Solveurs linéaires pour les problèmes industriels
Algorithmes du cours

Rodolphe Turpault

Ce document contient les algorithmes vus pendant le cours.

Etant donné que $A,\ b, x_0$ et ε seront des entrées de presque tous les algorithmes qui suivront et que x_k en sera toujours une sortie, on les omettra par souci de lisibilité à partir de l'algorithme 2.

Notation : $< x, y >$ représente le produit scalaire euclidien de deux vecteurs de $\mathbb{R}^n$ et $\|\cdot\|$ est la norme associée.

Algorithme 1 Algorithme générique d'une méthode itérative - version naïve

---

Entrées: Les données du problème $A \in \mathcal{M}_n(\mathbb{R})$ et $b \in \mathbb{R}^n$, un choix initial $x_0 \in \mathbb{R}^n$ et la tolérance max. $\varepsilon > 0$.
Sorties: x une approximation de la solution du système linéaire.
Calculer l'erreur initiale,
$k \leftarrow 0$,
Tantque (erreur$> \varepsilon$) **faire**
  $k \leftarrow k + 1$,
  Calculer x_k,
  Calculer l'erreur,
Fin tantque
Retourner $x \leftarrow x_k$.

Algorithme 2 Algorithme générique d'une méthode itérative - version améliorée

$r_0 \leftarrow b - Ax_0$,
$k \leftarrow 0$,
Tantque $(\|r_k\| > \varepsilon)$ et $(k \leq k_{max})$ **faire**
  $k \leftarrow k + 1$,
  Calculer x_k,
  $r_k \leftarrow b - Ax_k$,
Fin tantque
Retourner $x \leftarrow x_k$,
Si $(k > k_{max})$ **alors**
  Retourner 'Tolérance non atteinte :', $\|r_k\|$
Finsi

Algorithme 3 *Gradient à pas optimal*

$r \leftarrow b - Ax_0$,
$k \leftarrow 0$,
Tantque $(\|r\| > \varepsilon)$ et $(k \leq k_{max})$ **faire**
  $z \leftarrow Ar$,
  $\alpha \leftarrow < r, r > / < z, r >$,
  $x \leftarrow x + \alpha r$,
  $r \leftarrow r - \alpha z$,
  $k \leftarrow k + 1$,
Fin tantque
Retourner x,
Si $(k > k_{max})$ **alors**
  Retourner 'Tolérance non atteinte :', $\|r\|$
Finsi

Algorithme 4 Gradient Conjugué

$r_0 \leftarrow b - Ax_0$,
$p \leftarrow r_0$,
$\beta \leftarrow \|r_0\|$
$k = 0$,
Tantque $(\beta > \varepsilon)$ et $(k \leq k_{max})$ **faire**
  $z = Ap$,
  $\alpha \leftarrow < r_k, r_k > / < z, p >$,
  $x \leftarrow x - \alpha p$,
  $r_{k+1} \leftarrow r_k - \alpha z$,
  $\gamma \leftarrow < r_{k+1}, r_{k+1} > / < r_k, r_k >$,
  $p \leftarrow r_{j+1} + \gamma p$,
  $\beta \leftarrow \|r_k\|$
  $k \leftarrow k + 1$,
Fin tantque
Retourner x,
Si $(k > k_{max})$ **alors**
  Retourner 'Tolérance non atteinte :', $\|r\|$
Finsi

Algorithme 5 *Produit matrice/vecteur avec le format CSR*

$y \leftarrow 0$,
Pour $i = 1, \ldots, n$ **faire**
  Pour $k = IA(i), \ldots, IA(i+1)$ **faire**
    $y(i) \leftarrow y(i) + AA(k)x(JA(k))$,
  Fin pour
Fin pour

Algorithme 6 Résidu Minimum Préconditionné à gauche

$r \leftarrow b - Ax_0$,
Résoudre $Mq = r$,
$k \leftarrow 0$,
Tantque $(\|r\| > \varepsilon)$ et $(k \leq k_{max})$ **faire**
  $w =\leftarrow Aq$
  Résoudre $Mz = w$,
  $\alpha \leftarrow < q, z > / < z, z >$,
  $x \leftarrow x + \alpha q$,
  $r \leftarrow r - \alpha w$,
  $q \leftarrow q - \alpha z$,
  $k \leftarrow k + 1$,
Fin tantque
Retourner x,
Si $(k > k_{max})$ **alors**
  Retourner 'Tolérance non atteinte :', $\|r\|$
Finsi

Algorithme 7 ILU0

$M \leftarrow A$,
Pour $i = 2, n$ **faire**
  Pour $(k = 1, i - 1)$ et $((i, k) \in \mathcal{D})$ **faire**
    $m_{i,k} \leftarrow m_{i,k}/m_{k,k}$,
    Pour $(j = k + 1, n)$ et $((i, j) \in \mathcal{D})$ **faire**
      $m_{i,j} \leftarrow m_{i,j} - m_{i,k}m_{k,j}$,
    Fin pour
  Fin pour
Fin pour

Algorithme 8 Algorithme générique d'une méthode de projection 1D

$r_0 \leftarrow b - Ax_0$,
$k \leftarrow 0$,
Tantque $(\|r_k\| > \varepsilon)$ et $(k \leq k_{max})$ **faire**
  $\alpha \leftarrow < r_k, w > / < Av, w >$,
  $x_{k+1} \leftarrow x_k + \alpha v$,
  $r_k \leftarrow b - Ax_k$,
  $k \leftarrow k + 1$,
Fin tantque
Retourner $x \leftarrow x_k$,
Si $(k > k_{max})$ **alors**
  Retourner 'Tolérance non atteinte :', $\|r_k\|$
Finsi

Algorithme 9 Algorithme générique d'une méthode de projection 1D - version améliorée

$r \leftarrow b - Ax_0$,
$k \leftarrow 0$,
Tantque $(\|r\| > \varepsilon)$ et $(k \leq k_{max})$ **faire**
  $z \leftarrow Av$,
  $\alpha \leftarrow < r, w > / < z, w >$,
  $x \leftarrow x + \alpha v$,
  $r \leftarrow r - \alpha z$,
  $k \leftarrow k + 1$,
Fin tantque
Retourner x,
Si $(k > k_{max})$ **alors**
  Retourner 'Tolérance non atteinte :', $\|r\|$
Finsi

Algorithme 10 *Steepest Descent*

$r \leftarrow b - Ax_0$,
$k \leftarrow 0$,
Tantque $(\|r\| > \varepsilon)$ et $(k \leq k_{max})$ **faire**
  $z \leftarrow Ar$,
  $\alpha \leftarrow < r, r > / < z, r >$,
  $x \leftarrow x + \alpha r$,
  $r \leftarrow r - \alpha z$,
  $k \leftarrow k + 1$,
Fin tantque
Retourner x,
Si $(k > k_{max})$ **alors**
  Retourner 'Tolérance non atteinte :', $\|r\|$
Finsi

Algorithme 11 Résidu Minimum

$r \leftarrow b - Ax_0$,
$k \leftarrow 0$,
Tantque $(\|r\| > \varepsilon)$ et $(k \leq k_{max})$ **faire**
  $z \leftarrow Ar$,
  $\alpha \leftarrow < r, z > / < z, z >$,
  $x \leftarrow x + \alpha r$,
  $r \leftarrow r - \alpha z$,
  $k \leftarrow k + 1$,
Fin tantque
Retourner x,
Si $(k > k_{max})$ **alors**
  Retourner 'Tolérance non atteinte :', $\|r\|$
Finsi

Algorithme 12 Alogrithme d'Arnoldi (Gram-Schmidt) - version naïve

$v_1 := v/\|v\|$,
Pour $j = 1, \ldots, m$ **faire**
  Pour $i = 1, \ldots, m$ **faire**
    $h_{i,j} =< Av_j, v_i >$,
  Fin pour
  $z_j = Av_j - \sum_{i=1}^{m} h_{i,j} v_i$,
  $h_{j+1,j} = \|z_j\|$,
  Si $h_{j+1,j} = 0$ **alors**
    Stop
  Finsi
  $v_{j+1} = z_j / h_{j+1,j}$,
Fin pour

Algorithme 13 FOM

$r \leftarrow b - Ax_0$,
$\beta \leftarrow \|r\|$
$k = 0$,
Tantque $(\beta > \varepsilon)$ et $(k \leq k_{max})$ **faire**
  Utiliser la méthode d'Arnoldi en partant de r pour obtenir H_m et V_m,
  Résoudre $H_m y = \beta e_1$,
  $x \leftarrow x + V_m y$,
  $r \leftarrow -h_{m+1,m} < y, e_m > v_{m+1}$,
  $\beta \leftarrow \|r\|$
  $k \leftarrow k + 1$,
Fin tantque
Retourner x,
Si $(k > k_{max})$ **alors**
  Retourner 'Tolérance non atteinte :', $\|r\|$
Finsi

Algorithme 14 GMRes

$r \leftarrow b - Ax_0$,
$\beta \leftarrow \|r\|$
$k \leftarrow 0$,
Tantque $(\beta > \varepsilon)$ et $(k \leq k_{max})$ **faire**
  Utiliser la méthode d'Arnoldi en partant de r pour obtenir H_m et V_m,
  Calculer $y = \text{argmin}\|\beta e_1 - \bar{H}_m y\|$,
  $x \leftarrow x + V_m y$,
  $r \leftarrow \|\beta e_1 - \bar{H}_m y\|$,
  $\beta \leftarrow \|r\|$,
  $k \leftarrow k + 1$,
Fin tantque
Retourner x,
Si $(k > k_{max})$ **alors**
  Retourner 'Tolérance non atteinte :', $\|r\|$
Finsi

Algorithme 15 Méthode de la puissance

Entrées: $y_0 \in \mathbb{R}^n_\star$
$y \leftarrow y_0$,
$\beta \leftarrow 1$,
$k \leftarrow 0$,
Tantque $(\beta > \varepsilon)$ et $(k \leq k_{max})$ **faire**
  $z \leftarrow Ay$,
  $z \leftarrow \frac{z}{\|z\|}$,
  $\beta \leftarrow \|y - z\|$,
  $y \leftarrow z$,
  $k \leftarrow k + 1$,
Fin tantque
$\lambda \leftarrow \frac{<Ay,y>}{\|y\|^2}$,
Retourner y et λ,
Si $(k > k_{max})$ **alors**
  Retourner 'Tolérance non atteinte :', β
Finsi

Algorithme 16 Méthode de la puissance inverse

Entrées: $y_0 \in \mathbb{R}^n$

$y \leftarrow y_0$,
$\beta \leftarrow 1$,
$k \leftarrow 0$,
Tantque $(\beta > \varepsilon)$ et $(k \leq k_{max})$ **faire**
Résoudre $(A - \mu I)z = y$,
$z \leftarrow \frac{z}{\|z\|}$,
$\beta \leftarrow \|y - z\|$,
$y \leftarrow z$,
$k \leftarrow k + 1$,
Fin tantque
$\lambda \leftarrow \frac{<Ay,y>}{\|y\|^2}$,
Retourner y et λ,
Si $(k > k_{max})$ **alors**
Retourner 'Tolérance non atteinte :', β
Finsi

Algorithme 17 Méthode de déflation

$B = A$,
$j \leftarrow 1$,
Tantque $(j \leq j_{max})$ **faire**
Utiliser la méthode de la puissance sur B (résultat λ_j et v_j),
$B = B - \lambda_j v_j v_j^\top$,
$j \leftarrow j + 1$,
Fin tantque
Retourner $(y_j)_j$ et $(\lambda_j)j$.

Algorithme 18 Itération orthogonale

$B = A$,
$j \leftarrow 1$,
Tantque $(j \leq j_{max})$ **faire**
Faire la décomposition QR de B : $B = QR$,
$B \leftarrow RQ$,
$j \leftarrow j + 1$,
Fin tantque
Retourner les éléments diagonaux de R.