Projet MAP411 – X2015

Applications de méthodes d'optimisation à la résolution des EDP

Sujet proposé par Nicolas Augier – `nicolas.augier@ens-cachan.fr`

Lucas Lugão Guimarães – `lucas.lugao-guimaraes@polytechnique.edu`
Alexandre Ribeiro João Macedo – `alexandre.macedo@polytechnique.edu`

I. Méthodes de gradient

I.1. Gradient à pas constant

I.1.1. La figure 1 montre les lignes de niveau de la fonction de Rosenbrock,

$$J(u_1, u_2) = (u_1 - 1)^2 + 100(u_1^2 - u_2)^2, \tag{1}$$

dans le rectangle $(u_1, u_2) \in [-1.5, 1.5]$ x $[0.5, 1.5]$.

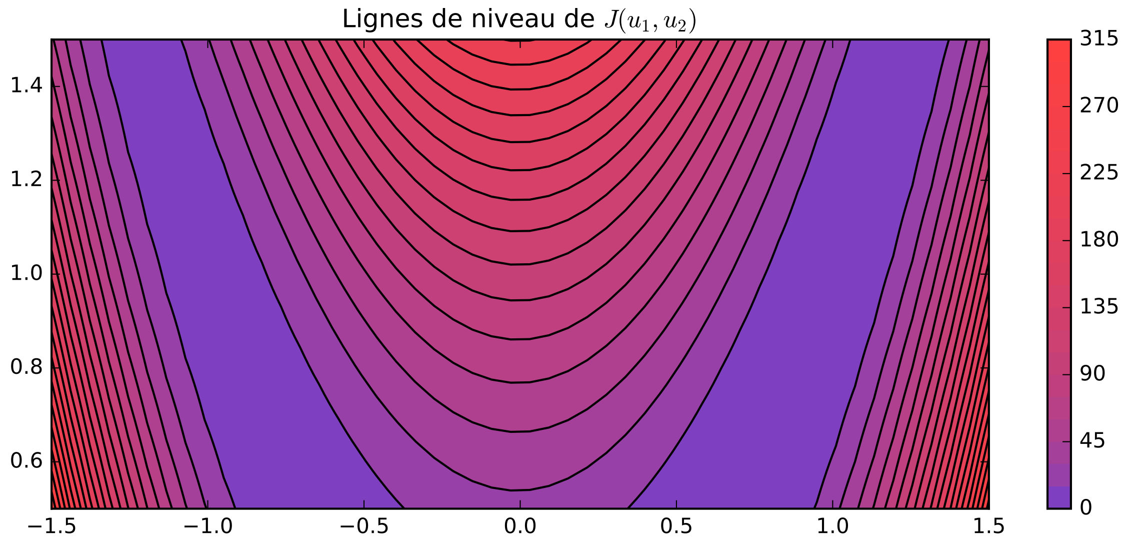

Figure 1 – Lignes de niveau de J.

I.1.2. On applique la méthode de gradient a pas constant $u_{k+1} = u_k - \rho\nabla J(u_k)$ ou, pour $u_k = (u_{k1}, u_{k2})$, on a $\nabla J(u_k) = (400u_{k1}^3 + 2u_{k1} - 400u_{k1}u_{k2} - 2, 200u_{k1}^2 - 200u_{k2})$. Pour les valeurs numériques $u_0 = (1, 0)$ et $\rho = 0.002$, et le critère d'arrêt $J(u_{k+1}) < 10^{-3}$, on a besoin de 2764 itérations. La figure 3 montre les lignes de niveau de l'équation 1 et le parcours du méthode avant l'arrête.

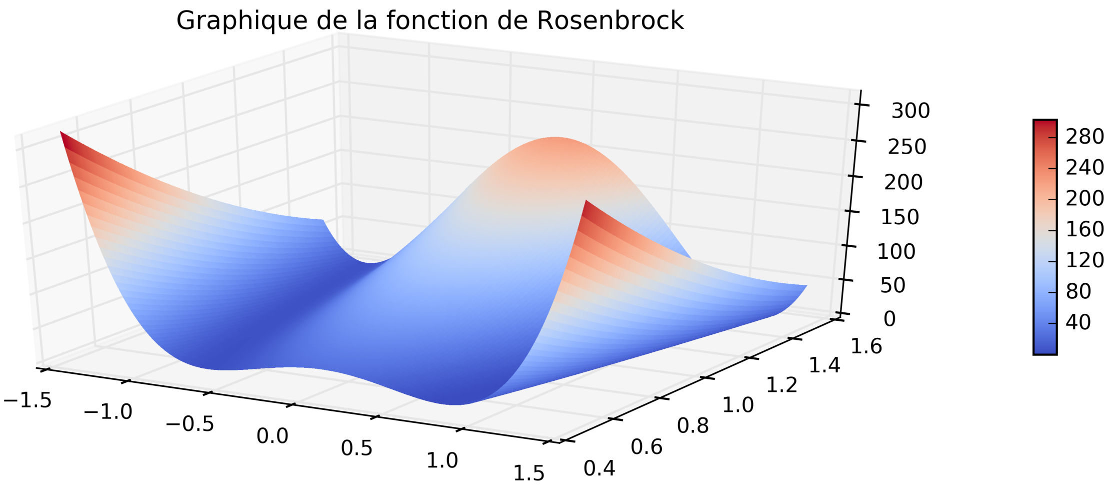

FIGURE 2 – Surface plot de la fonction de Rosenbrock.

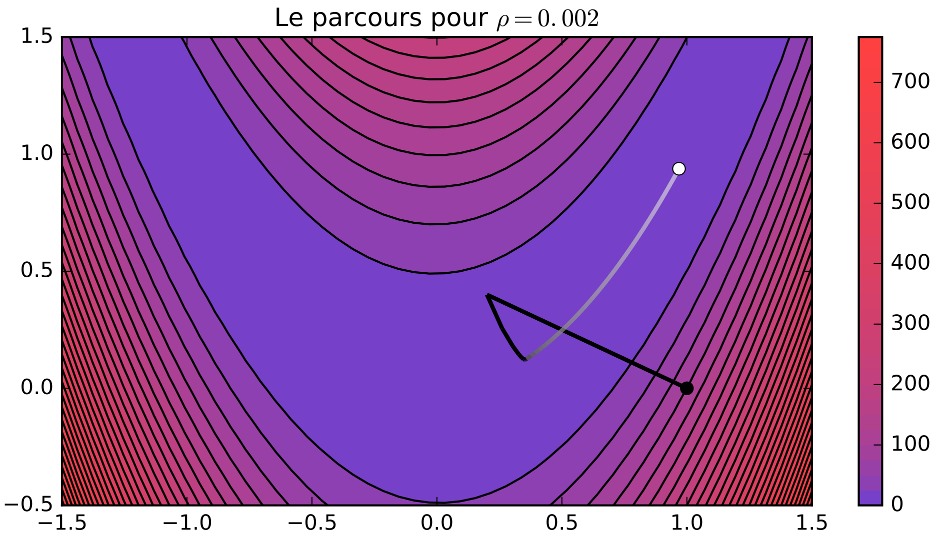

FIGURE 3 – Parcours du méthode de gradient à pas constant.

I.1.3. Si on applique la méthode pour $\rho = 0.0045$ et pour $\rho = 0.01$, on a que le méthode diverge. Pour la valeur $\rho = 0.0045$, la méthode oscille en tour du minimum et ne converge pas. Pour la valeur $\rho = 0.01$, la première itération est déjà très loin du point de minimum et il ne reviens pas.

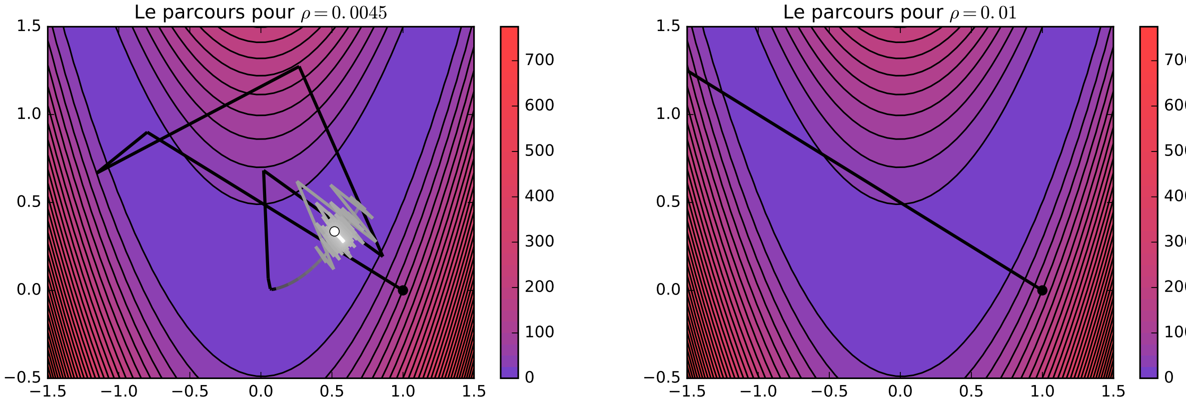

FIGURE 4 – Le parcours du méthode de gradient à pas constant $\rho = 0.045$ et $\rho = 0.01$.

Toutes les figures et résultats ci-dessus peuvent être crée par le code "problem1.py".

I.2. Gradient optimal

I.2.1. Soit $J(u) = \frac{1}{2}\langle Au, u\rangle - \langle b, u\rangle$. On veut trouver $\rho_k = \operatorname{argmin}_{\rho\geq 0} J(u_k - \rho\nabla J(u_k))$. Soit $r_k = \nabla J(u_k) = Au_k - b$, on a

$$
\begin{aligned}
J(u_k - \rho\nabla J(u_k)) &= J(u_k - \rho r_k) \\
&= \frac{1}{2}\langle A(u_k - \rho r_k), u_k - \rho r_k\rangle - \langle b, u_k - \rho r_k\rangle \\
&= \frac{1}{2}\langle Au_k, u_k\rangle - \langle b, u_k\rangle + \frac{1}{2}\langle Au_k, -\rho r_k\rangle + \frac{1}{2}\langle -\rho A r_k, u_k\rangle + \frac{1}{2}\langle -\rho A r_k, -\rho r_k\rangle \\
&\qquad - \langle b, -\rho r_k\rangle \\
&= J(u_k) - \frac{\rho}{2}[\langle Au_k, r_k\rangle + \langle Ar_k, u_k\rangle] + \frac{\rho^2}{2}\langle Ar_k, r_k\rangle + \rho\langle b, r_k\rangle \\
&= J(u_k) + \frac{\rho^2}{2}\langle Ar_k, r_k\rangle + \rho[\langle b, r_k\rangle - \langle Au_k, r_k\rangle] \\
&= J(u_k) + \frac{\rho^2}{2}\langle Ar_k, r_k\rangle + \rho[\langle b, r_k\rangle - \langle r_k - b, r_k\rangle] \\
&= J(u_k) + \frac{\rho^2}{2}\langle Ar_k, r_k\rangle + \rho\langle r_k, r_k\rangle.
\end{aligned}
$$

Alors, la valeur que minimise cette fonction quadratique en ρ est

$$\rho_k = \frac{\langle r_k, r_k\rangle}{\langle Ar_k, r_k\rangle}$$

I.2.2. Pour les valeur $A = \begin{pmatrix} 1 & 0 \\ 0 & \lambda \end{pmatrix}$ et $b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ on a

$$\begin{pmatrix} r_{k1} \\ r_{k2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \lambda \end{pmatrix} \begin{pmatrix} u_{k1} \\ u_{k2} \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} u_{k1} - 1 \\ \lambda u_{k2} - 1 \end{pmatrix}. \text{ Alors,}$$

$$\rho_k = \frac{(u_{k1} - 1)^2 + (\lambda u_{k2} - 1)^2}{(u_{k1} - 1)^2 + \lambda(\lambda u_{k2} - 1)^2} \tag{2}$$

À chaque itération, on actualise ρ_k en utilisant l'équation 2. Le figure 5 montre les résultats pour différentes condition initiales et pour le critère de arrête

$$\frac{\|r_k\|}{\|r_0\|} \leq 10^{-2}.$$

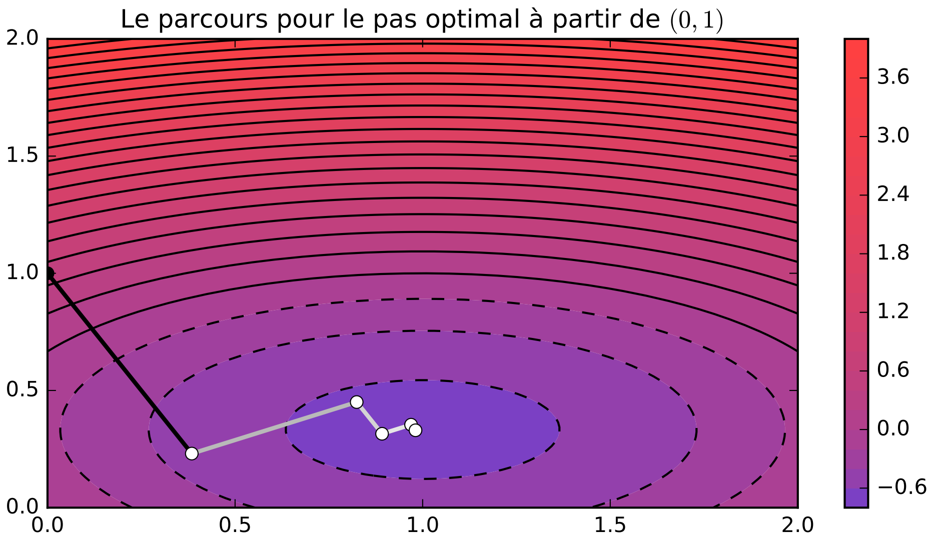

FIGURE 5 – Parcours du méthode de gradient à pas optimal.

I.2.3. Les figure 5 et 6 montre en échelle semi logarithme le rapport $\frac{\|r_k\|}{\|r_0\|}$ en fonction de k pour les cas montrés dans les figure 5 et 6. Ces résultats peuvent être trouver en utilisant le code python disponible dans "problem2.py"

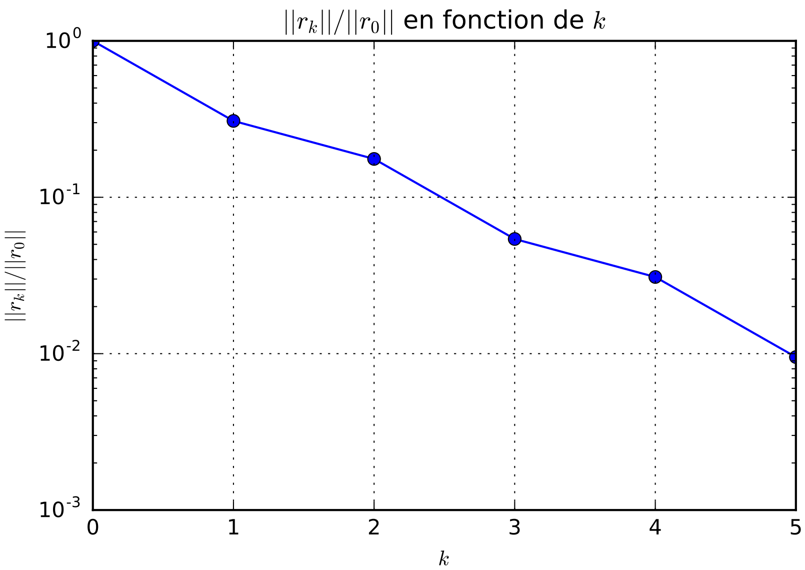

FIGURE 6 – Parcours du méthode de gradient à pas optimal.

I.2.4. Quand on met en œuvre le procédé de dichotomie pour l'équation 1, on trouve le résultat ci-dessous. Ce résultat a été géré par le code disponible dans le fichier "problem3.py".

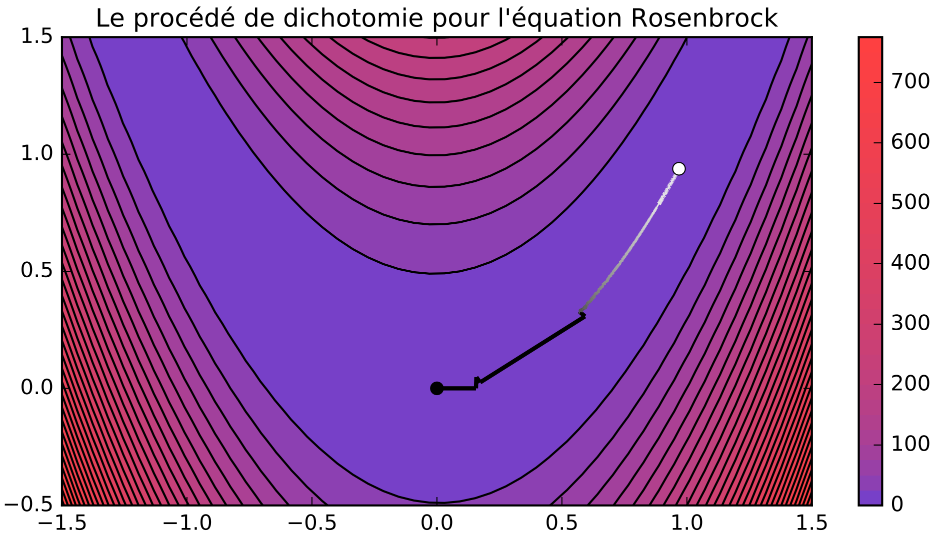

FIGURE 7 – Parcours obtenu à partir du procédé de dichotomie pour trouver le minimum global de la fonction de Rosenbrock.

II. Méthodes newtoniennes

II.1. Méthode de Newton

II.2. Méthode de Gauss-Newton

II.2.1. On note le vecteur des paramètres $\beta = \{\beta_1, \ldots, \beta_{3K}\}$ de manière que $\beta_j = \alpha_j$, $\beta_{j+K} = \sigma_j$ et $\beta_{j+2K} = x_j^0$. Ainsi la matrice jacobienne $Df(\beta)$ de dimension $N \times 3K$ est donné par $[Df(\beta)]_{ij} = \frac{\partial f_i(\beta)}{\partial \beta_j}$. À fin de la calculer on rappelle la définition du vecteur f $f_i(\beta) = y_i - g(x_i, \beta)$ avec $g(x_i, \beta) = \sum_{j=1}^{K} \alpha_j \exp\left(-\frac{1}{2}\frac{(x-x_j^0)^2}{\sigma_j^2}\right)$

$$[Df(\beta)]_{ij} = \frac{\partial f_i(\beta)}{\partial \alpha_j} = -\exp\left(-\frac{1}{2}\frac{(x-x_j^0)^2}{\sigma_j^2}\right), \quad 1 \le j \le K$$

$$[Df(\beta)]_{i(K+j)} = \frac{\partial f_i(\beta)}{\partial \sigma_j} = -\alpha_j \frac{(x-x_j^0)^2}{\sigma_j^3} \exp\left(-\frac{1}{2}\frac{(x-x_j^0)^2}{\sigma_j^2}\right), \quad 1 \le j \le K$$

$$[Df(\beta)]_{i(2K+j)} = \frac{\partial f_i(\beta)}{\partial x_j^0} = -\alpha_j \frac{(x-x_j^0)}{\sigma_j^2} \exp\left(-\frac{1}{2}\frac{(x-x_j^0)^2}{\sigma_j^2}\right), \quad 1 \le j \le K$$

On écrit le processus itératif à partir de l'algorithme de Newton $\beta_{k+1} = \beta_k - [DF(\beta_k)]^{-1}F(\beta_k)$ en prenant l'approximation $DF(\beta_k) = D[\nabla J(\beta_k)] = \nabla^2 J(\beta_k) \approx Df^T(u)Df(u)$ et en rappelant que $F(\beta_k) = \nabla J(\beta_k) = Df^T(\beta_k)f(\beta_k)$

$$\beta_{k+1} = \beta_k - [Df^T(\beta_k)Df(\beta_k)]^{-1}Df^T(\beta_k)f(\beta_k), \quad k \ge 0$$

II.2.2. Avec les conditions initiales $\alpha_1 = \alpha_2 = \sigma_1 = \sigma_2 = 1$, $x_1^0 = 3$ et $x_2^0 = 7$ la méthode de Gauss-Newton converge vers les valeurs suivantes

$$\alpha_1 = 0.22213775, \quad \alpha_2 = 0.98932187, \quad \sigma_1 = 1.27128674,$$
$$\sigma_2 = 2.64012043, \quad x_1^0 = 2.57935799, \quad x_2^0 = 9.16230523$$

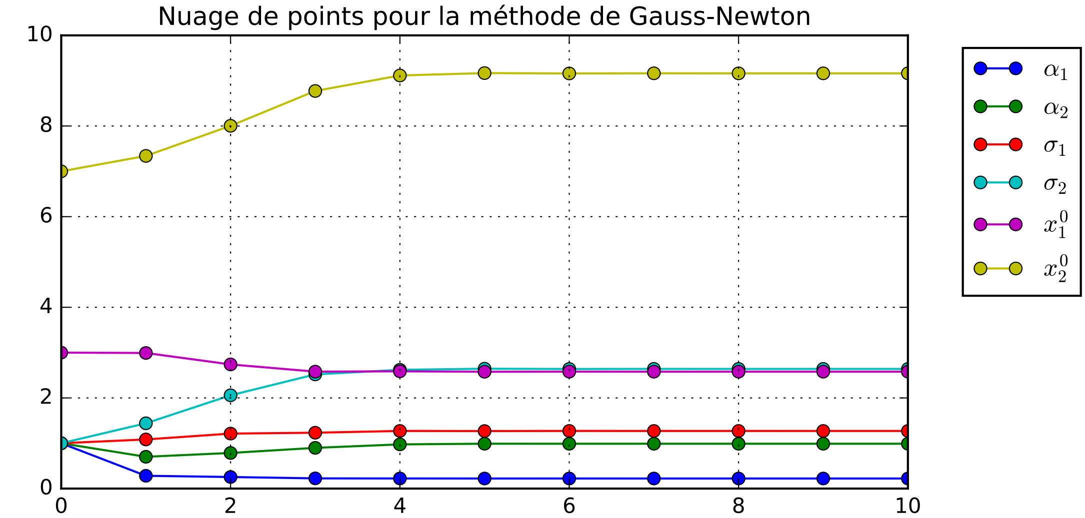

FIGURE 8 – Nuage de points obtenue à partir des conditions initiales $\alpha_1 = \alpha_2 = \sigma_1 = \sigma_2 = 1$, $x_1^0 = 3$ et $x_2^0 = 7$

II.2.3. D'après la figure 9 on voit la divergence de la méthode de Gauss-Newton avec des conditions initiales proches de la solution trouvé dans l'item précédent. De cette manière on voit la non-convergence locale de cette approximation de la Méthode de Newton classique.

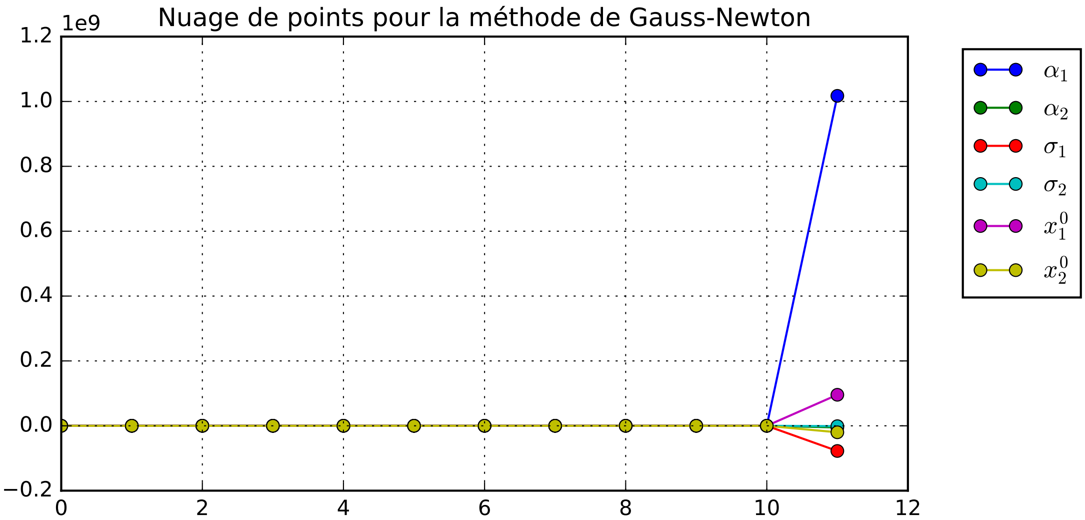

FIGURE 9 – Nuage de points obtenue à partir des conditions initiales $\alpha_1 = \alpha_2 = \sigma_1 = \sigma_2 = 1$, $x_1^0 = 3$ et $x_2^0 = 6$

Les résultats ci-dessus ont été crée par le code disponible dans "problem4.py".

III. Application à l'équation de convection-diffusion

III.1. Différences schémas d'approximation

III.1.1. On remplace u_i^n par $u(x_i, t_n)$ où u est une fonction régulière. On commence par le développement de Taylor au tour du point (x_i, t_n). Pour le schéma

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} - b\frac{u_{i+1}^n + u_{i-1}^n - 2u_i^n}{\Delta x^2} = 0, \text{ on a} \tag{3}$$

$$\begin{aligned} u(x_{i\pm1}, t_n) = u(x_i, t_n) \pm \Delta x\frac{\partial u}{\partial x}(x_i, t_n) + \frac{1}{2}\Delta x^2\frac{\partial^2 u}{\partial x^2}(x_i, t_n) \pm \frac{1}{6}\Delta x^3\frac{\partial^3 u}{\partial x^3}(x_i, t_n) \\ + \frac{1}{24}\Delta x^4\frac{\partial^4 u}{\partial^4 x}(x_i, t_n) + \mathcal{O}(\Delta x^5), \end{aligned}$$

$$u(x_i, t_{n+1}) = u(x_i, t_n) + \Delta t\frac{\partial u}{\partial t}(x_i, t_n) + \frac{1}{2}\Delta t^2\frac{\partial^2 u}{\partial t^2}(x_i, t_n) + \mathcal{O}(\Delta t^3).$$

On remplace le développement dans l'erreur de troncature qui est définit comme

$$\begin{aligned} E &= \frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} + b\frac{u_{i+1}^n + u_{i-1}^n - 2u_i^n}{\Delta x^2} \\ &= \frac{\partial u}{\partial t}(x_i, t_n) + a\frac{\partial u}{\partial x}(x_i, t_n) - b\frac{\partial^2 u}{\partial^2 t}(x_i, t_n) + \mathcal{O}(\Delta t + \Delta x^2) \end{aligned}$$

Le erreur ci-dessus est dans le "standard" mentionné dans le Remarque 2.2.5 du polycopié, c'est à dire, pour une fonction régulière qui n'est pas solution de notre problème, la limite de l'erreur de troncature n'est pas nulle. Alors c'est schéma est consistante et d'ordre 1 en temps et 2 en espace. Pour le schéma

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{2\Delta x} - b\frac{u_{i+1}^{n+1} + u_{i-1}^{n+1} - 2u_i^{n+1}}{\Delta x^2} = 0, \tag{4}$$

on fait le développement au tour du point (x_i, t_{n+1}) et on trouve

$$\begin{aligned} u(x_{i\pm1}, t_{n+1}) = u(x_i, t_{n+1}) \pm \Delta x\frac{\partial u}{\partial x}(x_i, t_{n+1}) + \frac{1}{2}\Delta x^2\frac{\partial^2 u}{\partial x^2}(x_i, t_{n+1}) \pm \frac{1}{6}\Delta x^3\frac{\partial^3 u}{\partial x^3}(x_i, t_{n+1}) \\ + \frac{1}{24}\Delta x^4\frac{\partial^4 u}{\partial^4 x}(x_i, t_{n+1}) + \mathcal{O}(\Delta x^5), \end{aligned}$$

$$u(x_i, t_n) = u(x_i, t_{n+1}) - \Delta t\frac{\partial u}{\partial t}(x_i, t_{n+1}) + \frac{1}{2}\Delta t^2\frac{\partial^2 u}{\partial t^2}(x_i, t_{n+1}) + \mathcal{O}(\Delta t^3), \text{ et finalement}$$

$$\begin{aligned} E &= \frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{2\Delta x} - b\frac{u_{i+1}^{n+1} u_{i-1}^{n+1} - 2u_i^{n+1}}{\Delta x^2} \\ &= \frac{\partial u}{\partial t}(x_i, t_{n+1}) + a\frac{\partial u}{\partial x}(x_i, t_{n+1}) - b\frac{\partial^2 u}{\partial^2 t}(x_i, t_{n+1}) + \mathcal{O}(\Delta t + \Delta x^2). \end{aligned}$$

Par le même argument du schéma antérieur, ce schéma est consistant d'ordre 1 en temps et 2 en espace.

III.1.2. On commence par le schéma explicite 3. Pour la stabilité l^∞, on va trouver les conditions CFL pour que le schéma vérifie le principe du maximum discret.

Soit k et l tel que

$$u_k^{n+1} = M = \max_j u_j^{n+1} \quad \text{et} \quad u_l^{n+1} = M = \min_j u_j^{n+1}$$

Notons que M est positif ou nul et m est négatif ou nul. On veut trouver les conditions pour que

$$M \leq \max(0, \max_j u_j^n) \tag{5}$$

$$\text{et} \min(0, \min_j u_j^n) \leq m. \tag{6}$$

On commence l'analyse par l'inégalité 5. Elle est trivialement vérifiée si $M = 0$. Alors, on analyse le cas où $M \neq 0$. Le maximum de u_j^{n+1} pour tout $j \in \{i_{\max} - 1,\, i_{\max} + 1\}$ est atteint en un élément $k \in \{i_{\max},\, i_{\max}\}$. Alors

$$\frac{M - u_k^n}{\Delta t} + a\frac{u_{k+1}^n - u_{k-1}^n}{2\Delta x} - b\frac{u_{k+1}^n + u_{k-1}^n - 2u_k^n}{\Delta x^2} \leq 0$$

soit

$$M \leq \left(-\frac{a\Delta t}{2\Delta x} + \frac{b\Delta t}{\Delta x^2}\right) u_{k+1}^n + \left(1 - \frac{2b\Delta t}{\Delta x^2}\right) u_k^n + \left(\frac{a\Delta t}{2\Delta x} + \frac{b\Delta t}{\Delta x^2}\right) u_{k-1}^n$$

Pour que le schéma de droite soit une combinaison convexe de u_n et l'inégalité 5 soit vérifiée on les conditions ci-dessus.

$$\Delta x \leq \frac{2b}{a} \quad , \quad \Delta x \geq -\frac{2b}{a} \quad \text{et} \quad \Delta t \leq \frac{\Delta x^2}{2b}.$$

Pour vérifiée l'inégalité 6, on remplace u^n par $-u^n$ et M par $-m$.

Pour la stabilité du schéma 4 en l^∞, on voit que c'est un schéma implicite qui vérifie le principe du maximum. Alors, il est stable.

Pour la stabilité du schéma 3 en l^2, on a, pour la transformé de Fourier,

$$\hat{u}^{n+1}(k) = \left(1 - \frac{a\Delta t}{2\Delta x}(e^{2i\pi k\Delta x} - e^{-2i\pi k\Delta x}) + \frac{b\Delta t}{\Delta x^2}(e^{2i\pi k\Delta x} + e^{-2i\pi k\Delta x} - 2)\right) \hat{u}^n(k)$$

Autrement dit

$$\hat{u}^{n+1}(k) = A(k)\hat{u}^n(k) = A(k)^{n+1}\hat{u}^0(k) \text{ avec } A(k) = \left(1 - \frac{ia\Delta t}{2\Delta x}\sin(2\pi k\Delta x) + \frac{4b\Delta t}{\Delta x^2}\sin^2(\pi k\Delta x)\right)$$

Comme $\|A(k)\| \leq 1$ pour tout mode de Fourier k, la formule de Plancherel permet de conclure à la stabilité l^2 du schéma. (Voir polycopié page 36).

III.2. Schéma implicite

III.2.1. On peut récrire le schéma implicite comme

$$u_i^n = \left(\frac{a\Delta t}{2\Delta x} - \frac{b\Delta t}{\Delta x^2}\right) u_{i+1}^{n+1} + \left(1 + \frac{2b\Delta t}{\Delta x^2}\right) u_i^{n+1} + \left(-\frac{a\Delta t}{2\Delta x} - \frac{b\Delta t}{\Delta x^2}\right) u_{i-1}^{n+1}.$$

On pose $\gamma = \frac{a\Delta t}{2\Delta x}$ et $\delta = \frac{b\Delta t}{\Delta x^2}$. Alors pour $U^n = MU^{n+1}$, où $U^n = \begin{pmatrix} u_{-i_{\max}}^n \\ \vdots \\ u_{i_{\max}}^n \end{pmatrix}$, on trouve

$$M = \begin{pmatrix} 1+2\delta & \gamma-\delta & & & \\ -(\gamma+\delta) & 1+2\delta & \gamma-\delta & & \\ & -(\gamma+\delta) & \ddots & \ddots & \\ & & \ddots & \ddots & \gamma-\delta \\ & & & -(\gamma+\delta) & 1+2\delta \end{pmatrix}$$

III.2.2. On définit $M^{\mathcal{S}} = \frac{1}{2}(M + M^T)$

$$M^{\mathcal{S}} = \begin{pmatrix} 1+2\delta & -\delta & & & \\ -\delta & 1+2\delta & -\delta & & \\ & -\delta & \ddots & \ddots & \\ & & \ddots & \ddots & -\delta \\ & & & -\delta & 1+2\delta \end{pmatrix} \tag{7}$$

D'après [1], M est positive définit si et seulement si $M^{\mathcal{S}}$ y est. D'après [2], comme $\|1+2\delta\| > \|-\delta\| + \|-\delta\|$, $M^{\mathcal{S}}$ est définit positive, alors M est définit positive. Ce qu'implique que M est inversible.

III.2.3. La solution a été calculé e la figure crée par le code disponible sur "problem5.py".

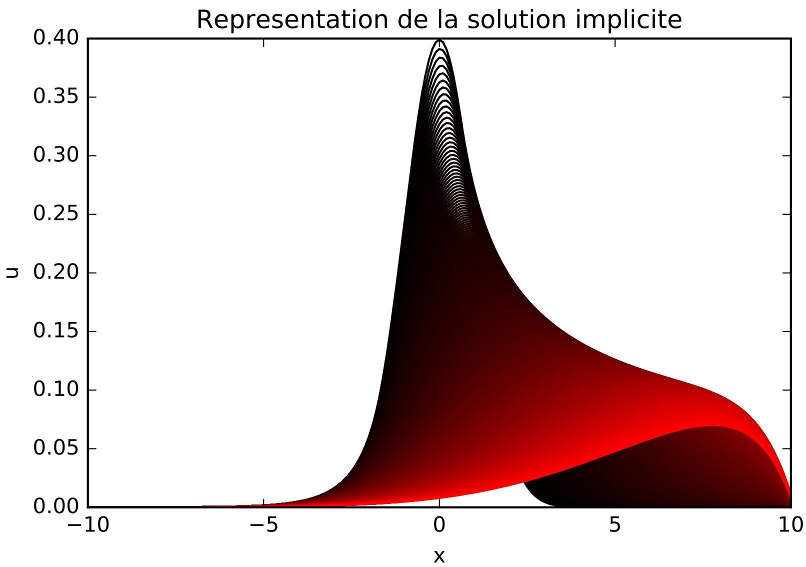

Figure 10 – Représentation de la solution implicite pour $\Delta x = 0.1$, $\Delta t = 10^{-4}$ et $n = 10^5$. Les fonctions sont superposées et l'intensité de la couleur rouge augmente avec le temps.

IV. Application à l'équation de la chaleur non linéaire

IV.1. Schéma implicite linéaire

IV.1.1. On peut récrire le schéma comme

$$u_i^n = \left(-\frac{b\Delta t}{\Delta x^2}\right) u_{i+1}^{n+1} + \left(1 + 2\frac{b\Delta t}{\Delta x^2}\right) u_i^{n+1} + \left(-\frac{b\Delta t}{\Delta x^2}\right) u_{i-1}^{n+1} + u_i^{n+1}(u_i^n)^3\Delta t.$$

Ainsi, on peut écrire $U^n = (A + \Delta t \text{ diag}((u^n_{-i_{\max}})^3, \ldots, (u^n_{i_{\max}})^3))U^{n+1}$, où $U^n = \begin{pmatrix} u^n_{-i_{\max}} \\ \vdots \\ u^n_{i_{\max}} \end{pmatrix}$,

$\delta = \frac{b\Delta t}{\Delta x^2}$ et$A = \begin{pmatrix} 1+2\delta & -\delta & & & \\ -\delta & 1+2\delta & -\delta & & \\ & -\delta & \ddots & \ddots & \\ & & \ddots & \ddots & -\delta \\ & & & -\delta & 1+2\delta \end{pmatrix}$.

IV.1.2. La solution a été calculé e la figure crée par le code disponible sur "problem5.py".

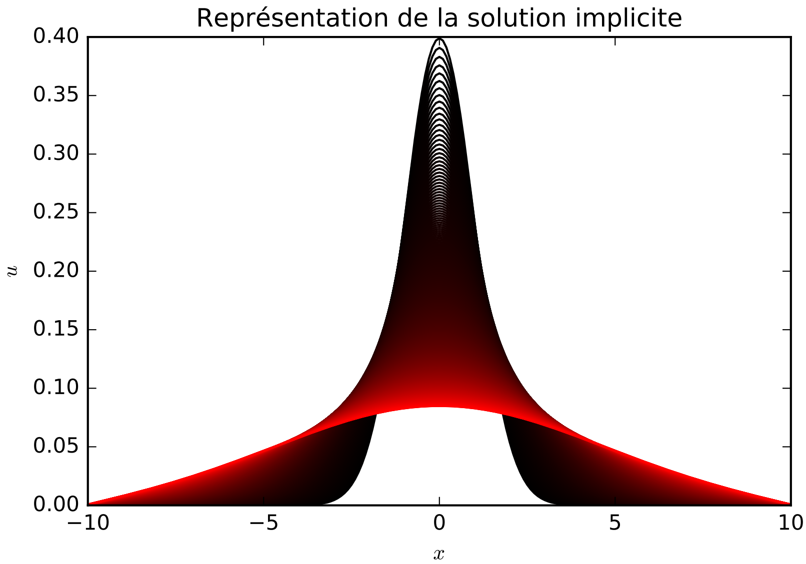

FIGURE 11 – Représentation de la solution implicite pour $\Delta x = 0.1$, $\Delta t = 10^{-4}$ et $n = 10^5$. Les fonctions sont superposées et l'intensité de la couleur rouge augmente avec le temps.

IV.2. Schéma pour l'équation stationnaire

IV.2.1. On va écrire la fonction G comme

$$G_i(U) = -b\frac{u_{i+1} + u_{i-1} - 2u_i}{\Delta x^2} + u_i^4 - Q_i, \ \forall i \in [-i_{\max}, i_{\max}]$$

de façon que pour $G = (G_{-i_{\max}}, \ldots, G_{i_{\max}})$, on a $G(U) = 0$.

IV.2.2. On sait que $[DG(\beta)]_{ij} = \frac{\partial G_i(\beta)}{\partial \beta_j}$. Pour Q=1 et $\theta = \frac{b}{\Delta x^2}$, on a

$$DG(U) = \begin{pmatrix} 2\theta + 4u^3_{-i_{\max}} & -\theta & & & \\ -\theta & 2\theta + 4u^3_{-i_{\max}+1} & -\theta & & \\ & -\theta & \ddots & \ddots & \\ & & \ddots & \ddots & -\theta \\ & & & -\theta & 2\theta + 4u^3_{i_{\max}} \end{pmatrix}.$$

Pour l'algorithme de Newton on aura

$$U^{k_1} = U^k - (DG(U^k))^{-1}G(U^k), \text{ou alors}$$

$$G(U^k) = DG(U^k)(U^k - U^{k+1}).$$

Le choix de la fonction initiale est arbitraire et on peut utiliser par exemple une parabole $u_0(x) = 1 - \frac{x^2}{100}$. On voit la convergence vers une fonction proche que 1 au milieu et avec une courbature qui compense les conditions de Dirichlet aux bords.

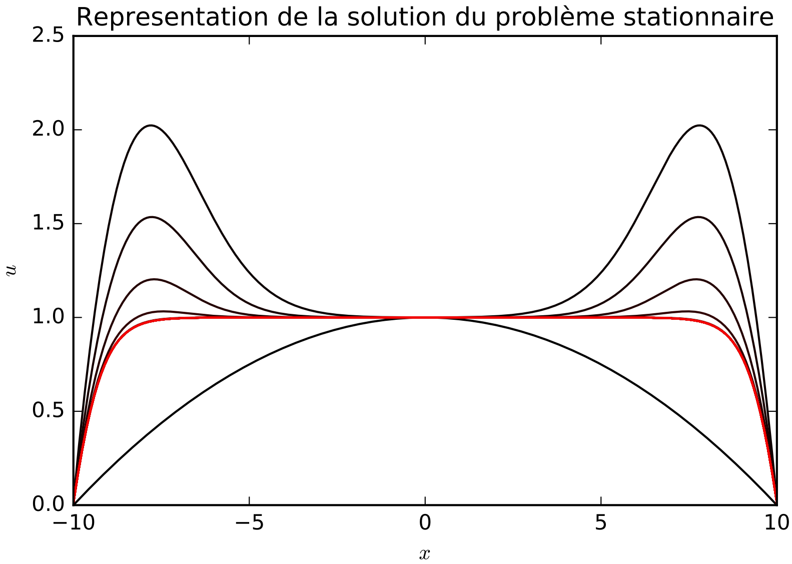

FIGURE 12 – Représentation de la solution du problème stationnaire avec $\Delta x = 0.01$ et $n = 20$ itérations. Les fonctions sont superposées et l'intensité de la couleur rouge augmente avec le temps.

Le calcule et la figure ont été faits par le code "problem6.py".

Références

[1] Weisstein, Eric W. *Positive Definite Matrices*. From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/PositiveDefiniteMatrix.html`

[2] Milica Anđelić and C. M. da Fonseca. *Sufficient Conditions for Positive Definiteness of Tridiagonal Matricies Revisited*. `http://www.mat.uc.pt/~cmf/papers/tri_positive_definite_revisited.pdf`