Travaux Pratiques d'analyse Hilbertienne

Moteur de recherche sur internet : un algorithme de *page ranking*

1 Introduction

L'objectif de ce TP est d'illustrer l'utilisation d'outils d'algèbre linéaire par les moteurs de recherche sur Internet pour le classement de pages Web. Pour cela, il vous est fourni un script MATLAB `pageranking` simulant le classement des pages Web. Celle-ci vous permet de définir l'internet sous forme d'un graphe orienté (pages et liens), puis de résoudre le problème de classement tel que décrit dans la suite de l'énoncé.

Le TP consiste à implanter toutes les fonctions nécessaires à la résolution de ce problème. Il vous est également demandé de rédiger un rapport (maximum 5 pages) synthétisant le travail réalisé et les résultats obtenus. **Les codes sources, ainsi que ce rapport, devront être envoyés à l'enseignant en charge de votre groupe de TP pour le vendredi 29 janvier 19h00 au plus tard.**

2 Modélisation mathématique du problème de classement de pages Web

Dans toute la suite de l'énoncé, nous noterons $\mathbb{N}_n$ l'ensemble des entiers naturels de 1 à n.

Soit n le nombre de pages Web et soit $i \in \mathbb{N}_n$ une page Web particulière. Posons I_i l'ensemble des pages Web ayant un lien vers la page i et O_i l'ensemble des pages Web vers lesquelles la page i a un lien. Le nombre de liens sortant de la page i est noté $N_i = |O_i|$.

Un moyen de mesurer l'importance relative de la page i dans le graphe de l'Internet consiste à additionner l'importance relative des pages Web ayant un lien vers i. Afin de réduire l'impact de possibles manipulations (p.e. création de nombreuses pages "artificielles" pointant vers la page i), une pondération tenant compte du nombre de liens sortant de chaque pages $j \in I_i$ est incluse lors de la somme. Notons $r \in \mathbb{R}^n$ le vecteur des importances relatives des pages Web, le problème s'écrit alors :

$$\forall i \in \mathbb{N}_n, \quad r_i = \sum_{j \in I_i} \frac{r_j}{N_j} \tag{1}$$

Un problème de recherche de vecteurs propres

Définissons Q de la manière suivante :

$$Q_{ij} = \begin{cases} 1/N_j & \text{si } j \in I_i \\ 0 & \text{sinon} \end{cases}$$

La ième ligne de Q contient les poids des pages Web ayant un lien vers la page i (I_i) tandis que la jème colonne a des valeurs égales à $1/N_j$ pour les pages vers lesquelles la page j a un lien (O_j).

Le problème (1) de classement des pages Web s'écrit alors comme un problème de recherche de vecteur propre :

$$r = Qr \tag{2}$$

Travail à réaliser

1. Ecrire la fonction `matrix_representation` implantant le calcul de la matrice Q. Cette fonction prend pour entrée le nombre de pages Web et un tableau contenant les liens entre ces pages.

Existence d'une solution du problème de classement de pages Web

L'existence de solution du problème (2) n'est pas garantie, la valeur 1 n'étant pas nécessairement une valeur propre de la matrice Q. Une approche permettant de résoudre le problème de classement consiste à modifier cette matrice pour travailler avec une matrice P ayant, par construction, la valeur 1 comme valeur propre. Une telle matrice peut être construite en réinterpretant d'un point de vue probabiliste (marche aléatoire) le problème.

Un modèle pour la navigation sur le Web consiste à faire l'hypothèse que l'utilisateur passe d'une page i à la page j en choisissant aléatoirement et de manière équiprobable $j \in O_i$. Afin de ne pas rester bloqué sur une page faute de lien sortant, il faut s'assurer que $\forall i \in \mathbb{N}_n, O_i \neq \emptyset$. Ceci est équivalent à ce que les colonnes de Q ne soient pas nulles. P va donc être construite en remplissant les colonnes nulles de Q par une valeur constante. Une fois sur une page n'ayant pas de lien sortant, l'utilisateur visite ainsi n'importe quelle autre page en supposant de nouveau l'équiprobabilité entre les pages.

Soit $d \in \mathbb{R}^n$ tel que

$$d_j = \begin{cases} 1 & \text{si } N_j = 0 \quad (O_j = \emptyset) \\ 0 & \text{sinon} \end{cases}$$

P s'écrit alors :

$$P = Q + \frac{1}{n} e d^T$$

avec $e \in \mathbb{R}^n$ tel que $\forall i \in \mathbb{N}_n, e_i = 1$. Par construction, P^T est une matrice stochastique : $\forall (i,j) \in \mathbb{N}_n^2, P_{ij} \geq 0$ et $\forall j \in \mathbb{N}_n \sum_{i=1}^{n} P_{ij} = 1$. Il en résulte que 1 est une valeur propre de P, ce qui garantit l'existence d'une solution au problème

$$r = Pr. \tag{3}$$

De plus, la ième composante r_i de r peut être interprétée comme la probablilité qu'après un très grand nombre d'étapes, l'utilisateur se retrouve sur la page i.

Travail à réaliser

1. Vérifier que $e^T P = e^T$. Que pouvez-vous en conclure ?
2. Construire un graphe de l'Internet pour lequel le problème de classement (2) n'admette pas de solution. A l'aide de la fonction MATLAB *eig*, vérifier que la valeur 1 n'est pas valeur propre de la matrice Q.
3. Ecrire la fonction `columnstochastic_matrix` calculant la matrices P
4. Vérifier que la valeur 1 est bien valeur propre de la matrice P associée au graphe construit en 2.

Unicité d'une solution du problème de classement de pages Web

L'unicité du vecteur r solution de (3) n'est pas garantie. Pour nous assurer que l'algorithme de classement obtienne toujours le même résultat lorsque appliqué plusieurs fois sur les mêmes données Web, nous commençons par introduire la notion de réductibilité d'une matrice.

Définition *Soit $A \in \mathcal{M}_n(\mathbb{R})$. A est dite réductible s'il existe une matrice de permutation P telle que*

$$PAP^T = \begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix}$$

avec X et Z carrées. Elle est dite irréductible dans le cas contraire.

Le graphe orienté associé à une matrice irréductible est fortement connecté. Dans notre cas, cela veut dire que pour toutes pages i et j, il existe un chemin de i vers j. Le théorème suivant nous assure, sous certaines hypothèses, l'unicité de la solution du problème de classement.

Théorème *Soit $A \in \mathcal{M}_n(\mathbb{R})$ une matrice irréductible telle que A^T est stochastique. La valeur propre dominante λ_1 est 1 et il existe un unique vecteur propre r associé satisfaisant $\forall i \in \mathbb{N}_n, r_i > 0$ et $\|r\|_1 = 1$.*
De plus, si $\forall (i,j) \in \mathbb{N}_n^2, A_{ij} > 0$, alors $|\lambda_k| < 1 \quad \forall k = 2, \ldots, n$.

Considérant la taille de l'Internet, il est certain que P est réductible. Ceci veut dire qu'il existe des sous-graphes de l'Internet dans lesquels un utilisateur peut se retrouver coincé. Une solution consiste à rajouter artificiellement à chacune des pages un lien vers toutes les autres pages du Web. D'un point de vue matricielle, ceci sécrit :

$$A_{\alpha,v} = \alpha P + (1-\alpha) v e^T$$

avec $0 < \alpha < 1$ et v satisfaisant $\forall i \in \mathbb{N}_n, v_i > 0$ et $\|v\|_1 = 1$. Dans notre modèle probabiliste de visite du Web, ceci peut être interprété de la manière suivante : sur chaque page Web, l'utilisateur est "téléporté" aléatoirement vers une autre page avec la probabilité $1-\alpha$. v est appelé vecteur de personnalisation et définit les probabilités d'attérir sur chacune des pages lors d'une téléportation. Il permet entre autre d'orienter le classement vers certaines pages Web.

Il peut être montré que $A_{\alpha,v}$ est irréductible et $A_{\alpha,v}^T$ est stochastique. Il en résulte que le problème

$$r = A_{\alpha,v} r. \tag{4}$$

admet une unique solution r satisfaisant $\forall i \in \mathbb{N}_n, r_i > 0$ et $\|r\|_1 = 1$.

Travail à réaliser

1. Ecrire la fonction `irreducible_matrix` calculant la matrice $A_{\alpha,v}$
2. Comparer Q, P et $A_{\alpha,v}$ à l'aide de la fonction MATLAB `spy`. Que constatez-vous ? Quelles conséquences sur le calcul du vecteur propre pouvez-vous anticiper ?

3 Résolution numérique

Le problème consiste donc à trouver le vecteur propre de la matrice $A_{\alpha,v}$ associé à la valeur propre 1. Considérant la très grande dimension de la matrice, et la nature même du problème (un seul vecteur propre recherché), il n'est pas envisageable d'utiliser un solveur direct tel que la fonction MATLAB `eig`. On utilisera plutôt un solveur itératif, tel que la méthode de la puissance itérée qui permet de calculer la valeur propre de plus fort module d'une matrice, ainsi qu'un vecteur propre associé. Dans le cadre du problème, cette méthode peut s'écrire :

Méthode de la puissance itérée pour le problème de classement de pages Web

Données : $A_{\alpha,v}$, r_0 première approximation de la solution cherchée, $\epsilon > 0$ précision demandée, k_{max} nombre maximal d'itérations.

Sortie : une approximation d'un vecteur propre associé à la plus grande valeur propre en module (1 dans notre cas).

Convergence : $\|A_{\alpha,v} r_k - r_k\|_1 < \epsilon \|r_k\|_1$ ou $k > k_{max}$.

1. Tant que le test de convergence est non satisfait :

a. Calculer $q_k = A_{\alpha,v} r_k$.

b. Normaliser : $r_k = q_k / \|q_k\|_1$.

c. $k = k + 1$.

2. Retourner r_k.

Travail à réaliser

1. Ecrire la fonction `power_matrix_dense` implantant la méthode de la puissance itérée appliquée au problème $r = A_{\alpha,v} r$.
2. Résoudre $r = A_{\alpha,v} r$ lorsque les paramètres α et v varient.
3. L'algorithme tel qu'il est proposé conduit à la construction et la manipulation d'une matrice dense $A_{\alpha,v}$ et ne permet plus d'exploiter la structure *creuse* (pleine de 0) de la matrice Q. Ceci peut conduire à une augmentation significative des coûts et temps de calcul lors des produits matrice-vecteur (comparativement à l'utilisation

d'outils d'algèbre linéaire creuse). En exploitant que $\|A_{\alpha,v}r\|_1 = e^T A_{\alpha,v}r = 1$ si $\|r\|_1 = e^T r = 1$, proposer un algorithme qui réalise le produit $A_{\alpha,v}r$ sans jamais calculer explicitement $A_{\alpha,v}$, P et d.

4. Ecrire une nouvelle fonction `power_matrix_sparse` implantant une méthode de la puissance itérée permettant l'exploitation de la structure *creuse* de la matrice Q.