Résolution des systèmes non-linéaires

Dichotomie

Soit l'équation

$$2 - x = \exp(x). \tag{0.1}$$

1. Montrer que cette équation admet une unique solution dans $[0, 1]$.
2. Calculer un majorant $n_{\max}$ du nombre d'itérations de la méthode de dichotomie qui assure une précision de $\varepsilon = 10^{-10}$.
3. Écrire un programme qui calcule cette solution par la méthode de dichotomie avec une précision $\varepsilon = 10^{-10}$. Combien d'itérations n sont-elles nécessaires ? Vérifier que $n \leq n_{\max}$ (!).

Point fixe

On souhaite maintenant résoudre l'équation (0.1) par la méthode du point fixe.

1. Soit $\lambda \neq 0$ et
$$g_\lambda(x) = \lambda((2 - x)\exp(-x) - 1) + x.$$
Montrer que la solution de (0.1) est aussi un point fixe de g_λ.
2. On choisit $\lambda = 1/4$. Montrer que g_λ est une application contractante de $[0, 1]$ dans $[0, 1]$. Que vaut la constante de contraction ?
3. Calculer un majorant $n_{\max}$ du nombre d'itérations de la méthode du point fixe qui assure une précision de $\varepsilon = 10^{-10}$.
4. Écrire un programme qui calcule la solution de (0.1) par la méthode du point fixe avec une précision $\varepsilon = 10^{-10}$ en partant de $x_0 = 0$. Combien d'itérations n sont-elles nécessaires ? Vérifier que $n \leq n_{\max}$.

Newton

Écrire un programme qui calcule la solution de (0.1) par la méthode de Newton avec une précision $\varepsilon = 10^{-10}$ en partant de $x_0 = 0$. Combien d'itérations n sont-elles nécessaires ?

Newton en dimension > 1

Le but de cet exercice est de construire une méthode pour trouver les racines d'un polynôme de la forme

$$P(x) = a_N + a_{N-1}x + \cdots + a_1 x^{N-1} + x^N.$$

(Aller directement à la question 7 pour le TP, le reste des questions est facultatif pour ceux qui veulent une justification de la méthode). On suppose que P a N racines réelles distinctes notées $X = (x_1 \cdots x_N)$. Pour $k = 1 \cdots N$, on note $\Sigma_k(x_1, \cdots x_N)$ la somme de tous les produits de k racines différentes. Par exemple $\Sigma_N = x_1 x_2 \cdots x_N$ et $\Sigma_1 = x_1 + x_2 + \cdots x_N$.

1. Combien y a-t-il de termes dans la somme Σ_k ? Montrer que $\forall k = 1 \cdots N$,
$$\Sigma_k(x_1, \cdots x_N) = (-1)^k a_k.$$

2. Trouver les racines de P est donc équivalent à trouver les zéros de la fonction F : $\mathbb{R}^N \to \mathbb{R}^N$ définie par

$$F_k(x_1, x_2, \cdots x_N) = (-1)^k \Sigma_k(x_1, \cdots x_N) - a_k.$$

Rappeler le principe de la méthode de Newton pour résoudre

$$F(X) = 0. \tag{0.2}$$

3. Montrer que

$$\sum_{k=1}^{N} (-1)^k \frac{\partial}{\partial x_i} \Sigma_k(X) x^{N-k} = - \prod_{1 \leq k \leq N, k \neq i} (x - x_i)$$

4. On note Δ la matrice diagonale $N \times N$ définie par

$$\Delta_{ii} = \prod_{k \neq i} (x_k - x_i)$$

et V la matrice de Vandermonde $N \times N$

$$V_{ij} = x_i^{N-j}.$$

Montrer que

$${}^t F'(X) {}^t V = -\Delta.$$

5. En déduire que

$$F'(X)^{-1} = -\Delta^{-1} V.$$

6. Montrer que la méthode de Newton appliquée à (0.2) consiste à choisir des valeurs initiales distinctes $(x_1^{(0)}, \cdots x_N^{(0)})$ puis à construire les suites $x_i^{(n)}$ définies par

$$x_i^{(n+1)} = x_i^{(n)} - \frac{P(x_i^{(n)})}{\prod_{k \neq i} (x_i^{(n)} - x_k^{(n)})}.$$

7. Tester votre méthode numériquement sur divers polynômes, y compris des polynômes avec des racines complexes.

Méthode d'Euler implicite

Soit α un réel > 0 et le problème de Cauchy

$$x'(t) = -x(t)^2, \quad x(0) = \alpha, \quad t \in [0, T].$$

1. Calculer la solution exacte de ce problème. Vérifier que $x(t) \to 0$ quand $t \to \infty$ et que pour tout $t \in [0, T]$, $x(t) > 0$.
2. Pour résoudre cette équation différentielle, on considère la méthode d'Euler implicite. Programmer cette méthode pour une discrétisation avec des pas de temps donnés par

$$t_k = (k-1)\Delta t, \quad \Delta t = \frac{T}{N}, \quad k = 1 \ldots N+1.$$

Le problème implicite sera résolu à chaque pas de temps par la méthode de Newton. La méthode de Newton sera initialisée par la solution du pas de temps précédent. Dans un premier temps, on fixera le nombre d'itérations de la méthode de Newton à $n_{\max} = 10$.

3. Améliorer l'algorithme en ajoutant un critère d'arrêt pour la méthode de Newton.
4. Tester la méthode pour plusieurs valeurs de α et de N. Vérifier que cette méthode est inconditionnellement stable, contrairement à la méthode d'Euler explicite.