FSAB1104 : solution de l'examen de janvier 2010

1 Approximation de Fourier

Nous disposons d'un ensemble de huit données d'une fonction inconnue $u(x)$ dont nous souhaitons calculer une approximation au sens des moindres carrés $u^h(x)$ à partir de trois fonctions cosinus

$$u^h(x) = \sum_{i=1}^{3} a_i \cos(ix)$$

	X_i	U_i
0	0	2
1	$\pi/4$	1
2	$\pi/2$	0
3	$3\pi/4$	2
4	π	1
5	$5\pi/4$	1
6	$3\pi/2$	0
7	$7\pi/4$	0

1. Ecrire la fonction $J(a_1, a_2, a_3)$ qu'il faut minimiser pour résoudre un tel problème.

$$J(a_1, a_2, a_3) = \sum_{i=0}^{7} \left(U_i - \sum_{j=1}^{3} \cos(jX_i)\ a_j \right)^2$$

Cette question est vraiment très simple. Il faut les deux sommes écrites de manière adéquate. Ne considérer que les 3 premiers points en effectuant donc une simple interpolation est impardonnable (et n'a donc pas été admis !)

2. En déduire le système à résoudre pour obtenir les trois coefficients a_i.

Il suffit d'annuler les dérivées partielles de J par rapport à a_1, a_2 et a_3 afin d'obtenir :

$$\begin{bmatrix} \sum_{i=0}^{7} \cos(X_i)\cos(X_i) & \sum_{i=0}^{7} \cos(X_i)\cos(2X_i) & \sum_{i=0}^{7} \cos(X_i)\cos(3X_i) \\ \sum_{i=0}^{7} \cos(2X_i)\cos(X_i) & \sum_{i=0}^{7} \cos(2X_i)\cos(2X_i) & \sum_{i=0}^{7} \cos(2X_i)\cos(3X_i) \\ \sum_{i=0}^{7} \cos(3X_i)\cos(X_i) & \sum_{i=0}^{7} \cos(3X_i)\cos(2X_i) & \sum_{i=0}^{7} \cos(3X_i)\cos(3X_i) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{7} \cos(X_i)U_i \\ \sum_{i=0}^{7} \cos(2X_i)U_i \\ \sum_{i=0}^{7} \cos(3X_i)U_i \end{bmatrix}.$$

3. Calculer les valeurs des coefficients a_i.

Le système linéaire devient simplement

$$\begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1-\sqrt{2} \\ 3 \\ 1+\sqrt{2} \end{bmatrix}$$

On obtient immédiatement $a_1 = (1-\sqrt{2})/4$, $a_2 = 3/4$ et $a_3 = (1+\sqrt{2})/4$

4. Compléter le programme MATLAB ci-dessous afin d'obtenir la courbe de l'approximation.

Voici un exemple de programme utilisant les données. (Certains petits taquins codent directement les valeurs analytiques : c'est gentiment se foutre de la gueule du correcteur qui n'en a pas tenu rigueur, toutefois)

```
X = [0 1 2 3 4 5 6 7] * pi / 4;
U = [2 1 0 2 1 1 0 0];
x = linspace(0,2*pi,100)
a = [cos(X) * U'  cos(2*X) * U'  cos(3*X) * U'] /4;
uh = a * cos([1 2 3]' * x);
plot(x,uh);
```

2 Méthode de Steffensen

Afin de trouver la solution d'un problème non-linéaire, nous allons comparer les performances des méthodes de Newton-Raphson, de la sécante et de Steffensen.

		Taux de convergence
méthode de Newton-Raphson	$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$	2
méthode de la sécante	$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$	1.618
méthode de Steffensen	$x_{i+1} = x_i - \frac{f(x_i)f(x_i)}{f(x_i + f(x_i)) - f(x_i)}$	r

1. Définir le taux de convergence d'une méthode itérative.

   Si les itérations convergent et s'il existe deux constantes strictement positives C et r telles que

   $$\lim_{i \to \infty} \frac{|e_i|}{|e_{i-1}|^r} = C$$

   on dit que la séquence converge avec un taux de convergence r.

2. Calculer le taux de convergence de la méthode de Steffensen, en justifiant rigoureusement votre réponse avec une petite démonstration[1].

   Si la méthode converge, la fonction f tend progressivement vers zéro, on peut donc écrire que :

   $$x_{i+1} = x_i - \frac{f(x_i)f(x_i)}{f(x_i + f(x_i)) - f(x_i)}$$

   $\downarrow$ *En effectuant en développement en série de Taylor :* $f(x + f(x)) = f(x) + f'(x)f(x) + f''(x)\frac{f^2(x)}{2} + \ldots$

   $$x_{i+1} = x_i - \frac{f(x_i)f(x_i)}{f'(x_i)f(x_i) + f''(x_i)\frac{f^2(x_i)}{2} + \ldots}$$

   $\downarrow$ *En observant que $f \gg f^2$ lorqu'on arrive à la convergence !*

   $$x_{i+1} = x_i - \frac{f(x_i)f(x_i)}{f'(x_i)f(x_i)} = x_i - \frac{f(x_i)}{f'(x_i)}$$

   Lorsqu'on converge, cette méthode tend vers celle de Newton-Raphson et a donc $r = 2$. □

Très très peu d'étudiants ont réussi à obtenir ce résultat.
Steffensen utilise simplement la définition usuelle de la dérivée avec comme incrément f.
Bravo aux quelques étudiants futés qui ont trouvé !

[1] Vous pouvez toutefois considérer comme résultat acquis les taux de convergence de la méthode de la sécante et de celle de Newton-Raphson fournis dans l'énoncé. Il ne faut donc PAS redémontrer que le taux de convergence de la méthode de Newton-Raphson est quadratique.

3. Comparer les *taux de convergence des trois méthodes par nombre d'évaluation* de f (ou f').
   Quelle est la méthode dont le taux de convergence par évaluation de fonction est le plus élevé ?

   Ici, il faut observer qu'une itération de Newton-Raphson nécessite le calcul de $f(x_i)$ et de $f'(x_i)$, tandis qu'une itération de la sécante ne nécessite que le calcul de $f(x_i)$ (puisque $f(x_{i-1})$ a été calculé à l'itération précédente). En d'autres mots, il faut comparer le gain d'une itération de Newton-Raphson, à celle des deux itérations de la sécante.

   Le coût d'une itération de Steffensen est identique à celui de Newton-Raphson, mais ne nécessite pas l'expression analytique de la dérivée. C'est donc une méthode plus intéressante que celle de Newton-Raphson dans le cas scalaire !

	Nombre d'évaluations	*Taux de convergence*
Newton-Raphson	2 : $f(x_i)$ *et* $f'(x_i)$	2
sécante	1 : $f(x_i)$ *(car* $f(x_{i-1})$ *a déjà été évalué !)*	$(1.618)^2 = 2.62$
Steffensen	2 : $f(x_i)$ *et* $f(x_i + f)$	2

   Honte aux multiples étudiants qui m'expliquent que les méthodes de la sécante et de Steffensen requièrent respectivement 3 et 5 évaluations de la fonction...

4. Ecrire le code MATLAB de `function x = steffensen(x,tol,nmax,f)` implémentant la méthode de Steffensen pour trouver une racine d'une fonction `f` avec une tolérance `tol` et un nombre maximal d'itérations `nmax`. Le candidat initial est `x`.

   Voici un exemple de programme. C'est vraiment élémentaire et pourtant...

```
function x= steffensen(x,tol,nmax,f);
n = 0; delta = tol + 1;
while abs(delta) >= tol && n <= nmax
   n = n + 1;
   fx = f(x);
   dfdx = (f(x+fx) - fx) / fx;
   delta = -fx/dfdx;
   x = x + delta;
end
if (n >= nmax)  error();
end
```

   Ecrire un programme qui effectue 5 évaluations de f en recopiant bêtement la formule est considéré comme un erreur peu pardonnable et fait perdre la majorité des points. A mon grand dam (et d'un nombre important d'étudiants), c'est malheureusement l'option de la plupart des étudiants. Et parfois, ces étudiants viennent de manière parfaitement correcte d'estimer le nombre requis d'évaluations de f pour la méthode de Steffensen dans la question qui précédait (si, si !).

   Il n'est pas interdit d'utiliser le même symbole pour la valeur d'entrée et la solution finale, malgré ce que pensait l'un ou l'autre d'entre vous.

3 Backward Differential Formulae : méthodes BDF

Pour intégrer une équation différentielle ordinaire

$$u'(x) = f(x, u(x)),$$

nous souhaitons utiliser des formules de différentiation rétrograde (notées BDF) définies par :

$$U_{i+1} = \sum_{j=0}^{p} a_j U_{i-j} + hb \underbrace{f(X_{i+1}, U_{i+1})}_{F_{i+1}}.$$

Ce sont des méthodes multi-pas implicites dont les coefficients a_i et b sont calculés en approchant directement la valeur de la dérivée première de u au noeud X_{i+1} par la dérivée première du polynôme interpolant u aux $p+2$ noeuds $X_{i+1}, X_i, \ldots, X_{i-p}$ avec $p \geq 0$. Le pas entre deux abscisses est noté h. Les méthodes BDF les plus simples ($p = 0$ et $p = 1$) sont définies par le tableau.

	a_0	a_1	a_2	b
$p=0$	1	0	0	1
$p=1$	$\frac{4}{3}$	$-\frac{1}{3}$	0	$\frac{2}{3}$

1. Donner l'expression du polynôme de Lagrange interpolant u aux 4 noeuds $X_{i+1}, X_i, X_{i-1}, X_{i-2}$
   La polynôme de Lagrange s'écrit simplement :

$$\begin{aligned} u_h(x) &= U_{i+1} \frac{(x-X_i)(x-X_{i-1})(x-X_{i-2})}{6h^3} + U_i \frac{(x-X_{i+1})(x-X_{i-1})(x-X_{i-2})}{-2h^3} \\ &\quad + U_{i-1} \frac{(x-X_{i+1})(x-X_i)(x-X_{i-2})}{2h^3} + U_{i-2} \frac{(x-X_{i+1})(x-X_i)(x-X_{i-1})}{-6h^3} \end{aligned}$$

2. Calculer la dérivée de ce polynôme en $x = X_{i+1}$ en termes de $U_{i+1}, U_i, U_{i-1}, U_{i-2}$ et h.
   En déduire les coefficients a_0, a_1, a_2 et b pour la méthode BDF2 ($p = 2$).
   En évitant de développer bêtement et mécaniquement, la dérivée s'écrit simplement :

$$\begin{aligned} u'_h(x) &= U_{i+1} \frac{(x-X_{i-1})(x-X_{i-2}) + (x-X_i)(x-X_{i-2}) + (x-X_i)(x-X_{i-1})}{6h^3} \\ &\quad + U_i \frac{(x-X_{i-1})(x-X_{i-2}) + \ldots}{-2h^3} \\ &\quad + U_{i-1} \frac{(x-X_i)(x-X_{i-2}) + \ldots}{2h^3} \\ &\quad + U_{i-2} \frac{(x-X_i)(x-X_{i-1}) + \ldots}{-6h^3} \end{aligned}$$

   En évaluant en $x = X_{i+1}$

$$u'_h(X_{i+1}) = U_{i+1} \frac{6h^2+3h^2+2h2}{6h^3} + U_i \frac{6h^2+0+0}{-2h^3} + U_{i-1} \frac{3h^2+0+0}{2h^3} + U_{i-2} \frac{2h^2+0+0}{-6h^3}$$

$$u'_h(X_{i+1}) = \frac{11U_{i+1} - 18U_i + 9U_{i-1} - 2U_{i-2}}{6h}$$

On obtient donc :
$$\boxed{U_{i+1} = \frac{18}{11}U_i - \frac{9}{11}U_{i-1} + \frac{2}{11}U_{i-2} + \frac{6h}{11}F_{i+1}}$$

3. Quel est l'ordre de précision de la méthode BDF2 ?

   En observant que la méthode BDF0 est la méthode d'Euler implicite d'ordre un, on pouvait immédiatement déduire que l'ordre de cette formule est 3. *Si nécessaire un petit développement de Taylor (non demandé !) pouvait vous rassurer complètement !*

4. Compléter le programme MATLAB ci-dessous afin d'obtenir la zone de stabilité[2] du plan complexe de $h\lambda$ de la méthode BDF2.

   Le programme s'écrit simplement en recherchant les valeurs possibles du facteur d'amplification α lorsqu'on pose $U_i = \alpha^i U_0$ et qu'on développe :

$$\begin{aligned} U_{i+1} &= \frac{18}{11}U_i - \frac{9}{11}U_{i-1} + \frac{2}{11}U_{i-2} + \frac{6h}{11}F_{i+1} \\ &\downarrow \\ \alpha^3 U_0 &= \frac{18}{11}\alpha^2 U_0 - \frac{9}{11}\alpha U_0 + \frac{2}{11}U_0 + \frac{6h}{11}\lambda\alpha^3 U_0 \\ 0 &= (6h\lambda - 11)\alpha^3 + 18\alpha^2 - 9\alpha + 2 \end{aligned}$$

   La méthode sera stable si le module de la plus grande racine est inférieur à un. La méthode BDF2 (ou méthode de Gear d'ordre 3) est stable pour l'entièreté du domaine réel.

```
[x,y]=meshgrid([-8:0.5:8],[-8:0.5:8]);
z = x+i*y;
alpha = z;   % pre-allocation de alpha :-)
for k=1:size(z,1)
    for l =1:size(z,2)
       c = [(z(k,l)*6 -11)  18 -9  2];
       r = roots(c);
       alpha(k,l) = max(abs(r));
    end
end
figure; contourf(x,y,-alpha,[-1:0.1:0]); grid;
```

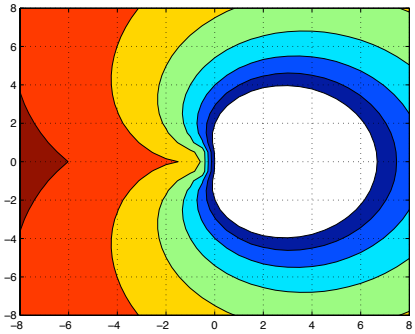

Figure 1: Zone de stabilité de la méthode de Gear d'ordre 3.

[2] En effectuant l'analyse du problème modèle $u' = \lambda u$.