INSA de Rennes
3 GM

Année universitaire 2018-2019
TP-Examen Méth. Num. Non Lin.
Mardi 2 avril 2019 — durée : 2h

Première partie : Fonctions équioscillantes

Soit f une fonction continue sur $[0,1]$ et une suite finie à $n+1$ éléments : $0 \leq x_1 < x_2 < \ldots < x_n < x_{n+1} \leq 1$. Il s'agit de déterminer le polynôme de degré $n-1$, $p(t) = \sum_{k=1}^{n} \alpha_k t^{k-1}$ tel que :

$$f(x_i) - p(x_i) = (-1)^{i+1}(f(x_1) - p(x_1)),\ i = 2, \ldots, n+1 \tag{1}$$

Les coefficients $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^T$ de p dans la base canonique vérifient alors le système $\boldsymbol{A} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \boldsymbol{b}$ où $\boldsymbol{A} = \boldsymbol{A}_1 - \boldsymbol{A}_2 \in \mathbb{R}^{n\times n}$ et $\boldsymbol{b} = \boldsymbol{b}_1 - \boldsymbol{b}_2 \in \mathbb{R}^n$ définis par

$$\boldsymbol{A}_1 = \begin{bmatrix} 1 & x_2 & x_2^2 & \ldots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \ldots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n+1} & x_{n+1}^2 & \ldots & x_{n+1}^{n-1} \end{bmatrix},\ \boldsymbol{A}_2 = \begin{bmatrix} -1 & -x_1 & -x_1^2 & \ldots & -x_1^{n-1} \\ 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ -1 & -x_1 & -x_1^2 & \ldots & -x_1^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ (-1)^n & (-1)^n x_1 & (-1)^n x_1^2 & \ldots & (-1)^n x_1^{n-1} \end{bmatrix}$$

et

$$\boldsymbol{b}_1 = \begin{bmatrix} f(x_2) \\ f(x_3) \\ \vdots \\ \vdots \\ f(x_{n+1}) \end{bmatrix},\ \boldsymbol{b}_2 = f(x_1) \begin{bmatrix} -1 \\ 1 \\ -1 \\ \vdots \\ (-1)^n \end{bmatrix}$$

1. Écrire un fichier de type fonction qui étant donné un vecteur $\boldsymbol{x}$ construit le vecteur $f(\boldsymbol{x})$. Sauvegarde sous `f1.m`. Test avec $f1(x) = e^x$ et $\boldsymbol{x} = 0:3$.

```
>> f1(0:3)'
ans =
    1.0000
    2.7183
    7.3891
   20.0855
```

2. Écrire un fichier function `vdm.m` qui étant donné un vecteur $\boldsymbol{u}$ (en colonne) construit la matrice de Vandermond $\boldsymbol{M} = \begin{bmatrix} 1 & u_1 & u_1^2 & \ldots & u_1^{n-1} \\ 1 & u_2 & u_2^2 & \ldots & u_2^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & u_n & u_n^2 & \ldots & u_n^{n-1} \end{bmatrix}$. Test avec $\boldsymbol{u} = (1:3)'$.

```
>> u=(1:3)'; M=vdm(u)
M =
     1     1     1
     1     2     4
     1     3     9
```

3. Écrire un fichier de type fonction qui étant donné le vecteur $\boldsymbol{x}$ (en colonne) construit les matrices $\boldsymbol{A}_1$ et $\boldsymbol{A}_2$. Sauvegarde sous `eqosVI1.m` (*Dans la suite* `VI` *signifie vos initiales...*). Test avec $\boldsymbol{x} = (2:5)'$. Attention il faudra déterminer n où $\dim(x) = n + 1$.

```
>> [A1,A2]=eqosVI1((2:5)')
A1 =
     1     3     9
     1     4    16
     1     5    25
A2 =
    -1    -2    -4
     1     2     4
    -1    -2    -4
```

4. Modifier le fichier de type fonction précédent en ajoutant en variable d'entrée la fonction f et construire $\boldsymbol{b}_1$ et $\boldsymbol{b}_2$ (on garde les constructions des matrices). Sauvegarde sous `eqosVI2.m`. Test avec $\boldsymbol{x} = (2:5)'$ et $fichier =' f1'$.

```
>> [b1,b2]=eqosVI2('f1',(2:5)')
b1 =
   20.0855
   54.5982
  148.4132
b2 =
   -7.3891
    7.3891
   -7.3891
```

5. Modifier la fonction précédente (paramètres de sortie) en résolvant le système linéaire pour calculer les coefficients de p. Sauvegarde sous `eqosVI3.m`. Même test.

```
>> coeff=eqosVI3('f1',(2:5)')
coeff =
  127.1028
  -98.0732
   20.2796
```

6. Ecrire un programme qui étant donné n et une fonction f définie sur $[0,1]$, calcule une sudivision régulière $\boldsymbol{x} = [0, \ldots, x_i = (i-1)/n, \ldots, 1]^T \in \mathbb{R}^{n+1}$ de $[0,1]$, détermine les coefficients du polynôme tel que $f - p$ équioscille, puis dessine les

graphes du polynôme p[1] et de la fonction f sur $[0,1]$. Test avec $n=5$, $f1(x)=e^x$. Sauvegarde sous `VI4.m`. Autre test avec $n=5$, $f2(x)=|x-1/3|$. Sauvegarde sous `VI5.m`

Seconde partie : Algorithme de Remez

Soit f une fonction continue sur $[0,1]$. Il s'agit de déterminer un polynôme de degré $n-1$, $p(t)=\sum_{k=1}^{n}\alpha_k t^{k-1}$ qui minimise $\|f-q\|_\infty=\max_{t\in[0,1]}(|f(t)-q(t)|)$ parmi les polynômes $q\in\mathbb{P}_{n-1}$. On montre que le polynôme p, solution du problème est tel que $f-p$ équioscille en au moins $n+1$ points distincts avec, en plus,
$\|f-p\|_\infty=\max_{0\le x_1<\ldots<x_{n+1}\le 1}(|f(x_i)-p(x_i)|)$.

Par rapport à la partie précédente, il s'agit donc de construire aussi les points x_i.

L'algorithme :

Initialisation : on choisit $n+1$ points $0\le x_1^0<x_2^0<\ldots<x_n^0<x_{n+1}^0\le 1$.
Etape k : on construit le polynôme p de degré $n-1$ tel que
$f(x_i^k)-p(x_i^k)=(-1)^{i+1}(f(x_1^k)-p(x_1^k))$, $i=2,\ldots,n+1$.

1. *Si $\|f-p\|_\infty=\max_{i=1,\ldots,n+1}(|f(x_i^k)-p(x_i^k)|)$, c'est fini,*

2. *Sinon, il existe $y\in[0,1]$ tel que*
   $\|f-p\|_\infty=|f(y)-p(y)|>\max_{i=1,\ldots,n+1}(|f(x_i^k)-p(x_i^k)|)$.
   On construit une nouvelle suite $0\le x_1^{k+1}\ldots<x_{n+1}^{k+1}\le 1$ en modifiant un des points x_i^k sans changer les autres. Il y a 3 possibilités à 2 choix:

   (a) *Si $y\in[0,x_1^k[$,*
   - *si $(f(x_1^k)-p(x_1^k))(f(y)-p(y))\ge 0$ alors $x_1^{k+1}=y$ et $x_i^{k+1}=x_i^k$ pour $i=2,\ldots,n+1$,*
   - *sinon i.e. $(f(x_1^k)-p(x_1^k))(f(y)-p(y))<0$ alors $x_1^{k+1}=y$ et $x_i^{k+1}=x_{i-1}^k$ pour $i=2,\ldots,n+1$,*

   (b) *Si $y\in]x_{n+1}^k,1]$,*
   - *si $(f(x_{n+1}^k)-p(x_{n+1}^k))(f(y)-p(y))\ge 0$ alors $x_{n+1}^{k+1}=y$ et $x_i^{k+1}=x_i^k$ pour $i=1,\ldots,n$,*
   - *sinon i.e. $(f(x_{x_{n+1}}^k)-p(x_{n+1}^k))(f(y)-p(y))<0$ alors $x_{n+1}^{k+1}=y$ et $x_i^{k+1}=x_{i+1}^k$ pour $i=1,\ldots,n$,*

   (c) *Si $y\in]x_{i_0}^k,x_{i_0+1}^k[$,*
   - *si $(f(x_{i_0}^k)-p(x_{i_0}^k))(f(y)-p(y))\ge 0$ alors $x_{i_0}^{k+1}=y$ et $x_i^{k+1}=x_i^k$ pour $i=1,\ldots,n+1$, $i\ne i_0$,*
   - *sinon i.e. $(f(x_{i_0}^k)-p(x_{i_0}^k))(f(y)-p(y))<0$ alors $x_{i_0+1}^{k+1}=y$ et $x_i^{k+1}=x_i^k$ pour $k=1,\ldots,n+1$, $i\ne i_0+1$.*

[1] Attention `polyval.m` calcule le polynôme quand les coefficients sont donnés de celui de plus haut degré au coefficient constant

Programmation :

De manière pratique $\|f-p\|_\infty$ est estimé par $erreurt = \max_{j=1,\ldots 1001} |f(t_j)-p(t_j)|$ où $t_1,\ldots,t_{1001}$ est une subdivision uniforme de l'intervalle.

Si $erreurx = \max_{i=1,\ldots,n+1}(|f(x_i^k)-p(x_i^k)|)$, la condition $\|f-p\|_\infty = \max_{i=1,\ldots,n+1}(|f(x_i^k)-p(x_i^k)|)$ est remplacée par $diff < precis$ où $diff = erreurt - erreurx$ et $precis$ est fixée a priori. Ainsi le y cherché est l'un des t_j. Par ailleurs, le nombre $iter$ d'itérations doit être limité par une constante $maxiter$.

1. Ecrire un programme `VI6.m` qui étant donné un vecteur ordonné de réels de $[0,1]$, $\boldsymbol{x} = [x_1,\ldots,x_{n+1}]^T$ calcule les coefficients du polynôme p par `eqosVI3.m`, puis calcule $erreurx$, construit la subdivision $t_j = (j-1)/1000$ pour $j = 1,\ldots,1001$ et calcule $erreurt = \max_{j=1,\ldots 1001} |f(t_j)-p(t_j)|$ ainsi que j_0 où le maximum est obtenu [2]. Sauvegarde dans `errVI6`. Test avec $n = 10$, $x = 0 : 1/n : 1$, $f = f2$. Afficher $erreurx$, $erreurt$, $diff$ et j_0. On désignera par $\boldsymbol{fx}$ le vecteur $f(x)$, $\boldsymbol{px}$ pour $p(x)$, $\boldsymbol{ft}$, $\boldsymbol{pt}$ de même.

   Note : Pour une suite ordonnée $0 \le x_1 < x_2 < \ldots < x_{n+1} \le 1$ *et* $y \in [0,1]$, *les instructions (*`>>i=find(x>y,1)-1;if x(n+1)<y i=n+1; end`*) donne un indice* i *tel que*

$i = 0$ *si* $y < x_1$	$i = i_0$ *si* $x_{i_0} < y \le x_{i_0+1}$	$i = n+1$ *si* $y > x_{n+1}$

2. Programmer l'algorithme de Remez, (test avec $n = 10$, $x = 0 : 1/n : 1$, $f = f2$, $maxiter = 20$, $precis = 1e-4$) en utilisant la structure de programme de la page suivante :

[2] utiliser `[erreurt,j0]=max..`

```
clear
n=10;
maxiter=20; iter=1;
precis=1e-4;

nomf='f2';
%initialisations: x,coeff, px,fx, ft, pt, erreurx,diff...
...
[erreurt,j0]=max(abs(pt-ft));
diff=

% iterations
while (iter<=maxiter)&(diff>precis)
    y=t(j0);
    i=find(x>y,1)-1;if x(n+1)<y   i=n+1; end
    switch i
        case 0 % cas (a)
            ...
        case n+1 % cas (b)
            ...
        otherwise % cas (c)
            ...
    end
  % mise a jour coeff, px,fx, ft, pt, erreurx,diff...
                ...
    iter=iter+1;
end
iter
erreurt
plot(x,fx,'bx',t,ft,'b',x,px,'rx',t,pt,'r')
%graphiques
```

Dessiner les graphes de f et p, afficher *erreurt*. Sauvegarde sous `RemezVI7.m`.