}\langle A_i v_i, v_i\rangle \\ \textbf{sous} \qquad -v_i^{(1)} - u_i^{(1)} + a_i \leq 0 \\ \qquad\qquad v_i^{(1)} + v_i^{(2)} + u_i^{(1)} + u_i^{(2)} - b_i = 0 \\ \qquad\qquad u_i = w_i^{(k)} \\ \qquad\qquad u_i \in [0, a_i] \times [0, b_i - a_i] \end{cases}$$

et pour $i = N + 1$:

$$\begin{cases} \min\limits_{(u_{N+1})\in\mathbb{R}^{\not 4}} \dfrac{1}{2}\langle A_{N+1} u_{N+1}, u_{N+1}\rangle \\ \textbf{sous} \qquad u_{N+1} = -w_{N+1}^{(k)} \end{cases}$$

En résolvant les sous-problèmes, on obtient la solution évidente $u_i^{(k+1)} = w_i^{(k)}$ pour $i = 1, ..., N$ et $u_{N+1}^{(k+1)} = -w_N^{(k)}$. L'étape de coordination consiste à mettre à jour les quantités :

$$\forall i = 1, ..., N + 1, \quad w_i^{k+1} = w_i^{(k)} + \rho\big(p_i^k - \frac{1}{N+1}\sum_{j=1}^{N+1} p_j^{(k)}\big)$$

où $p_i^{(k)}$ est le multiplicateur de Lagrange associé à la contrainte $u_i = w_i^{(k)}$

En remplaçant $u_i^{(k)} = w_i^{(k)}$, il suffit d'optimiser le problème avec seulement 2 contraintes. Cela simplifie énormément le problème.

2.1.3 Décomposition par prédiction

Enfin, on va utiliser une méthode de décomposition par prédiction pour résoudre notre problème. Nous allons affecter la contrainte couplante au sous-problème $N+1$, associé à l'usine de refoulement. On introduit un prix p_{N+1} qui évoluera au cours des itérations. La version parallèle de la décomposition par prédiction s'écrit :

1. Initialisation des prédictions pour les prix $p_{N+1}^{(0)}$ et pour le niveau de production $u_{(N+1)}^{(0)}$
2. À l'étape k, on résout :
   a) Pour l'unité $N+1$, à production $u_i^{(k)}$ $\forall i = 1, ..., N$ fixées :

$$\begin{cases} \min\limits_{u_{N+1} \in \mathbb{R}^{\not\in}} & J_{N+1}(u_{N+1}) \\ \textbf{sous} & u_{N+1} = \sum_{i=1}^{N+1} u_i^{(k)} \end{cases}$$

   On obtient $p_{N+1}^{(k+1)} = \lambda_{N+1}^{(k+1)}$ et $u_{N+1}^{(k)}$, où $\lambda_{N+1}^{(k+1)}$ le multiplicateur optimal de Lagrange.

   b) pour les unités $i \neq N+1$, à prix $p_{N+1}^{(k)}$ fixé :

$$\begin{cases} \min\limits_{u_i \in \mathbb{R}^{\not\in}} & J_i(u_i) + \langle p_{N+1}^{(k)}, u_i \rangle \end{cases}$$

   On obtient $u_i^{(k+1)}$

Dans la version séquentielle, dans le cas où on commence par l'unité $N+1$:

1. Initialisation des prédictions pour les prix $p_{N+1}^{(0)}$ et pour le niveau de production $u_{N+1}^{(0)}$
2. À l'étape k, on résout :
   a) Pour l'unité $N+1$, à production $u_i^{(k)}$ $\forall i = 1, ..., N$ fixées :

$$\begin{cases} \min\limits_{u_{N+1} \in \mathbb{R}^2} & J_{N+1}(u_{N+1}) \\ \textbf{sous} & u_{N+1} = \sum_{i=1}^{N+1} u_i^{(k)} \end{cases}$$

   On obtient $p_{N+1}^{(k+1)} = \lambda_{N+1}^{(k+1)}$ et $u_{N+1}^{(k)}$, où $\lambda_{N+1}^{(k+1)}$ le multiplicateur optimal de Lagrange.

   b) pour les unités i ≠ N+1, à prix $p_{N+1}^{(k)}$ fixé :

$$\begin{cases} \min\limits_{u_i \in \mathbb{R}^{\not\in}} & J_i(u_i) + \langle p_{N+1}^{(k+1)}, u_i \rangle \end{cases}$$

   On obtient $u_i^{(k+1)}$

2.2 Question 2

La version parallèle et séquentielle ont la même complexité de calcul donc cela ne change pas les temps de calcul. Par contre, si on introduit les techniques de calcul parallèle (par exemple OpenMP en C), la version parallèle aurait été plus rapide. Un avantage de la version séquentielle est que la solution est admissible à chaque itération.

2.3 Question 3

Pour la décomposition par les prix, en observant le temps d'exécution pour $\epsilon = 10^{-3}$ et pour des pas ρ différents, on constate qu'il n'y a pas de grande différence en temps pour des pas différentes. Les paramètres $a_i, b_i, \alpha_i, \beta_i$ sont choisis aléatoirement pour chaque exécution.

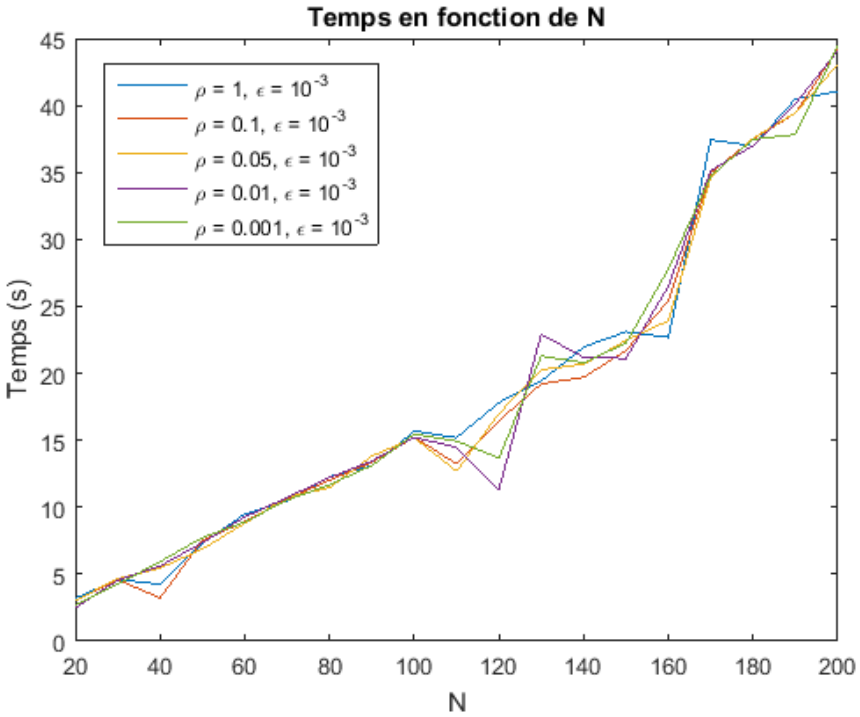

Fig. 2.1 – Temps d'exécution de la décomposition par prix en fonction de N

3 Exercice 3

Dans cet exercice on souhaite utiliser les méthodes de décomposition ainsi que le principe du problème auxiliaire pour répondre à un problème réaliste. On s'intéresse ainsi à l'optimisation d'un portefeuille d'actions (modèle de Markowitz) :

Soient N actions représentées par des variables aléatoires $(X_i)_{i=1,\dots,N}$ qui rapportent en moyenne $e_i := \mathbb{E}(x_i)$ au bout d'un an. On connaît de plus la matrice de covariances Q de ces actions. Une personne veut investir une certaine somme d'argent dans ces N actions et l'on représente par les variables $(u_i)_{i=1,\dots,N} \in \mathbb{R}^N$ la proportion de la somme investie dans chacune des actions (on a donc $\sum_{i=1}^{N} u_i = 1$). Le portefeuille total de cette personne peut être représenté par la variable $X := \sum_{i=1}^{N} u_i X_i$ qui rapporte en moyenne $\mathbb{E}(x) = \sum_{i=1}^{N} u_i e_i$ en encourant le risque $var(X) = \langle u, Qu \rangle$, où $u := (u_i)_{i=1,\dots,N}$.

L'objectif de l'exercice alors est d'aider la personne à déterminer comment répartir son argent entre ces actions, i.e. trouver les $(u_i)_{i=1,\dots,N}$. On peut ainsi s'intéresser à deux différentes stratégies de répartition :

- problème 1 : on veut minimiser le risque encouru tout en s'imposant un rendement minimal $R_e > 0$,
- problème 2 : ou bien on veut maximiser le gain tout en se fixant un certain risque maximal $D_e > 0$

Dans la suite, on va essayer de résoudre ces deux problèmes en considérant d'abord que la matrice des covariances est diagonale (les problèmes sont alors décomposables), puis dans un second temps on prendra en compte l'existence de corrélations entre les actions.

Générations de données

On doit tout d'abord se créer des données pour tester les futurs algorithmes. On propose de générer aléatoirement N actions sur 12 mois en tirant des échantillons de taille 12 de lois normales $\mathcal{N}(m_i, \sigma_i^2)_{i=1,\dots,N}$. Pour obtenir des moyennes m_i et des écarts-types s_i pour ces lois normales, on propose de tirer les moyennes aléatoirement d'une loi uniforme ($\mathcal{U}([10, 13])$ dans l'exemple ci-dessous) et de même pour les écarts-types ($\mathcal{U}([1, 4])$ ci-dessous). On procède ainsi afin d'obtenir des actions avec des moyennes et des variances différentes.

Pour obtenir ensuite les gains moyens $e := (e_i)_{i=1,\dots,N}$ ainsi que les risques Q des ces actions, il suffit de prendre l'espérance expérimentale et la matrice de covariance expérimentale respectivement.

Concernant le rendement minimal R_e, on choisit de le prendre égal à la moyenne des gains e_i (c'est le gain que l'on obtiendrait si l'on répartissait uniformément l'argent sur les N actions). Quant au risque maximal encouru D_e, on le prend égal à la plus petite des variances des actions, i.e. $D_e = \min(diag(Q))$ (c'est le risque que l'on prendrait si tout l'argent était investi dans l'action avec la variance minimale).

On souhaitera traiter le même exemple avec tous les algorithmes, on fixe donc les graines de génération de nombres aléatoire et on obtient pour $N = 4$ les actions suivantes :

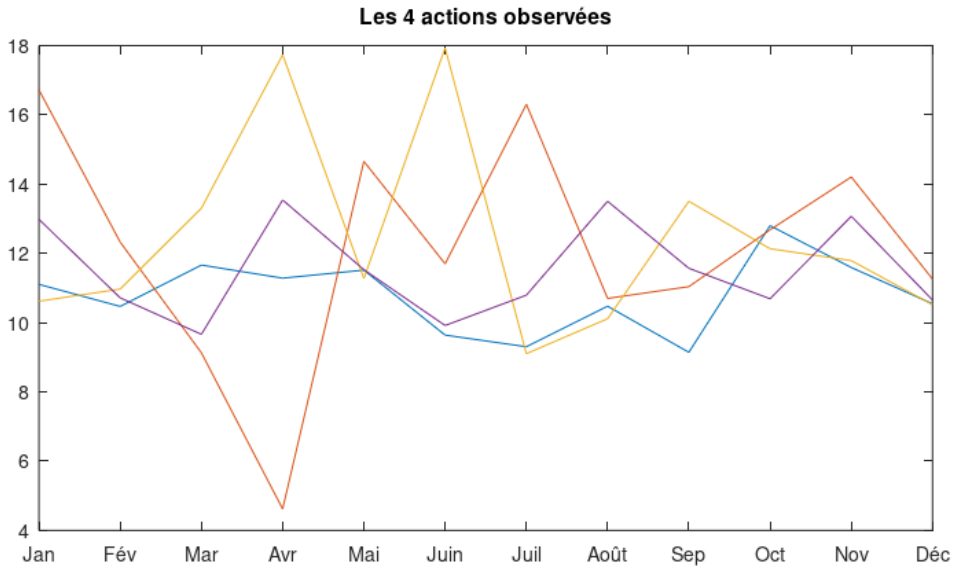

Fig. 3.1 – Les 4 actions générées aléatoirement qui seront utilisé comme exemple jouet dans cet exercice

Ce sont ces quatre actions qui seront utilisées par la suite pour tester les algorithmes (exemple jouet). On utilisera ainsi les données suivantes : $e \approx (10.80, 12.11, 12.42, 11.55)^T$, $R_e \approx 11.72$ et $D_e \approx 1.17$. Q sera reprécisé plus bas en fonction de la présence ou non des corrélations.

3.1 Problème 1

Le problème 1 peut se formuler de la manière suivante :

$$\begin{aligned} \min_{u \in \mathbb{R}^N} \quad & \frac{1}{2}\langle u, Qu \rangle & \\ \text{sous} \quad & \sum_{i=1}^{N} u_i = 1 & (1) \\ & \sum_{i=1}^{N} u_i e_i \geq R_e & (2) \\ & u_i \geq 0, \forall i \in [\![1, N]\!] & (3) \end{aligned}$$

On peut d'ores et déjà noter une chose : une matrice de covariance (ou corrélations) est symétrique et surtout toujours semi-définie positive ce qui garantira donc toujours la **convexité de la fonction objectif** de ce problème.

En effet, si $\{Y_1, ..., Y_n\}$ est une famille de variables aléatoires et si on note $Z := (Y_1 - \mathbb{E}(Y_1), ..., Y_n - \mathbb{E}(Y_n))^T$ le vecteur centré associé, alors la matrice de covariances de $\{Y_1, ..., Y_n\}$ est $C := Cov(Y_1, ..., Y_n) = \mathbb{E}[ZZ^T]$. Supposons maintenant que C ne soit pas semi-définie positive, alors il existe un $v \in \mathbb{R}^n$ tel que : $v^T C v < 0$. Or on a aussi $v^T C v = v^T \mathbb{E}[ZZ^T] v = \mathbb{E}[v^T Z Z^T v] = \mathbb{E}[(Z^T v)^T (Z^T v)] = \mathbb{E}[(Z^T v)^2] \geq 0$, ce qui mène à une contradiction. La matrice de covariance C est donc semi-définie positive.

3.1.1 Sans corrélation entre les actions

Pour commencer, on considère que la matrice Q est diagonale, i.e. il n'y a pas de corrélation entre deux différentes actions. Dans ce cas, le problème est quadratique et donc directement décomposable à l'aide d'un algorithme de décomposition (par prix, par ressources ou par prédictions).

Avec les quatre actions générées ci-dessus, on a donc ici : $Q \approx \begin{bmatrix} 1.17 & 0 & 0 & 0 \\ 0 & 10.83 & 0 & 0 \\ 0 & 0 & 7.97 & 0 \\ 0 & 0 & 0 & 1.93 \end{bmatrix}$.

Pour résoudre ce problème, on utilise les trois algorithmes de décomposition présentés en cours en les appliquant simplement à ce problème quadratique. On essaie de choisir des hyperparamètres qui permettent de converger rapidement et on obtient les résultats suivant (voir les codes Matlab pour plus de détails) :

```
U_decomp_prix = 0.5369  0.064812  0.09027  0.3474
valeur_optimale = 0.33958
nombre_iterations = 239
Les conditions KKT sont vérifiées au seuil d'erreur de 0.05
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
U_decomp_ressources = 0.53584  0.064977  0.090573  0.34796
valeur_optimale = 0.33963
nombre_iterations = 157
Les conditions KKT sont vérifiées au seuil d'erreur de 0.05
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
U_decomp_pred = 0.53701  0.064865  0.090354  0.3476
valeur_optimale = 0.33988
nombre_iterations = 15
Les conditions KKT sont vérifiées au seuil d'erreur de 0.05
```

Les trois algorithmes respectent les conditions KKT au seuil d'erreur de 0.05 et semblent bien converger vers la même solution approchée $u = (u_1, u_2, u_3, u_4) \approx (0.54, 0.06, 0.09, 0.35)$ avec pour valeur optimale environ 0.34. Ici, on conseillerait ainsi à la personne qui investi son argent d'en mettre environ 54% sur la première action, environ 6% sur la deuxième, environ 9% sur la troisième et environ 35% sur la quatrième action avec un risque de minimal de $2 \times 0.34 = 0.68$ (qui est bien inférieur à $D_e = 1.17$ d'ailleurs). Enfin, on constate des différences dans le nombre d'itérations effectuées qui semble indiquer que la décomposition parallèle peut être plus rapide que les deux autres.

3.1.2 Avec des corrélations entre les actions

On considère désormais qu'il y a des corrélations entre des actions, i.e. Q n'est plus diagonale. Le problème n'est alors plus décomposable car des termes en $u_i u_j$, avec $i \neq j$, sont ajoutés dans la fonction objectif. On propose donc d'utiliser le principe du problème auxiliaire pour reformuler le problème et pouvoir utiliser un algorithme de décomposition pour le résoudre.

On note d'abord $\lambda \in \mathbb{R}$, $\mu_0 \in \mathbb{R}_+$ et $\mu \in \mathbb{R}_+^N$ les multiplicateurs de Lagrange associées aux contraintes (1), $(2')$ et $(3')$ respectivement, où $(2')$ et $(3')$ sont les réécritures "naturelles" des contraintes (2) et (3) :

$$-\sum_{i=1}^{N} u_i e_i \leq -R_e \quad (2') \qquad \text{et} \qquad -u_i \leq 0, \forall i \in [\![1, N]\!] \quad (3')$$

(On précise ici cette réécriture car c'est ainsi que le problème est implémenté dans les codes Matlab.)

La fonction lagrangienne associé au problème est alors :

$$\tilde{L}(u,p) := \frac{1}{2}\langle u, Qu\rangle + p^T\Theta(u)$$

où $p := (\lambda, \mu_0, \mu)^T$ et $\Theta(u) := (\sum_{i=1}^n u_i - 1, R_e - \sum_{i=1}^n u_i e_i, -u)^T$

En notant $D := diag(Q)$, on peut réécrire ce Lagrangien comme suit :

$$\tilde{L}(u,p) = \frac{1}{2}\langle u, (Q-D)u\rangle + \frac{1}{2}\langle u, Du\rangle + p^T\Theta(u) = L(u) + L^A(u,p)$$

où $L^A(u,p) := \frac{1}{2}\langle u, Du\rangle + p^T\Theta(u)$ est décomposable et $L(u) := \frac{1}{2}\langle u, (Q-D)u\rangle$.

Afin de rendre le problème décomposable, on introduit maintenant dans ce problème auxiliaire avec contraintes, le terme auxiliaire suivant (à l'itération k) :

$$A^{(k)}(u,p) := \langle u, (Q-D)u^{(k)}\rangle + \frac{1}{2}\langle u, Du\rangle - \frac{1}{2}\langle p, p\rangle$$

dans lequel les termes $\frac{1}{2}\langle u, Du\rangle$ et $-\frac{1}{2}\langle p, p\rangle$ permettent d'assurer pour $A^{(k)}(u,p)$ la forte convexité en u et la forte concavité en p.

On obtient ainsi le Lagrangien associé au problème auxiliaire (en fixant $\epsilon^{(k)}$ à 1) :

$$\begin{aligned}
\mathcal{L}^{(k)}(u,p) &:= A^{(k)}(u,p) + \langle \epsilon^{(k)}\nabla_u L(u^{(k)}) - \nabla_u A^{(k)}(u^{(k)}, p^{(k)}), u\rangle + \langle \epsilon^{(k)}\nabla_p L(u^{(k)}) - \nabla_p A^{(k)}(u^{(k)}, p^{(k)}), p\rangle \\
&\quad + \epsilon^{(k)} L^A(u,p) \\
&= A^{(k)}(u,p) + \langle \nabla_u L(u^{(k)}) - \nabla_u A^{(k)}(u^{(k)}, p^{(k)}), u\rangle - \langle \nabla_p A^{(k)}(u^{(k)}, p^{(k)}), p\rangle + L^A(u,p) \\
&= \langle u, (Q-D)u^{(k)}\rangle + \frac{1}{2}\langle u, Du\rangle - \frac{1}{2}\langle p, p\rangle + \langle (Q-D)u^{(k)} - \left((Q-D)u^{(k)} + Du^{(k)}\right), u\rangle \\
&\quad + \langle p, p\rangle + \frac{1}{2}\langle u, Du\rangle + p^T\Theta(u) \\
&= \langle u, Du\rangle + \langle (Q-2D)u^{(k)}, u\rangle + \langle p, \Theta(u)\rangle + \frac{1}{2}\langle p, p\rangle
\end{aligned}$$

Grâce au principe du problème auxiliaire, on a réussi à reformuler le problème sans termes croisés de type $u_i u_j$, de sorte qu'il est finalement décomposable. Le problème est même quasiment quadratique de type $\frac{1}{2}\langle u, Au\rangle - \langle b, u\rangle$ où on a $A := 2D$ ainsi que $b := -(Q-2D)u^{(k)}$ à l'itération $k+1$. Il faudra donc seulement penser à mettre b à jour à chaque itération de l'algorithme de décomposition.

On peut ainsi résoudre ce problème auxiliaire à l'aide par exemple de la décomposition par prix en utilisant l'algorithme suivant :

On considère la suite $(A^{(k)})_{k\in\mathbb{N}}$ et un pas $\rho > 0$

1. Initialisation ($k := 0$) : on se donne $(u^{(0)}, p^{(0)})$ et un test d'arrêt $\xi > 0$
2. Pour $k \geq 0$
   i) Résoudre $\inf_{u\in\mathbb{R}^N} \mathcal{L}^{(k)}(u, p^{(k)})$ par l'algorithme de décomposition par les prix (avec pour pas ρ), et obtenir ainsi $u^{(k+1)}$
   ii) Si $\|u^{(k+1)} - u^{(k)}\| + \|p^{(k+1)} - p^{(k)}\| \leq \xi$ alors on s'arrête,
   sinon $k := k+1$ et retourner en étape i).

On peut maintenant tester cet algorithme sur l'exemple jouet en distinguant deux cas :

Avec des corrélations négatives entre les actions

On prend d'abord la matrice de corrélation "complète" des actions, i.e. $Q := Cov(\{X_i\}_{i=1,\dots,N})$ (matrice qui contient des corrélations positives et négatives). Pour l'exemple jouet, on a :

$$Q \approx \begin{bmatrix} 1.17 & -0.33 & -0.06 & 0.19 \\ -0.33 & 10.83 & -6.18 & -0.43 \\ -0.06 & -6.18 & 7.97 & -0.22 \\ 0.19 & -0.43 & -0.22 & 1.93 \end{bmatrix}.$$

En essayant de choisir des hyperparamètres qui permettent une convergence rapide, on obtient le résultat suivant :

```
U_auxiliaire_corr_negatives = 0.36741  0.18296  0.20999  0.25366
valeur_optimale = 0.22046
nombre_iterations = 26
Les conditions KKT sont vérifiées au seuil d'erreur de 0.05
```

L'algorithme semble bien converger puisque les conditions KKT sont respectées (à un seuil d'erreur de 0.05) au bout des 26 itérations effectuées ici. Cela semble ainsi aussi indiquer que le principe auxiliaire a été correctement utilisé et que le code est bien implémenté. Concernant les résultats maintenant, l'ajout des termes de corrélations dans la matrice en plus des variances donne plus d'informations sur les actions et permet ainsi de proposer une répartition de l'argent à investir plus adaptée. Plus précisément, par rapport à la résolution qui ne prenait que les variances en compte, des termes de covariances négatives permettent de réduire encore plus la valeur de l'objectif qui est maintenant à 0.22 alors qu'elle est était de 0.34 lorsqu'on ne connaissait que les variances. Les répartitions ont évolué bien sûr pour utiliser au mieux ces termes de covariances négatives, ainsi on est passé de $u \approx (0.54, 0.06, 0.09, 0.35)$ à $u \approx (0.37, 0.18, 0.21, 0.25)$ ici. On peut par exemple remarquer que la covariance entre les actions 2 et 3 est la plus négative (-6.18), ce qui explique en partie pourquoi la proportion d'argent à investir dans ces deux actions a augmenté par rapport au cas avec les variances seules, dans lequel très peu d'argent devait être investi dans les actions 2 et 3.

Avec uniquement des corrélations positives

On simule enfin un cas un peu différent en considérant que toutes les corrélations entres les actions sont positives. On prend pour cela tout simplement $Q := abs(Cov(\{X_i\}_{i=1,\dots,N}))$, i.e. la même matrice de covariances mais avec tous les termes en valeur absolue. Pour l'exemple jouet, on a alors :

$$Q \approx \begin{bmatrix} 1.17 & 0.33 & 0.06 & 0.19 \\ 0.33 & 10.83 & 6.18 & 0.43 \\ 0.06 & 6.18 & 7.97 & 0.22 \\ 0.19 & 0.43 & 0.22 & 1.93 \end{bmatrix}.$$

En essayant encore une fois de choisir des hyperparamètres qui permettent une convergence rapide (voir le code Matlab), on obtient alors :

```
U_auxiliaire_corr_positives = 0.5799  -0.0059  0.1069  0.3635
valeur_optimale = 0.4156
nombre_iterations = 30
Les conditions KKT sont vérifiées au seuil d'erreur de 0.05
```

L'algorithme semble converger ici aussi et respectant les conditions KKT à un seuil d'erreur de 0.05. Cette fois-ci, la valeur de l'objectif a augmenté un peu par rapport au cas avec seulement les variances : on est passé de 0.34 à 0.42. La répartition de l'argent est aussi un peu différente par rapport au cas avec les variances

seules : on est passé de $u \approx (0.54, 0.06, 0.09, 0.35)$ à $u \approx (0.58, 0.0, 0.10, 0.36)$. Ces petits changements ne sont pas surprenant puisque la présence de termes de covariance uniquement positifs force l'augmentation de la valeur optimale, mais en essayant d'utiliser au mieux les covariances les plus faibles pour limiter cette augmentation.

3.2 Problème 2

Le problème 2 dans lequel on cherche à maximiser le gain tout en respectant un certain risque maximal D_e, peut s'écrire de la manière suivante :

$$\begin{aligned} \max_{u \in \mathbb{R}^N} \quad & \sum_{i=1}^{n} u_i e_i \\ \text{sous} \quad & \sum_{i=1}^{N} u_i = 1 && (1) \\ & \langle u, Qu \rangle \leq D_e && (2) \\ & u_i \geq 0, \forall i \in [\![1, N]\!] && (3) \end{aligned}$$

Comme pour le problème 1, on étudiera d'abord le cas où l'on ne connaît que les variances, i.e. avec Q diagonale, puis on considérera l'existence de terme de corrélations.

3.2.1 Sans corrélation entre les actions

Si on ne prend pas en compte de corrélation entre deux actions différentes, alors Q est diagonale et le problème est alors décomposable. Le petite difficulté est que la contrainte (2) n'est pas linéaire et qu'il faut donc réécrire un code pour faire une décomposition. Concrètement, en récrivant ce problème sous forme de minimisation et avec des contraintes naturelles :

$$\begin{aligned} \min_{u \in \mathbb{R}^N} \quad & - e^T u \\ \text{sous} \quad & \sum_{i=1}^{N} u_i = 1 && (1) \\ & \langle u, Qu \rangle \leq D_e && (2) \\ & - u_i \leq 0, \forall i \in [\![1, N]\!] && (3') \end{aligned}$$

on a le Lagrangien suivant :

$$\begin{aligned} L(u, \lambda, \mu_0, \mu) &:= -e^T u + \lambda(\sum_{i=1}^{N} u_i - 1) + \mu_0(\langle u, Qu \rangle - D_e) - \mu^T u \\ &= \sum_{i=1}^{N} \left(-e_i u_i + \lambda u_i + \mu_0 Q_{ii} u_i^2 - \mu_i u_i \right) - \lambda - \mu_0 D_e \end{aligned}$$

où $\lambda \in \mathbb{R}$, $\mu_0 \in \mathbb{R}_+$ et $\mu \in \mathbb{R}_+^N$ sont les multiplicateurs de Lagrange associés aux contraintes (1), (2) et $(3')$.

Ce problème est bien décomposable et on veut utiliser ici l'algorithme de décomposition par prix dans lequel

il faut résoudre les sous-problèmes suivant (à l'itération k) :

$$\min_{u_i \in \mathbb{R}} \; L_i(u_i, \lambda^{(k)}, \mu_0^{(k)}, \mu_i^{(k)}) := -e_i u_i + \lambda^{(k)} u_i + \mu_0^{(k)} Q_{ii} u_i^2 - \mu_i^{(k)} u_i \quad \forall i = 1, ..., N$$

La solution de ces sous problèmes est directement obtenue en annulant les gradients des L_i :

$$\begin{aligned} \nabla_{u_i} L_i(u_i, \lambda^{(k)}, \mu_0^{(k)}, \mu_i^{(k)}) = 0 \;\; &\Rightarrow \;\; -e_i + \lambda^{(k)} + 2\mu_0^{(k)} Q_{ii} u_i - \mu_i^{(k)} = 0 \\ &\Rightarrow \;\; u_i^{(k+1)} = (2\mu_0^{(k)} Q_{ii})^{-1}(e_i - \lambda^{(k)} + \mu_i^{(k)}) \end{aligned}$$

Enfin, l'étape de coordination se fera de la manière suivante, avec un pas $\rho > 0$:

$$(\lambda^{(k+1)}, \mu_0^{(k+1)}, \mu^{(k+1)}) := \Pi_{\mathbb{R}\times\mathbb{R}_+\times\mathbb{R}_+^N} \left((\lambda^{(k)}, \mu_0^{(k)}, \mu^{(k)}) + \rho\big(\sum_{i=1}^{N} u_i^{(k+1)} - 1, \langle u^{(k+1)}, Q u^{(k+1)} \rangle - D_e, -u^{(k+1)} \big) \right)$$

On a implémenté cette décomposition par prix sur Matlab et pour l'exemple jouet on obtient les résultats suivant (on rappelle que ici Q est diagonale) :

```
U_decomp_prix_sans_corr = 0.12196  0.14407  0.23695  0.49705
valeur_optimale = 11.746
nombre_iteration = 210
Les conditions KKT sont vérifiées au seuil d'erreur de 0.001
```

L'algorithme converge bien et respecte les conditions KKT au seuil d'erreur de 0.001. On obtient la solution optimale $u = (u_1, u_2, u_3, u_4) \approx (0.12, 0.14, 0.24, 0.50)$. On conseillerait donc à la personne qui investi son argent, d'en mettre environ 12% dans la première, environ 14% dans la deuxième, environ 24% dans la troisième et environ 50% dans la quatrième pour espérer obtenir un gain maximal de 11.75 (gain qui est d'ailleurs un peu supérieur au gain minimal $R_e = 11.72$ autorisé dans le problème 1).

3.2.2 Avec corrélations entre les actions

On souhaite maintenant considérer la présence de corrélations entre les actions. La contrainte (2) n'est ainsi plus décomposable. On propose donc de la linéariser en utilisant un développement de Taylor d'ordre 1. Soit $f(u) := \langle u, Qu \rangle$, on a alors $f(u) = f(u_0) + \langle u - u_0, \nabla f(u_0) \rangle + \mathcal{O}(\|u - u_0\|)$. On peut donc faire l'approximation suivante :

$$\langle u, Qu \rangle \approx \langle u_0, Qu_0 \rangle + \langle u - u_0, Qu_0 \rangle$$

en choisissant un $u_0 \in \mathbb{R}^N$ très proche de u.

À l'itération k de l'algorithme de résolution de ce problème, on propose ainsi d'utiliser en tant que u_0 le $u^{(k)}$ déterminé à l'étape précédente. De plus, afin de s'assurer que le $u^{(k+1)}$ déterminé à l'itération k ne soit pas trop loin de $u^{(k)}$, on décide d'ajouter le terme de pénalisation $-\alpha\|u - u^{(k)}\|$ dans l'objectif, avec $\alpha > 0$. À l'itération k le problème devient ainsi :

$$
\begin{aligned}
\max_{u\in\mathbb{R}^N} \quad & \sum_{i=1}^{n} u_i e_i - \alpha\|u-u^{(k)}\| \\
\text{sous} \quad & \sum_{i=1}^{N} u_i = 1 && (1)\\
& \langle u^{(k)}, Qu^{(k)}\rangle + \langle u-u^{(k)}, Qu^{(k)}\rangle \le D_e && (2')\\
& u_i \ge 0, \forall i \in [\![1,N]\!] && (3)
\end{aligned}
$$

que l'on peut réécrire sous la forme du problème de minimisation suivant :

$$
\begin{aligned}
\min_{u\in\mathbb{R}^N} \quad & \alpha\langle u,u\rangle - \langle e+2\alpha u^{(k)}, u\rangle && (4)\\
\text{sous} \quad & \sum_{i=1}^{N} u_i = 1 && (1)\\
& \langle u, Qu^{(k)}\rangle \le D_e && (2')\\
& -u_i \le 0, \forall i \in [\![1,N]\!] && (3')
\end{aligned}
$$

car $\langle u^{(k)}, Qu^{(k)}\rangle + \langle u-u^{(k)}, Qu^{(k)}\rangle = \langle u, Qu^{(k)}\rangle$, $\|u-u^{(k)}\| = \langle u,u\rangle + \langle u^{(k)},u^{(k)}\rangle - 2\langle u,u^{(k)}\rangle$.

Cette dernière formulation est bien décomposable de sorte que l'on pourra par exemple utiliser un algorithme de décomposition par prix pour résoudre le problème à chaque itération k. À k fixé, le problème est même quadratique de type :

$$
\begin{aligned}
\min_{u\in\mathbb{R}^N} \quad & \frac{1}{2}\langle u, Au\rangle - \langle b, u\rangle \\
\text{sous} \quad & C_{\text{e}} u = d_e && \text{(contraintes d'}\mathbf{\text{é}}\text{galité)}\\
& C_{\text{i}} u \le d_i && \text{(contraintes d'}\mathbf{\text{i}}\text{négalité)}
\end{aligned}
$$

où $A := 2\alpha I_N$, $b := e + 2\alpha u^{(k)}$, $C_e := (1,...,1) \in \mathbb{R}^N$, $C_i := (Qu^{(k)}, -u)^T$, $d_e := 1$ et $d_i := (D_e, 0)^T$

(c'est précisément avec ces écritures que le code est implémenté dans Matlab)

On utilise donc la reformulation $(1)(2')(3')(4)$ et la décomposition par prix pour résoudre ce problème. L'algorithme utilisé est le suivant :

On considère une constante $\alpha > 0$ et un pas $\rho > 0$

1. Initialisation ($k := 0$) : on se donne $(u^{(0)}, \lambda^{(0)}, \mu_0^{(0)}, \mu^{(0)})$ et un test d'arrêt $\xi > 0$
2. Pour $k \ge 0$
   i) Résoudre la reformulation $(1)(2')(3')(4)$ par l'algorithme de décomposition par les prix (avec pour pas ρ), et obtenir ainsi $u^{(k+1)}$
   ii) Si $\|u^{(k+1)} - u^{(k)}\| + \|(\lambda^{(k+1)}, \mu_0^{(k+1)}, \mu^{(k+1)}) - (\lambda^{(k)}, \mu_0^{(k)}, \mu^{(k)})\| \le \xi$ alors on s'arrête, sinon $k := k+1$ et retourner en étape i).

Comme pour le problème 1, on veut distinguer deux cas : avec des corrélations négatives, et avec des corrélations uniquement positives.

Avec des corrélations négatives entre les actions

Comme pour le premier problème, on prend d'abord la matrice de corrélation "complète" des actions, i.e. $Q := Cov(\{X_i\}_{i=1,...,N})$ (matrice qui contient des corrélations positives et négatives). Pour l'exemple jouet, on a :

$$Q \approx \begin{bmatrix} 1.17 & -0.33 & -0.06 & 0.19 \\ -0.33 & 10.83 & -6.18 & -0.43 \\ -0.06 & -6.18 & 7.97 & -0.22 \\ 0.19 & -0.43 & -0.22 & 1.93 \end{bmatrix}.$$

En essayant de choisir des hyperparamètres qui permettent une convergence rapide (voir le code Matlab), on obtient le résultat suivant :

```
U_reformulation_corr_negatives = -0.0002  0.3814  0.4971  0.1214
valeur_optimale = 12.192
nombre_iteration = 12
Les conditions KKT sont vérifiées au seuil d'erreur de 0.001
```

Les conditions KKT sont vérifiées au seuil d'erreur de 0.001 et l'algorithme converge bien ce qui semble indiquer que la reformulation du problème est bonne et que le code est correctement implémenté. La connaissance supplémentaire des corrélations entre les actions modifie bien entendu la répartition optimale de l'argent que l'on peut faire. Ainsi, on a obtenu la solution optimale $u \approx (0.0, 0.38, 0.50, 0.12)$ avec pour valeur optimale 12.19 contre $u \approx (0.12, 0.14, 0.24, 0.50)$ et 11.75 dans le cas avec les variances seules. En fait, la connaissance des covariances permet de mieux estimer le risque et donc même d'augmenter encore le gain maximal ici, car il existe des covariances négatives qui permettent d'assurer le niveau de risque D_e tout en choisissant des gains plus élevés.

Avec uniquement des corrélations positives entre les actions

On simule enfin le cas un peu différent où l'on considère que toutes les corrélations entres les actions sont positives. On prend pour cela $Q := abs(Cov(\{X_i\}_{i=1,...,N}))$, i.e. la même matrice de covariances mais avec tous les termes en valeur absolue. Pour l'exemple jouet, on a alors :

$$Q \approx \begin{bmatrix} 1.17 & 0.33 & 0.06 & 0.19 \\ 0.33 & 10.83 & 6.18 & 0.43 \\ 0.06 & 6.18 & 7.97 & 0.22 \\ 0.19 & 0.43 & 0.22 & 1.93 \end{bmatrix}.$$

En essayant ici aussi de choisir des hyperparamètres qui permettent une convergence rapide (voir le code Matlab), on obtient :

```
U_reformulation_corr_positives = 0.2269  0.0005  0.235  0.5378
valeur_optimale = 11.586
nombre_iteration = 18
Les conditions KKT sont vérifiées au seuil d'erreur de 0.001
```

L'algorithme semble bien converger puisque les conditions KKT sont respectées (au seuil d'erreur de 0.001). On obtient cette fois-ci la solution optimale $u \approx (0.23, 0.0, 0.23, 0.54)$ avec pour valeur optimale 12.19 (alors qu'on avait $u \approx (0.12, 0.14, 0.24, 0.50)$ et 11.75 dans le cas avec les variances seules). Concrètement, le fait que toutes les covariances soient positives force à faire plus attention sur la répartition de l'argent pour respecter un risque maximal $D_e = 1.17$. Le gain maximal se voit lui ainsi diminué par rapport au cas où l'on ne connaissait que les variances.

4 Exercice 4

4.1 Description du problème

Nous nous intéressons à la gestion de smart grids. L'objectif est d'assurer l'équilibre entre les puissances injectées dans le réseau et celles qui sont consommées en convenant un prix pour la puissance. Si on dispose N agents dont N_p agents producteurs, chacun consommant ou produisant une puissance p_i sur le réseau et possédant une fonction de coût f_i, le problème global se ramène alors à :

$$\min_{P \in K^{ad}} \sum_{i=1}^{N} f_i(p_i)$$

sous les contraintes

$$\sum_{i=1}^{N} p_i = 0$$

avec $P = (p_1, \dots, p_N)^T = (P_p, P_c)^T$, $K^{ad} = \Pi_{i=1}^{N} K_i^{ad}$ et $f_i(p_i) = a_i(p_i - p_i^0)^2 + b_i$, où

- p_i la puissance électrique injectée sur le réseau par l'agent i ($p_i < 0$ pour un agent consommateur)
- P_p est le vecteur des puissances pour les agents producteurs, P_c pour les agents consommateur
- p_i^0 la puissance idéale pour chaque agent i
- $a_i > 0$ un coefficient de pénalité de la différence entre la puissance réelle et la puissance idéale pour l'agent i
- b_i le coût associé à la puissance idéale de l'agent i
- $K_i^{ad} = [0, P_i^{max}]$ pour un agent producteur et $K_i^{ad} =]-\infty, 0]$ pour un agent consommateur.

La fonction de coût pour l'agent i peut être développée comme $f_i(p_i) = a_i p_i^2 - 2a_i p_i^0 p_i + a_i (p_i^0)^2 + b_i$, le problème global peut donc être écrit sous forme matricielle :

$$\min_{P \in \mathbb{R}^N} \frac{1}{2}\langle AP, P\rangle - \langle B, P\rangle + cste$$

sous les contraintes

$$C_{\text{éga}} P = d_{\text{éga}} \qquad \text{(contraintes d'**éga**lité)}$$
$$C_{\text{iné}} P \leq d_{\text{iné}} \qquad \text{(contraintes d'**iné**galité)}$$

où $A := \begin{pmatrix} 2a_1 & 0 & & 0 \\ 0 & \ddots & \ddots & \\ & \ddots & \ddots & 0 \\ 0 & & 0 & 2a_N \end{pmatrix} \in \mathbb{R}^{N \times N}$, $B := \begin{pmatrix} 2a_1 p_1^0 \\ \vdots \\ 2a_N p_N^0 \end{pmatrix} \in \mathbb{R}^N$, $C_{\text{éga}} := \big(1, ..., 1\big) \in \mathbb{R}^N$,

$$C_{\text{iné}} := \left(\begin{array}{c|ccc} & I_N & & \\ \hline & 0 & \dots & 0 \\ -I_{N_p} & \vdots & \ddots & \vdots \\ & 0 & \dots & 0 \end{array}\right) \in \mathbb{R}^{(N+N_p)\times N}, \quad d_{\text{éga}} := 0, \quad d_{\text{iné}} := \begin{pmatrix} p_1^{max} \\ \vdots \\ p_N^{max} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^{N+N_p}$$

et $cste := (a_1(p_1^0)^2 + b_1) + \cdots + (a_N(p_N^0)^2 + b_N)$.

On se trouve de nouveau dans un cas où la fonction objectif est quadratique, les contraintes sont affines, donc le problème admet une seule solution.

4.2 Décomposition

Les contraintes peuvent être décomposées comme suit :

$$\Theta^{\text{éga}}(P) = \sum_{i=1}^{N} \theta_i^{\text{éga}}(p_i) = 0 \text{ avec } \theta_i^{\text{éga}}(p_i) = p_i, \ \forall i = 1, \dots, N$$

$$\Theta^{\text{iné}}(P) = \sum_{i=1}^{N} \theta_i^{\text{iné}}(p_i) \leq d_{\text{iné}} \text{ avec } \theta_i^{\text{iné}}(p_i) = C_{\text{iné}}(:, i) \times p_i, \ \forall i = 1, \dots, N$$

4.2.1 Décomposition par prix

À l'itération k, soient les prix $\lambda^{(k)}$ et $\mu^{(k)}$ connus :

(I) Décomposition : $\forall i = 1, \dots, N$, on résout

$$\inf_{p_i \in \mathbb{R}} f_i(p_i) + \langle \lambda^{(k)}, \theta_i^{\text{éga}}(p_i) \rangle + \langle \mu^{(k)}, \theta_i^{\text{iné}}(p_i) \rangle$$

ce qui nous donne les solutions $p_i^{(k+1)}$.

(II) Coordination : Avec $\rho_k > 0$,

$$\lambda^{(k+1)} = \lambda^{(k)} + \rho_k \sum_{i=1}^{N} \theta_i^{\text{éga}}(p_i^{(k+1)})$$

$$\mu^{(k+1)} = \max\left\{0, \mu^{(k)} + \rho_k(\sum_{i=1}^{N} \theta_i^{\text{iné}}(p_i^{(k+1)}) - d_{\text{iné}})\right\}$$

4.2.2 Décomposition par ressources

À l'itération k, soient les allocations $(w_i^{\text{éga},(k)})_{i=1,\dots,N}$ telles que $\sum_{i=1}^{N} w_i^{\text{éga},(k)} = 0$ et $w_i^{\text{iné},(k)}{}_{i=1,\dots,N}$ telles que $\sum_{i=1}^{N} w_i^{\text{iné},(k)} = d_{\text{iné}}$ connus :

(I) Décomposition : $\forall i = 1, \ldots, N$, on résout

$$\inf_{p_i \in \mathbb{R}} f_i(p_i)$$

sous les contraintes

$$\theta_i^{\text{éga}}(p_i) = w_i^{\text{éga},(k)}$$
$$\theta_i^{\text{iné}}(p_i) \leq w_i^{\text{iné},(k)}$$

ce qui nous donne les solutions $p_i^{(k+1)}$, $\lambda_i^{(k)}$ et $\mu_i^{(k)}$.

(II) Coordination : Avec $\rho_k > 0$,

$$w_i^{\text{éga},(k+1)} = w_i^{\text{éga},(k)} + \rho_k(\lambda_i^{(k)} - \frac{1}{N}\sum_{i=1}^{N} \lambda_i^{(k)})$$

$$w_i^{\text{iné},(k+1)} = w_i^{\text{iné},(k)} + \rho_k(\mu_i^{(k)} - \frac{1}{N}\sum_{i=1}^{N} \mu_i^{(k)})$$

Notons que pour l'implémentation de cet algorithme, contrairement à ce qu'on a dit dans l'avant propos au début du rapport, on ne stocke pas les solutions à chaque itération pour accélérer la vitesse.

4.3 Implémentations

On étudie deux scénarios pour tester les algorithmes :
- La puissance produite est conséquente et une partie de la puissance vient d'énergie renouvelable
- La puissance produite est diminuée car il n'y a pas de vents ($P^{max} = 0$ pour les éoliennes)

La variable duale λ correspond au prix en M€/MW.

Pour cette implémentation, on s'intéresse à l'étude de cas suivante :

Agent	Nombre	Capacité/Consommation (MW)	P_0 (MW)	a	b
Centrale à charbon	1	+10	8.5	10	1000
Éolienne	5	+2	1.9	1	10
Barrage	1	+16	3	0.1	100
Panneau photovoltaïque	2	+2	1.9	0.9	11
Data Center	1	-10	-10	5	20
Logement	7500	-1 kW	-7.5	1	10
Usine	1	-9	-9	5	8
Trameway	2	-0.12	-0.12	6	9
Hôpital	1	-0.2	-0.2	200	30

4.3.1 Scénario 1 : Puissance maximale

Par la décomposition par prix, on obtient :

Agent	Puissance fournie (MW)	P_0 (MW)
Centrale à charbon	8.5122	8.50
Éolienne	2.0002	1.90
Barrage	4.2152	3.00
Panneau photovoltaïque	2.0002	1.90
Data Center	-9.9757	-10
Logement $\times 7500$	-7.3785	-7.5
Usine	-8.9757	-9
Trameway	-0.099747	-0.12
Hôpital	-0.19939	-0.20

Le prix est de -0.24304 M€/MW. Le coût est 1258.2431 M€. De plus $\sum_{i=1}^{N} p_i \approx 0$.

Par la décomposition par ressources, en prenant $\rho = 0.03$ et $\rho_{uzawa} = 0.1$, on obtient :

Agent	Puissance fournie (MW)	P_0 (MW)
Centrale à charbon	8.5122	8.50
Éolienne	2	1.90
Barrage	4.2164	3.00
Panneau photovoltaïque 1	2	1.90
Data Center	-9.9756	-10
Logement $\times 7500$	-7.3783	-7.5
Usine	-8.9756	-9
Trameway	-0.09969	-0.12
Hôpital	-0.19924	-0.20

Le prix est de -0.24331 M€/MW. Le coût est 1258.2433 M€. De plus $\sum_{i=1}^{N} p_i \approx 0$.

Les deux algorithmes donnent des solutions assez proches, et les solutions sont aussi proches des puissances idéales. Néanmoins, la décomposition par ressources prend plus de temps que la décomposition par prix, on remarque également que le choix des valeurs du pas est moins évident pour la décomposition par ressources.

4.3.2 Scénario 2 : Diminution des énergies renouvelables

Par la décomposition par prix, on obtient :

Agent	Puissance fournie (MW)	P_0 (MW)
Centrale à charbon	8.5973	8.50
Éolienne	0.00002	1.90
Barrage	12.734	3.00
Panneau photovoltaïque	2.0002	1.90
Data Center	-9.8053	-10
Logement $\times 7500$	-6.5266	-7.5
Usine	-8.8053	-9
Trameway	0.00061	-0.12
Hôpital	-0.19513	-0.20

Le prix est -1.9467 M€/MW, qui est plus important et reflète un rapport offre/demande plus faible. Le coût est de 1287.1425 M€, ce qui est plus important car les solutions sont plus éloignées des puissances idéales. On a toujours $\sum_{i=1}^{N} p_i \approx 0$.

La puissance fournie par les éoliennes est bien proche de 0, néanmoins on remarque que la puissance consommée par les trameways est quasiment nulle, les trameways sont donc hors-service dans ce cas là. On voit également que la puissance produite par les barrages sont largement supérieure à P_0. Le réseau n'est pas bien dimensionné.

Par la décomposition par ressources, en prenant $\rho = 0.002$ et $\rho_{uzawa} = 0.05$, on obtient :

Agent	Puissance fournie (MW)	P_0 (MW)
Centrale à charbon	8.5976	8.50
Éolienne	-0.028639	1.90
Barrage	13.152	3.00
Panneau photovoltaïque	1.9071	1.90
Data Center	-9.8069	-10
Logement $\times 7500$	-6.5331	-7.5
Usine	-8.8069	-9
Trameway	-0.024159	-0.12
Hôpital	-0.19159	-0.20

Le prix est -1.9317 M€/MW, le coût est de 1288.4308 M€, ce qui est plus important car les solutions sont plus éloignées des puissances idéales. On a $\sum_{i=1}^{N} p_i \approx 0$.

Dans cette solution obtenue par la décomposition par ressources, les résultats semblent moins précis qu'avec la décomposition par prix. En particulier, les éoliennes semblent consommer un tout petit peu de puissance alors que celle-ci devrait être nulle. On retrouve quand même l'idée que les trameways consomment très peu de puissance mais les valeurs restent plus grandes que dans la décomposition par prix. La solution trouvée n'est donc pas complètement satisfaisante car elle ne respecte pas réellement les contraintes (notre algorithme s'est arrêté au nombre maximal d'itérations de 5000 et non nécessairement à la convergence). On en conclut donc que la décomposition par ressources est moins flexible. Cela est peut être du au fait qu'on considère les contraintes qui définissent le domaine admissible comme les contraintes explicites, on a donc ajouté $N_p + N$ contraintes d'inégalités. Si on les avait considérées comme des contraintes implicites, on aurait pu faire une projection sur le domaine admissible des solutions, ce qui aurait peut-être permit d'aller plus vite.