Université Pierre et Marie Curie
LM383
Année universitaire 2007-2008

Equations Différentielles,
méthodes de résolution
T. Capricelli et **D. Yakoubi**

TP 2 : Intégration numérique et résolution d'équations non linéaires

1. Intégration numérique.

- Ecrire un programme Scilab permettant d'intégrer la fonction sin sur $[0, \pi]$ par la méthode des trapèzes en fonction d'un pas de discrétisation $h = \pi/n$. Même question avec la méthode de Simpson.
- Comparer les résultats avec la valeur exacte pour différentes valeurs de h.
- Estimer numériquement, en représentant graphiquement le logarithme de l'erreur pour différentes valeurs de $\log(h)$, les ordres de convergence des deux méthodes. Vérifier qu'on retrouve bien les ordres théoriques de convergence.

2. Nous allons étudier le comportement de la suite $x_n = f(x_{n-1})$ avec $f(x) = x^2 + c$

- Ecrire une fonction *Scilab recevant en argument*
  - *c le paramêtre de la suite*
  - *nmax le nombre de termes de la suite que l'on va calculer*
  - x_0 *la valeur du premier terme de la suite*

  et représentant la courbe $(n; x_n)$.
- *Ecrire un programme Scilab réalisant les calculs suivants*
  - *Pour* N_c *valeurs de c décrivant l'intervalle* $[-2, 1/4]$ *faire*
    - *calculer les* N_{init} *premiers termes de la suite* xu_n *avec* $x_0 = 0$
    - *calculer les* N_{suiv} *termes suivants en les sauvant dans un tableau de taille* N_{suiv}.
    - *représenter dans une fenêtre graphique* $[-2, -2, 1/4, 2]$ *les points de la suite* x_n *sur la verticale de c*
  - *faire varier* N_{init}, *et* N_{suiv} *pour avoir une figure stable.*

2. On va maintenant étudier le comportement de l'algorithme de Newton pour résoudre un système d'équations non linéaires

$$F(X) = 0 \quad \text{avec} \quad X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad \text{et} \quad F \begin{pmatrix} f_1(X) \\ \vdots \\ f_n(X) \end{pmatrix}$$

On rappelle l'algorithme:

- $X = X_0$,$G = J(F)(X)$ avec $J(F)_{ij} = \frac{\partial f_i}{\partial x_j}$
- Tant que $\|G\| > \varepsilon$ et $iter < iter_{max}$ faire
  - trouver DX tel que $G.DX = F(X)$
  - $X = X - DX$
  - $G = J(F)(X)$
  - $iter = iter + 1$

- Ecrire une fonction *Scilab* **newton** *recevant en argument d'entrée*
  - x_0 *le vecteur contenant la solution initiale*
  - f *et* $jacf$ *le nom des fonctions Scilab calculant la fonction et son jacobien.*
  - ε *la tolérance.*
  - $iter_{max}$ *le nombre maximum d'iterations.*

  newton *renvoit la solution de l'equation* $f(x) = 0$ *si elle existe ou affiche un message d'erreur si le nombre maximal d'itérations est atteint avant la convergence.*

- *Tester* **newton** *sur l'exemple suivant:*

$$y \ln y - x \ln x = 3$$
$$x^4 + xy + y^3 = a$$

- *Comparer avec la fonction* **fsolve** *pré-programmée dans scilab, décrite ci-dessous qui met en oeuvre la méthode de Powell*
  - $[x[, v[, info]]] = fsolve(x0, fct[, fjac][, tol])$
  - *PARAMETRES*
    - *x0 : vecteur reel (valeur initiale).*
    - *fct : fonction externe (définie par deff, ou dans un fichier chargé préalablement)*
    - *fjac : fonction externe calculant le jacobien de la fonction fct*
    - *tol : réel, tolérance. La fin a lieu quand l'algorithme estime que l'erreur relative entre x et la solution est au plus tol (10^{-10} étant la valeur par défaut)*
    - *x : vecteur réel contenant la valeur finale calulée par l'algorithme qui annule fct*
    - *v : vecteur réel contenant la valeur de fct en x*
    - *info : indicateur de terminaison*
      0 : mauvais parametres d'entrée
      1 :l'algorithme estime que l'erreur relative entre x et la solution est au plus tol
      2 : le nombre maximum d'appels à la fonction fct a été atteint
      3 : tol est trop petit, ca ne sert à rien de continuer.
      4 : l'algorithme ne converge pas.

- *On va maintenant étudier le comportement de l'algorithme de Newton pour résoudre l'équation* $x^3 = 1$ *dans* $\mathbb{C}$. *L'expérience consiste à faire varier la condition initiale* x_0 *et à regarder vers quelle racine Newton converge parmi* $\{r_1 = 1, r_2 = e^{2i\pi/3}, r_3 = e^{-2i\pi/3}\}$.
  - *Ecrire la fonction* $f(x) = x^3 - 1$ *et son jacobien comme fonctions de* $\mathbb{R}^2 \to \mathbb{R}^2$ *et* $\mathbb{R}^2 \to \mathbb{R}^{2\times 2}$.
  - *Pour* x_0 *parcourant le carré* $[-1, 1] \times [-1, 1]$ *par* nx *increments* $d_x = 2/n_x$ *sur l'axe réel et* n_y *incréments* $d_y = 2/n_y$ *sur l'axe imaginaire, calculer la solution de* $x^3 = 1$ *par l'algorithme de Newton.*
  - *Pour* $x_0 = kd_x + ild_y$ *on stocke*

$$racine(k, l) = \begin{cases} 1 & \text{si Newton a convergé vers } 1 \\ 2 & \text{si Newton a convergé vers } = e^{2i\pi/3} \\ 3 & \text{si Newton a convergé vers } = e^{-i\pi/3} \\ 4 & \text{si Newton n 'a pas convergé} \end{cases}$$

  - *Utiliser la fonction Matplot pour représenter la matrice racine*
  - *Recommencer l'expérience en faisant varier* x_0 *dans un carré plus petit* $[-0.1, 0.1] \times [-0.1, 0.1]$ *par exemple.*