TP Licence. M2AO 2008/2009

Introduction à Matlab: programmation de schémas pour les équations différentielles ordinaires

Définition et tracé de fonctions

i) Construire f et tracer le graphe de f pour:

- $f : [0,1] \to \mathbb{R}$, fonction en escalier,
- $f : [0,1] \to \mathbb{R}$, continue, non C^1, affine par morceaux,
- $f : [0,1] \to \mathbb{R}$, régulière,
- $f : [0,1] \to \mathbb{R}$, régulière, périodique (les valeursde f et de ses dérivées sont identiques aux extrmits).

ii) Tracer sur un même graphique, le graphe des fonctions $P_k : x \mapsto x^k$ pour $k = 0..3$.

Méthode de Newton

On rappelle que la méthode de Newton pour résoudre une équation non linéaire du type $f(x) = 0$ où $f : \Omega \to \mathbb{R}$ est une fonction de classe $\mathcal{C}^1$ sur un ouvert $\Omega \subset \mathbb{R}^n$ est donné par l'algorithme suivant

$$x_{n+1} = x_n - df(x_n)^{-1} f(x_n),$$

où $df(x_n)^{-1}$ est l'inverse de la matrice jacobienne de f en x_n. On peut montrer que cet alogorithme converge vers $\overline{x}$ solution de $f(x) = 0$ sous l'hypothèse $df(\overline{x})$ inversible et $|\overline{x} - x_0|$ suffisammet petit.

Exercice. i) Programmer un algorithme de Newton permettant de calculer la plus grande racine du polynôme $f(x) = x^5 + 7x^4 - 12x^3 + 3x^2 + 4x - 1$.

Schemas d'Euler pour les équations différentielles

On souhaite résoudre numériquement l'équation ou le système différentiel

$$\frac{dx}{dt} = f(t,x), \quad x(0) = x^0.$$

On suppose que f vérifie les hypothèses du thèorème de Cauchy Lipschitz et la solution x est définie sur un intervalle $I = [0, T[$. On se donne une subdivision $(t_n)_{n=0,..,N}$ tel que $t_0 = 0$ et $t_N = T$. On peut définir alors le schéma d'Euler explicite par une suite X_n tel que

$$X_{n+1} = X_n + \delta_n f(t_n, X_n), \forall n \geq 0.$$

et $X_0 \approx x^0$ (en général on ne part par *exactement* de la donnée initiale théorique à cause d'erreur d'arrondis, ou parce qu'on n'en connait qu'une valeur approchée). Ici

$\delta_n = t_{n+1} - t_n$. Pour simplififer, on suposera $\delta_n = \delta$, un pas de temps constant. Le schema d'Euler est *convergent* c'est à dire que

$$\lim_{\delta \to 0} \max_{n=0,...N} |X_n - x(t_n)| = 0.$$

La suite X_n est donc une approximation de la solution exacte x au point t_n. De la même manière, on peut définir le scéma d'Euler implicite sous la forme

$$X_{n+1} = X_n + \delta_n \, f(t_{n+1}, X_{n+1}), \forall n \geq 0$$

C'est encore un schéma convergent mais il faut cette fois ci résoudre un problème non linéaire de la forme

$$g(X_{n+1}) = X_{n+1} - X_n - \delta_n \, f(t_{n+1}, X_{n+1}) = 0,$$

pour obtenir X_{n+1}. En fait ces deux schémas sont *d'ordre 1*,, c'est à dire qu'il existe une constant C (qui dépend de la solution exacte x) tel que

$$\max_{n=0,..,N} |x(t_n) - X_n| \leq C\delta,$$

si on a choisi une subdivision uniforme $\delta_n = \delta$. On se propose d'illustrer ces différents points.

i) Résoudre l'équation $x'(t) = -x(t)$, $x(0) = 2$ sur l'intervalle de temps $t = [0, 1]$.

ii) Programmer les schémas d'Euler explicite et Euler implicite pour résoudre cette équation différentielle. On choisira successivement des pas d'espaces $\delta = (0.1)^k$ avec $k \in 1, ..., 8$ pour illustrer graphiquement la convegence.

iii) Pour chaque schéma, représenter l'erreur dans un diagramme log-log. On rappelle que pour un pas δ donné, on a

$$\text{erreur}(\delta) = \max_{n=0,..,N} |x(t_n) - X_n|,$$

où $t_n = n\,\delta$. À l'aide de polyfit calculer la pente des droites obtenues pour chaque schéma et faire le lien avec l'ordre du schéma.

iv) On définit le schéma de Crank Nicolson par la suite

$$X_{n+1} = X_n + \frac{\delta}{2}\big(f(t_n, X_n) + f(t_{n+1}, X_{n+1})\big), \quad t_n = n\delta.$$

Utiliser ce schéma pour résoudre l'équation différentielle initiale et illustrer graphiquement la convergence. Tracer dans un diagramme log-log l'erreur du schéma et calculer la pente de la droite obtenue par la méthode des moindres carrés à l'aide de la commande polyfit. En déduire que le schéma est d'ordre 2. Retrouver théoriquement ce résultat.