MT94/P24/TD - Problèmes non linéaires (1)

NB : la section 1 a pour but de vous faire découvrir comment fonctionnent les méthodes de point fixe et méthodes de Newton. À partir de la Section 2, nous utiliserons la macro `fsolve()` de Scilab.

1 Résolution d'une équation à une inconnue $f(x) = 0$

1.1 Ordre de convergence

On cherche à résoudre numériquement l'équation

$$x^2 = 2. \tag{1}$$

1. Donner un intervalle $[a, b]$ assez petit dans lequel se situe la solution positive $\hat{x}$ de (1). Dans la suite on considérera $x_0 = (a + b)/2$.

2. L'équation (1) peut s'écrire de façon équivalente $f(x) = 0$ ou $x = g(x)$ avec

$$f(x) = x^2 - 2,\ g(x) = \frac{x+2}{x+1}.$$

On va comparer plusieurs méthodes itératives pour trouver $\hat{x}$:
(a) méthode de dichotomie,
(b) méthode de point fixe

$$x_{k+1} = g(x_k), \quad x_0 \text{ donné},$$

(c) méthode de Newton

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad x_0 \text{ donné},$$

(d) méthode de la sécante

$$x_{+1} = x_k - \frac{f(x_k)}{f(x_k) - f(x_{k-1})}(x_k - x_{k-1}), \quad x_0,\, x_1 \text{ donnés}\ .$$

L'ordre de convergence $p \geq 1$ d'une méthode est défini par

$$\lim_{k \to +\infty} \frac{|x_{k+1} - \hat{x}|}{|x_k - \hat{x}|^p} = \alpha > 0.$$

En programmant les différentes méthodes en Scilab, retrouvez expérimentalement que (a) et (b) sont d'ordre 1, (c) d'ordre 2 et (d) d'ordre le nombre d'or ! Le test d'arrêt sera défini par $|x_{k+1} - x_k| < \varepsilon$ avec $\varepsilon = 10^{-12}$. Pour chaque méthode, afficher le nombre d'itérations pour atteindre l'arrêt. On tracera $\log_{10} |x_k - \hat{x}|$ en fonction de k pour chaque méthode.

1.2 Division sans effectuer de ... division :)

En appliquant la méthode de Newton à la fonction f définie par

$$f(x) = \frac{1}{x} - a$$

où $a \neq 0$, montrer que l'on peut calculer $\frac{1}{a}$ sans effectuer de division. Sachant que l'on peut toujours se ramener au cas où $a \in [1, 2[$ sans perte de précision (expliquer pourquoi), donner une borne supérieure du nombre d'itérations nécessaires pour approcher $\frac{1}{a}$ à la précision machine. Expérimenter le résultat obtenu en écrivant un programme Scilab approchant $\frac{1}{\pi}$.
NB : c'est comme cela que les processeurs d'aujourd'hui effectuent les divisions (ils ne savent faire de base que des additions et des multiplications).

2 Applications

2.1 La macro `fsolve()` de Scilab

La macro `fsolve()` de Scilab permet de résoudre des systèmes d'équations non linéaires avec une méthode de Newton ou de quasi-Newton. Par exemple, pour résoudre $x^2 - 2 = 0$, on fait :

```
function y = f(x)
        y = x^2 - 2;
endfunction

function d = fprime(x)
        d = 2*x;
endfunction

x =  1.5 // Init
x = fsolve(x0, f, fprime)
```

2.2 L'étoile de la mort (surfaces implicites)

Dans un premier temps, analysez et exécutez le code Scilab suivant :

```
// f(x,y,z) = 1 - x^2 - y^2 - z^2  = 0
N = 1200
theta = 2*%pi * rand(N,1)
phi   = -%pi/2 + %pi*rand(N,1)
x = cos(theta).*cos(phi)
y = sin(theta).*cos(phi)
z = sin(phi)
scatter3d(x, y, z)
mtlb_axis('equal')
```

Dans la suite, on définit trois fonctions f_1, f_2 et f de $\mathbb{R}^3$ dans $\mathbb{R}$:

$$f_1(x,y,z) = 1 - x^2 - y^2 - z^2, \quad f_2(x,y,z) = 1 - (x-0.9)^2 + (y-0.9)^2 - (z-0.9)^2$$

et

$$f(x,y,z) = \min\left(f_1(x,y,z),\ -f_2(x,y,z)\right).$$

Les points (x,y,z) tels que $f_1(x,y,z) = 0$ décrivent la sphère de rayon 1 et de centre 0, ceux de $f_2(x,y,z) = 0$ la sphère de rayon 1 et de centre (0.9, 0.9, 0.9). Essayez d'imaginer quelle est la surface décrite par l'équation implicite

$$f(x,y,z) = 0.$$

Programmer d'abord les 3 fonctions `Scilab`:

```
function fout = f1(x,y,z)
fout = // A COMPLETER
endfunction

function fout = f2(x,y,z)
fout = // A COMPLETER
endfunction

function fout = f(x,y,z)
fout = // A COMPLETER
endfunction
```

On considère un ensemble de N points (x_i^0, y_i^0, z_i^0) calculés par

$$\begin{aligned} x_i^0 &= 0.5\ \cos(\theta_i)\ \cos(\phi_i), \\ y_i^0 &= 0.5\ \sin(\theta_i)\ \cos(\phi_i), \\ z_i^0 &= 0.5\ \sin(\phi_i), \end{aligned}$$

où les angles θ_i et ϕ_i sont tirés selon les lois uniformes

$$\theta \sim \mathcal{U}(0, 2\pi), \quad \phi \sim \mathcal{U}(-\frac{\pi}{2}, \frac{\pi}{2}).$$

Méthode de faisceaux. Dans $\mathbb{R}^3$, la droite D_i passant par l'origine et le point (x_i^0, y_i^0, z_i^0) a (par exemple) pour équations (vérifier)

$$\left\{ \begin{array}{l} x_i^0\, y - y_i^0\, x = 0, \\ x_i^0\, z - z_i^0\, x = 0. \end{array} \right.$$

Pour chaque (x_i^0, y_i^0, z_i^0), on va chercher l'intersection de la droite D_i avec la surface d'équation implicite $f(x, y, z) = 0$ en résolvant le système

$$\left\{ \begin{array}{l} f(x, y, z) = 0, \\ x_i^0\, y - y_i^0\, x = 0, \\ x_i^0\, z - z_i^0\, x = 0 \end{array} \right. \quad \Leftrightarrow F(X;\, x_i^0, y_i^0, z_i^0) = 0, \quad X = (x, y, z)$$

pour trouver un point sur la surface implicite décrite par $f(x, y, z) = 0$. On utilisera la macro `fsolve()` de Scilab pour résoudre le système d'équations, on prendra bien sûr $X_i^0 = (x_i^0, y_i^0, z_i^0)$ comme initialisation. On prendra $N = 5000$ (5000 systèmes à résoudre, donc pour tous les points). Avec `scatter3d()`, afficher tous les points ainsi trouvés.

2.3 Recherche d'un couple valeur propre-vecteur propre

Soit A une matrice carrée de taille n. La recherche de couples propres (x, λ) solutions de $Ax = \lambda x$ peut se faire par résolution d'un système algébrique $F(X) = 0$. En cherchant un vecteur propre x de norme 1, on cherche alors la paire $X = (x, \lambda) \in \mathbb{R}^n \times \mathbb{R}$ telle que

$$\left\{ \begin{array}{l} Ax - \lambda x = 0, \\ \|x\|^2 - 1 = 0. \end{array} \right.$$

Ecrire une fonction `[y] = Fvp(X)` qui calcule $y = F(X)$ ainsi qu'une fonction `[J] = Fvpjac(X)` qui calcule la matrice jacobienne associée $J = D_X F(X)$ pour la matrice

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}.$$

Utiliser ensuite la macro `fsolve()` pour trouver une solution de $F(X) = 0$. On utilisera la donnée initiale suivante : $X^0 = ((1, 1, 1), 1)$. A l'aide de la fonction `spec()` de Scilab, vérifier que la solution est bien un couple valeur propre-vecteur propre de A.

2.4 Formules d'intégration numérique sur un triangle

On considère le triangle unité $T = ((0, 0), (0, 1), (1, 0))$. On souhaite construire une formule d'intégration numérique (formule de quadrature) sur le triangle T, i.e.

$$I(f) = \sum_{i=1}^{3} \omega_i\, f(P_i) \approx \iint_T f(x, y)\, dx\, dy$$

avec $P_i(x_i, y_i) \in T$ et $\omega_i \in \mathbb{R}$, $i = 1, 2, 3$ qui soit exacte au moins pour les polynômes de degré inférieur ou égal à 2 en les variables x et y. Comme $\iint_T 1\, dxdy = \frac{1}{2}$, on doit avoir

$$\omega_1 + \omega_2 + \omega_3 = \frac{1}{2}.$$

Les points d'intégration P_1, P_2, P_3 sont inconnus. Pour des raisons de symétrie sur le triangle unité, on impose $x_1 = y_1$ et $x_2 = y_3$, $x_3 = y_2$ et $\omega_2 = \omega_3$. En y ajoutant les propriétés d'exactitude des polynômes de degré 1 et 2, on obtient le système algébrique à résoudre

$$\begin{array}{l}
\omega_1\, x_1 + \omega_2\, x_2 + \omega_3\, x_3 = \frac{1}{6} \quad \left(= \iint_T x\, dxdy\right), \\
\omega_1\, y_1 + \omega_2\, y_2 + \omega_3\, y_3 = \frac{1}{6} \quad \left(= \iint_T y\, dxdy\right), \\
\omega_1\, x_1^2 + \omega_2\, x_2^2 + \omega_3\, x_3^2 = \frac{1}{12} \quad \left(= \iint_T x^2\, dxdy\right), \\
\omega_1\, y_1^2 + \omega_2\, y_2^2 + \omega_3\, y_3^2 = \frac{1}{12} \quad \left(= \iint_T y^2\, dxdy\right), \\
\omega_1\, x_1 y_1 + \omega_2\, x_2 y_2 + \omega_3\, x_3 y_3 = \frac{1}{24} \quad \left(= \iint_T xy\, dxdy\right), \\
x_2 = y_3, \\
x_3 = y_2, \\
\omega_2 = \omega_3.
\end{array}$$

On choisit de prendre $\omega_1 = \omega_2 = \omega_3 = \frac{1}{6}$. Vérifiez dans ce cas que le problème peut se réduire à un système $F(X) = 0$ de 3 équations algébriques à 3 inconnues $X = (x_1, x_2, y_2)$:

$$\begin{array}{l}
x_1 + x_2 + y_2 - 1 = 0, \\
x_1^2 + x_2^2 + y_2^2 - \frac{1}{2} = 0, \\
x_1^2 + 2x_2 y_2 - \frac{1}{4} = 0.
\end{array}$$

Résoudre d'abord ce système "à la main" (si c'est possible !). On va ensuite vérifier que `fsolve()` retourne la même solution.

Ecrire la fonction F, calculer sa matrice jacobienne J et programmer les fonctions Scilab correspondantes. Puis appliquer `fsolve()` avec les initialisations suivantes :

$$P_1^0 = (0, 0),\ P_2^0 = (0, 1),\ P_3^0 = (1, 0).$$

La formule d'intégration ainsi trouvée s'appelle la formule de Hammer-Stroud d'ordre 2. Tracer le triangle unité et afficher les points de Hammer ainsi trouvés. Il existe bien sûr des formules d'ordre plus élevé faisant intervenir plus de points et de poids d'intégration.