Université Paul Sabatier Année 2020-2021

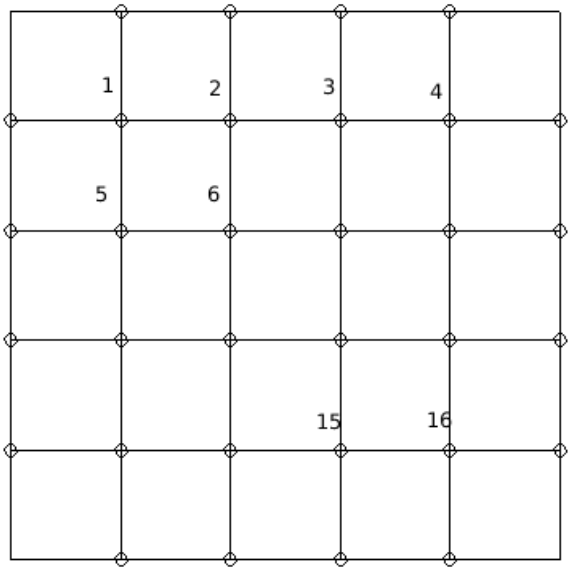
Licence L2 CUPGE

Module d'ouverture « Analyse de données et grands systèmes linéaires »

Mise en œuvre d'une méthode de projection pour la résolution d'un système linéaire de grande dimension

1. **Rappel du problème considéré**

   On considère un réseau rectangulaire régulier constitué de résistances électriques (la figure montre un réseau carré). Deux nœuds voisins du réseau (représentés par des ∘ sur la figure) sont reliés par un fil conducteur de résistance R avec un courant d'intensité I qui vérifie la loi d'Ohm $\delta U = RI$, où δU est la différence de potentiel électrique aux deux nœuds. Aux bords du réseau la valeur du potentiel est fixée (conditions aux bords fixées). Sur la figure on a ainsi 4 valeurs de potentiel fixées sur chaque côté du carré.



   Le problème que l'on se pose est celui de calculer la valeur du potentiel aux nœuds situés à l'intérieur du réseau (numérotés de 1 à 16 sur la figure) à partir des valeurs du potentiel fixées sur les 4 côtés du carré. Cependant, la simple considération de la loi d'Ohm ne permet pas d'obtenir une solution unique à ce problème. Il faut considérer en outre la loi de Kirchhoff qui impose qu'en chaque nœud la somme de tous les courants qui arrivent au nœud soit égale à la somme des courants qui en sortent (loi de conservation de la charge électrique).

La conjonction des deux lois (loi d'Ohm et loi de Kirchhoff) permet de définir un système linéaire dans lequel l'inconnue est le vecteur u ayant pour coordonnées le potentiel aux nœuds intérieurs du réseau (sur la figure le vecteur u est de dimension 16).
En considérant des résistances constantes égales à 1, la matrice du système K (correspondant à la figure) est obtenue par la produit

$$K = (I_4 \otimes D^T \, D^T \otimes I_4) \begin{pmatrix} I_4 \otimes D \\ D \otimes I_4 \end{pmatrix} = (I_4 \otimes D^T D) + (D^T D \otimes I_4)$$

où

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

est la matrice des différences (prise ici de dimension 5×4 correspondant à la figure), I_4 la matrice identité 4×4 et $\otimes$ est le produit de Kronecker de deux matrices.
En explicitant la somme $(I_4 \otimes D^T D) + (D^T D \otimes I_4)$ on constate que la matrice K est une matrice tridiagonale par blocs de la forme

$$K = \begin{pmatrix} T_4 & -I_4 & 0 & 0 \\ -I_4 & T_4 & -I_4 & 0 \\ 0 & -I_4 & T_4 & -I_4 \\ 0 & 0 & -I_4 & T_4 \end{pmatrix}$$

avec

$$T_4 = \begin{pmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{pmatrix}$$

La matrice K est de plus définie positive. Le membre de droite du système (vecteur noté f) correspond aux données du problème c.-à-d. les valeurs du potentiel sur les 4 côtés du carré. En respectant l'ordre qui a été choisi dans la figure pour le vecteur u, le vecteur f est donné par

$$f = (e_1 \otimes h) + (g \otimes e_1) + (e_4 \otimes b) + (d \otimes e_4)$$

où $e_1 = (1, 0, 0, 0)^T$, $e_4 = (0, 0, 0, 1)^T$ (e_1 et e_4 sont le premier et le dernier vecteur de la base canonique de $\mathbb{R}^4$), h le vecteur colonne de $\mathbb{R}^4$ qui contient les valeurs du potentiel donné sur le bord haut du carré, g pour le bord gauche, b pour le bord bas et d pour le bord droit. Le vecteur f a la dimension du vecteur inconnu u c.-à-d. 16.

Le système linéaire que l'on considère s'écrit donc $Ku = f$. Pour réaliser des tests numériques en grande dimension on prendra des valeurs de n (nombre de nœuds par côté, $n = 4$ pour la figure) de l'ordre de 10^2, voire $5\,10^2$ ou plus. La matrice K du système aura donc une dimension $n^2 \times n^2$ c.-à-d. $10^4 \times 10^4$, voire $25\,10^4 \times 25\,10^4$. Une particularité de cette matrice est qu'elle est très « creuse », c.-à-d. « beaucoup »de ses coefficients sont égaux à zéros (environ $5\,n^2$ coefficients non nuls pour un total de n^4 coefficients). Cela signifie que le produit matrice fois vecteur peut se faire avec un nombre de multiplications bien inférieur à la valeur nominale. Pour une matrice

$m \times m$ il faut en théorie m^2 multiplications pour réaliser le produit matrice fois vecteur. Dans le cas présent ce produit nécessite seulement $5m$ mulplications (c.-à-d. $5\,n^2$ multiplications puisque $m = n^2$). Pour tirer parti de cette propriété il faut utiliser le format « sparse »(qui veut dire creux en anglais) qui ne stocke que les coefficents non nuls d'une matrice.

Remarques :

(a) Pour initialiser la matrice K il est plus simple d'utiliser directement la formule $K = (I_n \otimes D^T D) + (D^T D \otimes I_n)$ où le produit $D^T D$ est donné par la matrice tridiagonale $n \times n$

$$\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}.$$

(fonctions diags et kron de la librairie python scipy.sparse)

(b) Pour intialiser le membre de droite f pour tout n il faut bien entendu utiliser les vecteurs de base e_1 et e_n de $\mathbb{R}^n$ dans l'expression de f ci-dessus.

2. Description de la méthode de résolution numérique du système linéaire

(a) Principe de la méthode de projection

Il s'agit de résoudre le système linéaire

$$Ku = f \tag{1}$$

avec $K \in \mathbb{R}^{m\times m}$, m grand (de l'ordre de $10^4, 10^5, 10^6$ ou plus) et K matrice creuse. Le principe de la méthode de projection consiste à chercher une solution approchée du système (1) dans un sous-espace vectoriel $\mathcal{V}_k$ de $\mathbb{R}^m$ de dimension $k << m$ (la notation $<<$ indique « beaucoup »plus petit que). Ce faisant, l'égalité $Kv = f$ avec $v \in \mathcal{V}_k$ ne pourra être réalisée qu'au sens des moindres carrés puisqu'à priori on ne sait pas dans quel sous-espace $\mathcal{V}_k$ se trouve la solution u du problème (1).

On note V_k la matrice $m \times k$ ayant pour colonnes les k vecteurs d'une base du sous-espace $\mathcal{V}_k$. Tout vecteur $v \in \mathcal{V}_k$ s'écrit donc $v = V_k\alpha$ avec $\alpha \in \mathbb{R}^k$ vecteur des coefficients de v dans la base donnée par les colonnes de la matrice V_k.

La solution $v \in \mathcal{V}_k$ est donc obtenue en résolvant le problème de moindres carrés

$$\min_{\alpha\in\mathbb{R}^k} \|KV_k\alpha - f\|^2$$

De manière plus générale, si v_0 est une première approximation de la solution u du système (1), on peut chercher une approximation v dans le sous-espace affine $v_0 + \mathcal{V}_k$. Sachant que tout vecteur $v \in v_0 + \mathcal{V}_k$ s'écrit $v = v_0 + V_k\alpha$ avec $\alpha \in \mathbb{R}^k$, cela conduit donc à considérer le problème de moindres carrés

$$\min_{\alpha\in\mathbb{R}^k} \|KV_k\alpha - r_0\|^2 \tag{2}$$

où $r_0 = f - Kv_0$ est le résidu obtenu pour l'approximation v_0. Soit α_k la solution du problème (2). Notons $r_k = r_0 - KV_k\alpha_k$ le résidu obtenu à partir de α_k.

La méthode d'approximation consiste ensuite à faire croître le sous-espace d'approximation $\mathcal{V}_k$ en lui adjoignant le vecteur résidu r_k (en supposant que le vecteur r_k n'appartient pas au sous-espace $\mathcal{V}_k$ sans quoi on n'augmente pas la dimension du sous-espace d'approximation). On recommence ensuite dans ce nouvel espace noté $\mathcal{V}_{k+1}$ et généré par les $k+1$ colonnes de la matrice $V_{k+1} = (V_k\ r_k) \in \mathbb{R}^{m\times(k+1)}$ (matrice V_k augmentée de la colonne r_k) la recherche d'une nouvelle solution du problème de moindres carrés (2) pour $k = k+1$ (sous-espace d'approximation augmenté).

On a ainsi un procédé itératif dans lequel à chaque étape k on résout un problème de moindres carrés (2) et on utilise ensuite le résidu obtenu r_k pour faire croître le sous-espace d'approximation $\mathcal{V}_k$. Si on trouve $r_k = 0$ alors le processus se termine puisque cela signifie que la solution du problème de départ (1) a été trouvée dans l'espace $\mathcal{V}_k$. Le procédé démarre avec le sous-espace $\mathcal{V}_1 = \mathbb{R}r_0$ (droite vectorielle générée par le vecteur r_0).

(b) **Résolution numérique des problèmes de moindres carrés**

On sait que la solution du problème classique des moindres carrés

$$\min_{x\in\mathbb{R}^k} \|Mx - c\|^2$$

avec $M \in \mathbb{R}^{m\times k}$, $m \geq k$ et $\mathrm{rang}(M) = k$, $c \in \mathbb{R}^m$, est donnée par la solution du système

$$M^TMx = M^Tc \text{ (appelé aussi équations normales)}$$

Pour des raisons numériques liées à l'utilisation d'une arithmétique de précision finie (représentation des nombres sur un ordinateur) qui ne sont pas analysées dans cette présentation, on sait que la résolution directe des équations normales peut conduire à une perte de la précision de la solution x obtenue. La méthode numérique préconisée pour résoudre le problème des moindres carrés est d'utiliser la décomposition dite QR de la matrice M.

i. **Décomposition QR et procédé d'orthonormalisation de Gram-Schmidt**

On appelle décomposition QR de la matrice $M \in \mathbb{R}^{m\times k}$, $m \geq k$, l'égalité $M = QR$ où $Q \in \mathbb{R}^{m\times k}$ est orthogonale c.-à-d. vérifie $Q^TQ = I_k$ et $R \in \mathbb{R}^{k\times k}$ est triangulaire supérieure. On sait que cette décomposition est unique si $\mathrm{rang}(M) = k$ et $r_{ii} > 0, \forall i = 1,\ldots,k$ (autrement dit tous les coefficients de la diagonale de la matrice R sont strictement positifs).

Cette décomposition est de fait la traduction matricielle du procédé très général dit d'orthonormalisation de Gram-Schmidt qui calcule à partir de k vecteurs indépendants $m_1,\ldots,m_k$, de $\mathbb{R}^m$, k vecteurs orthonormés $q_1,\ldots,q_k$, tels que les sous-espaces vectoriels engendrés respectivement par $\{m_1,\ldots,m_j\}$ et $\{q_1,\ldots,q_j\}$ sont identiques pour tout $j = 1,\ldots,k$.

Notons m_j les colonnes de M (ainsi $M = (m_1\, m_2 \ldots m_k)$) et q_j les colonnes de Q (ainsi $Q = (q_1\, q_2 \ldots q_k)$). Notons r_{ij} les coefficients de la matrice triangulaire supérieure R. On a donc $r_{ij} = 0$ pour tout $i > j$. L'égalité $M = QR$ se traduit par l'égalité des k colonnes des deux membres de l'égalité, autrement dit, pour tout $j = 1,\ldots,k$, on a

$$m_j = r_{1j}q_1 + r_{2j}q_2 + \ldots + r_{jj}q_j.$$

Cette égalité permet de calculer de proche en proche à partir des différentes colonnes de M les colonnes de Q (les vecteurs orthonormés) et les coefficients de la matrice R.

— Commençons par $j = 1$: on a $m_1 = r_{11}q_1$. Afin que q_1 soit normé, il faut donc prendre $r_{11} = \|m_1\|$ et $q_1 = \frac{1}{r_{11}}m_1$.

— Pour $j = 2$, on a l'égalité $m_2 = r_{12}q_1 + r_{22}q_2$. Cette égalité permet de calculer les coefficients r_{12}, r_{22}, et q_2 normé et orthogonal à q_1, de la façon suivante : l'orthogonalité $q_1^T q_2 = 0$ et la normalité $q_1^T q_1 = 1$ donnent $r_{12} = q_1^T m_2$, puis $r_{22} = \|m_2 - r_{12}q_1\|$, et enfin $q_2 = \frac{1}{r_{22}}(m_2 - r_{12}q_1)$.

— Plus généralement, pour tout $j = 2, \ldots, k$ on obtient $r_{ij}, i = 1, \ldots, j$ et q_j à partir des vecteurs $q_1, \ldots, q_{j-1}$: $r_{ij} = q_i^T m_j, i = 1, \ldots, j-1$, puis $r_{jj} = \|m_j - \sum_{i=1}^{j-1} r_{ij}q_i\|$, et enfin

$$q_j = \frac{1}{r_{jj}}(m_j - \sum_{i=1}^{j-1} r_{ij}q_i).$$

En notant $Q_{j-1} = (q_1\, q_2 \ldots q_{j-1})$ la matrice $m \times (j-1)$ ayant pour colonnes les vecteurs $q_i, i = 1, \ldots j-1$, on a une écriture matricielle de l'égalité donnant q_j à partir des vecteurs $q_i, i = 1, \ldots j-1$:

$$q_j = \frac{1}{r_{jj}}(I_m - Q_{j-1}Q_{j-1}^T)m_j,$$

avec $r_{jj} = \|(I_m - Q_{j-1}Q_{j-1}^T)m_j\|$.

ii. **Résolution du problème des moindres carrés par la méthode QR**

Si l'on remplace M par le produit QR dans les équations normales $M^T Mx = M^T c$ associées au problème des moindres carrés, on obtient $R^T Q^T QRx = R^T Q^T c$, et donc $R^T Rx = R^T Q^T c$, puisque $Q^T Q = I_k$ et donc

$$Rx = Q^T c \tag{3}$$

car la matrice R est inversible (du fait que $\mathrm{rang}(M) = k$). Ce système est particulièrement simple à résoudre puisqu'il s'agit d'un système où la matrice est triangulaire supérieure. La résolution se fait de proche en proche en remontant à partir de la dernière ligne.

Le calcul du résidu $r = c - Mx$ avec x solution de (3) s'obtient alors à partir de la matrice Q : on a $r = c - QRx = c - QQ^T c = (I_m - QQ^T)c$. On retrouve le fait que le problème des moindres carrés consiste à projeter orthogonalement le vecteur c sur le sous-espace vectoriel $\mathrm{Im}(M)$ avec $\mathrm{Im}(M) = \mathrm{Im}(Q)$.

(c) **Mise à jour de la décomposition QR**

On rappelle que dans l'algorithme décrit au paragraphe 2-a le sous-espace d'approximation $\mathcal{V}_{k+1}$ est obtenu à partir du sous-espace $\mathcal{V}_k$ en ajoutant le nouveau vecteur résidu r_k. Le sous-espace $\mathcal{V}_{k+1}$ est généré par les vecteurs colonnes de la matrice augmentée $V_{k+1} = (V_k\, r_k) \in \mathbb{R}^{m\times(k+1)}$. C'est donc à nouveau le problème des moindres carrés (2) qu'il faut résoudre pour $k+1$. Pour cela, il est nécessaire de mettre à jour la décomposition QR (que l'on suppose connue) de la matrice des moindres carrés précédente à l'ordre k, que l'on note M_k : on a $M_k = KV_k = Q_k R_k$, avec $Q_k = (q_1\, q_2 \ldots q_k) \in \mathbb{R}^{m\times k}$ orthogonale et $R_k \in \mathbb{R}^{k\times k}$ triangulaire supérieure.

Il est facile de voir par identification que le nouveau vecteur normé q_{k+1} orthogonal aux vecteurs $\{q_1, q_2, \ldots, q_k\}$ est donné par

$$q_{k+1} = \frac{1}{s}(I_m - Q_k Q_k^T)Kr_k,$$

où $s = \|(I_m - Q_k Q_k^T) K r_k\|$.
La nouvelle colonne $r_{k+1} \in \mathbb{R}^{k+1}$ (colonne $k+1$) de la matrice triangulaire supérieure R_{k+1} est donnée par

$$r_{k+1} = \begin{pmatrix} Q_k^T K r_k \\ s \end{pmatrix}.$$

On a donc

$$Q_{k+1} = (Q_k \; q_{k+1})$$

et

$$R_{k+1} = \begin{pmatrix} R_k & Q_k^T K r_k \\ 0 & s \end{pmatrix}.$$

Ainsi, grâce à la décomposition QR de la matrice $M_{k+1} = KV_{k+1} = Q_{k+1}R_{k+1}$, on peut résoudre le problème des moindres carrés (2) à l'ordre $k+1$.

Remarques :
À chaque étape k de l'algorithme il n'est pas nécessaire de calculer la solution α_k du problème de moindres carrés (2) mais uniquement le résidu

$$r_k = (I_m - Q_k Q_k^T) r_0.$$

C'est grâce à ce résidu r_k que l'espace d'approximation $\mathcal{V}_k$ est enrichi. La formule précédente donne la récurrence $r_k = r_{k-1} - q_k q_k^T r_0$. Il est préférable d'utiliser cette formule qui est plus « économique »que la précédente.
La solution approchée $v_k = v_0 + V_k \alpha_k$ ne sera calculée que lors de la dernière itération.

(d) **Propriétés de l'algorithme de projection**

i. On a

$$\|r_{k+1}\| \leq \|r_k\|, \; \forall k.$$

La norme des résidus est décroissante. Ce résultat découle simplement de l'inclusion $\mathcal{V}_k \subset \mathcal{V}_{k+1}$.

ii. Sachant que K est définie positive, on a le résultat suivant : si $r_k \neq 0$, alors r_k est indépendant des vecteurs colonnes de la matrice V_k (autrement dit des vecteurs résidus précédents $\{r_0, r_1, \ldots, r_{k-1}\}$). De fait, on a $r_k^T K r_j = 0$, $\forall j = 0, \ldots, k-1$. On a orthogonalité de r_k avec $r_j, j = 0, \ldots, k-1$, au sens du produit scalaire défini par la matrice K définie positive. Ce résultat implique l'indépendance de r_k avec les vecteurs $r_j, j = 0, \ldots, k-1$.

iii. L'algorithme converge vers la solution du système $Ku = f$ au terme de $m-1$ itérations au plus. En effet, soit pour $k < m-1$ on a $r_k = 0$ et donc la solution u est dans l'espace affine $v_0 + \mathcal{V}_k$, soit on a $r_k \neq 0$ pour tout $k = 1, \ldots, m-1$, et donc $V_m = \mathbb{R}^m$ d'après la propriété précédente. En pratique, pour des systèmes de grande dimension, le nombre d'itérations utilisées reste toujours très inférieur à la dimension du système.

iv. (À titre d'information générale) On montre que pour tout k le sous-espace $\mathcal{V}_k$ est égal au sous-espace dit de Krylov (noté $\mathcal{K}_k(K, r_0)$) généré par les vecteurs $\{r_0, Kr_0, K^2 r_0, \ldots, K^{k-1} r_0\}$.

(e) **Méthode de « restarting »**

La méthode dite de « restarting »(de rédémarrage) est une variante de l'algorithme précédent pour remedier au problème de stockage des variables engendrés tout au long des itérations. En

effet l'algorithme nécessite le stockage de la matrice V_k, de la matrice orthogonale Q_k, et de la matrice triangulaire supérieure R_k (au total $2mk + k^2/2$ nombres flottants). Ce stockage augmente avec k. Le principe de la méthode de « restarting »et de reboucler sur l'algorithme en repartant de la valeur du résidu courant r_k obtenu au terme de k itérations. Cela signifie, dans la description précédente de l'algorithme, qu'on démarre une seconde fois l' algorithme en prenant $r_0 = r_k$, et on itère ensuite suivant la procédure normale. Ce mécanisme de restarting peut ensuite être pratiqué plusieurs fois. On a ainsi une boucle extérieure qui contrôle le nombre de redémarrages et une boucle intérieure qui contrôle le nombre d'itérations à partir d'un vecteur de résidu initial fixé (suivant l'algorithme décrit). Pour tester cette approche, il suffit d'appeler dans une boucle extérieure la fonction réalisant l'algorithme, en modifiant à chaque itération la valeur en entrée du résidu r_0 (ou plutôt de l'approximation v_0) grâce à la valeur obtenue en sortie de l'appel précédent (au terme d'une boucle intérieure).

3. **Travail demandé**

   (a) Initialiser la matrice $K \in \mathbb{R}^{m\times m}$ (matrice du système) et le vecteur $f \in \mathbb{R}^m$ (membre de droite). Utiliser le format « sparse »sans quoi les calculs ne pourront être exécutés pour de grandes valeurs de m. Considérer plusieurs valeurs aux bords du domaine (membre de droite). Par exemple u constant égal à 1 sur certains côtés du carré et égal à 0 sur d'autres.
   Fonctions python : diags, kron.

   (b) Écrire une fonction python réalisant les différentes étapes de l'algorithme décrit plus haut.
   Arguments d'entrée : matrice K, membre de droite f, approximation initiale v_0 (on peut prendre $v_0 = 0$ ou un vecteur aléatoire), nombre d'itérations k de l'algorithme.
   Arguments de sortie : liste des normes des résidus successifs, approximation v_k obtenue au terme des k itérations, matrices V_k, Q_k, R_k (pour vérifications).
   Utiliser la fonction python solve_triangular pour résoudre le système triangulaire supérieur (une seule fois au terme des k itérations).

   (c) Implémenter la variante « restarting »(appels successifs de la fonction précédente). Faire des tests avec plusieurs valeurs pour le nombre d'itérations sur la boucle extérieure ($next$) et sur la boucle intérieure ($nint$). Il n'existe pas de résultats donnant des indications sur des valeurs optimales quant au rapport $next/nint$. Faire des tests avec $\frac{next}{nint} \approx 1, 0.5, 2$, et comparer avec l'algorithme simple (par de redémarrage).

   (d) Tracer les solutions obtenues v_k avec la fonction graphique contour (ou contourf). Pour cela il est necessaire de transformer le vecteur $v_k \in \mathbb{R}^m$ en matrice $n \times n$ ($m = n^2$). Le réarrangement se fait ligne par ligne à l'aide de la fonction python reshape.

   (e) Comparer les solution obtenues avec les solutions « exactes »données par la résolution directe du système (fonction python spsolve). La fonction spsolve utilise la méthode directe du pivot de Gauss et tient compte du fait que la matrice est creuse. Néanmoins elle peut être défaillante dans le cas de dimensions trop importantes ($m \geq 10^6, 10^7$).

   (f) Présenter les résultats avec les fonctions, graphes et commentaires dans un fichier notebook.

Remarque et critique de la méthode : On notera que la vitesse de convergence de la méthode de projection développée dans cette étude est en définitive assez faible. Pour augmenter la vitesse de convergence il aurait fallu « préconditionner »le système afin de diminuer le conditionnement de la matrice du système. Mais cela demande plus de développements et de considérations théoriques et dépasse le cadre simplifié de cette étude.