Méthodes numériques
Année : 2018-2019
Formation : L3 Mathématiques

TP noté

Ce sujet est composé de quatre parties indépendantes. Pour chaque partie il faudra créer deux fichiers dans lesquels vous enregistrerez votre code : un fichier en .sci qui contiendra toutes les fonctions de la partie et un fichier en .sce qui contiendra les autres commandes.

Des points supplémentaires pourront être donnés pour la présentation du code (indentation, qualité des commentaires). Tout code contenant une erreur devra être commenté. Des points pourront être enlevés si des erreurs (code d'erreur dans la console Scilab) surviennent lors de l'exécution de vos fichiers.

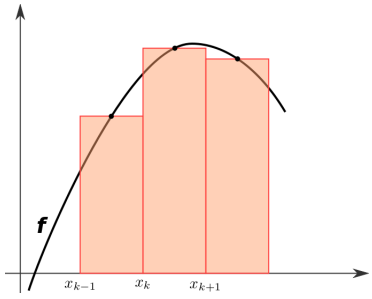
A la fin du TP vous devez envoyer vos fichiers par mail à l'adresse suivante : florian.le-manach@math.u-bordeaux.fr. N'oubliez pas d'indiquer votre nom et prénom (et celui de votre binôme si vous faites le TP à deux) dans le mail.

1 Intégration numérique ($\sim$ 5 points)

Soit f une fonction continue sur $[a,b]$. On souhaite calculer numériquement la valeur de $\int_a^b f(t)dt$. Pour cela on subdivise l'intervalle $[a,b]$ par une suite de points équidistants $(x_k)_{1\leq k\leq n}$ vérifiant

$$a = x_1 < x_2 < \cdots < x_n = b. \tag{1}$$

On approche alors l'intégrale $\int_a^b f(t)dt$ par la somme des aires des rectangles définis comme ci-dessous :



1. Définissez une fonction function s = calculint(f,a,b,n) qui prend en paramètre une fonction f, deux réels a et b, un entier n et qui renvoie l'approximation de $\int_a^b f(t)dt$.

2. Définissez la fonction $t \mapsto \sin(t)e^{\cos(t)}$ puis dessinez là sur l'intervalle $[0,3\pi]$. A l'aide de la fonction calculint, calculez une valeur approchée de $\int_0^{3\pi} \sin(t)e^{\cos(t)}dt$ puis comparez-là avec la valeur exacte.
Indication : dans la définition de la fonction, il est recommandé d'utiliser . pour la multiplication.*

3. Tracez sur un graphe l'erreur entre la valeur approchée de $\int_0^{3\pi} \sin(t)e^{\cos(t)}dt$ et la valeur exacte, en fonction de n pour n variant entre 6 et 50.

4. Définissez une fonction function L = primitive(f,a,T) qui prend en paramètre une fonction f, un réel a, une liste de réels T et qui renvoie la liste des valeurs, sur T, que prend la primitive de f s'annulant en a. Pour cela on utilisera la fonction calculint et on prendra $n = 100$. Affichez sur un graphe la primitive qui s'annule en 1 de la fonction $t \mapsto \frac{1}{t}$ sur $[1,5]$.

2 Polynômes de Laguerre (∼ 6 points)

La suite $(L_n)_{n\geq 0}$ des polynômes de Laguerre est définie par

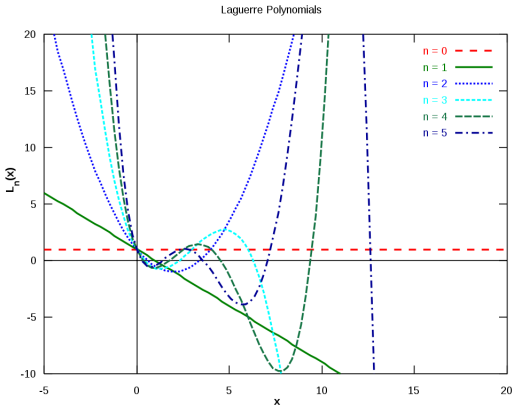
$$L_0(X) = 1, \quad L_1(X) = -X + 1, \quad (n+1)L_{n+1}(X) + (X - 2n - 1)L_n(X) + nL_{n-1}(X) = 0.$$

Ces polynômes forment une suite de polynômes orthogonaux pour le produit scalaire

$$\langle f, g\rangle = \int_0^\infty f(t)g(t)e^{-t}dt.$$

1. Définissez une fonction `function R = laguerre(n,T)` qui prend en paramètre un entier n, une liste T et qui renvoie la valeur du n-ième polynôme de Laguerre L_n sur les éléments de la liste T.

2. Tracez sur un même graphe les six premiers polynômes de Laguerre $L_0, \ldots, L_5$ (avec des couleurs différentes) dans la fenêtre $[-5, 20] \times [-10, 20]$. On pourra vérifier le résultat avec le graphe :



3. Soit f une fonction dérivable sur $[a, b]$ On considère, comme dans l'exercice 1 (voir (1)), une subdivision de l'intervalle $[a, b]$. Pour $k \notin \{1, 2, n-1, n\}$, on approche la dérivée de f en x_k par

$$f'(x_k) = \frac{f(x_{k-2}) - 8f(x_{k-1}) + 8f(x_{k+1}) - f(x_{k+2})}{12h}, \tag{2}$$

avec h la distance entre deux éléments consécutifs de la suite (x_k).
Définissez une fonction `function L = derive(F,a,h)` qui prend en paramètre la liste F des valeurs que prend la fonction f sur les x_k, $a = x_1$ et le pas h. Cette fonction doit calculer la liste des approximations de la dérivée de f en x_k à l'aide de (2). Pour l'évaluation de la dérivée de f en x_1, x_2, x_{n-1} et x_n vous pouvez utiliser une approximation d'ordre 1 ou 2 de votre choix.

4. A l'aide des questions précédentes, justifiez (numériquement) jusqu'à $n = 5$ que la suite L_n vérifie

$$xL_n'(x) - nL_n(x) + nL_{n-1}(x) = 0, \quad \forall x \in [-1, 1].$$

3 Interpolation (∼ 9 points)

Cet exercice est consacré à l'étude de deux méthodes d'interpolation : l'interpolation d'Hermite qui généralise l'interpolation lagrangienne (question 1, 2, 3 et 4) et les B-splines qui généralisent les courbes de Bézier (questions 5 et 6). Ces deux méthodes peuvent être traitées indépendamment.

L'interpolation d'Hermite permet d'interpoler sur $(x_i)_{1\leq i\leq n}$ une fonction f dérivable par un polynôme P vérifiant

$$P(x_i) = f(x_i) \quad \text{et} \quad P'(x_i) = f'(x_i), \quad 1 \leq i \leq n.$$

Le polynôme P est donné par

$$P(x) = \sum_{i=1}^{n} h_i(x) f(x_i) + \sum_{i=1}^{n} k_i(x) f'(x_i), \quad x \in \mathbb{R},$$

avec

$$h_i(x) = (1 - 2(x - x_i) l_i'(x_i)) l_i(x)^2,$$
$$k_i(x) = (x - x_i) l_i(x)^2$$

et où

$$l_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

1. Définissez une fonction `function y = li(LX,i,x)` qui prend en paramètre une liste LX représentant les réels $(x_i)_{1 \leq i \leq n}$, un entier $1 \leq i \leq n$, un réel x et qui renvoie la valeur $l_i(x)$ définie ci-dessus.

2. Dérivez formellement (à la main) la fonction l_i puis déterminez l'expression de $l_i'(x_i)$ en fonction de la suite $(x_j)_{1 \leq j \leq n}$. Définissez une fonction `function y = lider(LX,i)` qui prend en paramètre une liste LX représentant les réels $(x_i)_{1 \leq i \leq n}$, un entier $1 \leq i \leq n$ et qui renvoie la valeur $l_i'(x_i)$.
Indication : vous pouvez indiquer en commentaire l'expression de la dérivée formelle de l_i si vous n'êtes pas sûr de votre fonction lider. De plus on rappelle la formule de la dérivée d'un produit :

$$\left(\prod_{j=1}^{n} f_j \right)' = \sum_{i=1}^{n} f_i' \prod_{\substack{j=1 \\ j \neq i}}^{n} f_j.$$

3. Définissez une fonction `function S = hermite(LX,F,FP,T)` qui prend en paramètre une liste LX représentant les réels $(x_i)_{1 \leq i \leq n}$, une liste F représentant les valeurs $f(x_i)$, une liste FP représentant les valeurs $f'(x_i)$, une liste de réels T et qui renvoie la valeur du polynôme d'interpolation d'Hermite sur les éléments de la liste T.

4. Testez la fonction `hermite` pour $f = cos$, $LX = [-2, -1, 0, 1, 2]$ et $T = [-5 : 0.01 : 5]$. Tracez la courbe du polynôme d'interpolation d'Hermite.
Indication : Vous pouvez vérifier que la courbe ressemble à la courbe de la fonction cos.

Soit $(P_i)_{1 \leq i \leq m}$ des points de $\mathbb{R}^2$ et $(t_i)_{1 \leq i \leq m}$ une suite de réels vérifiant

$$0 \leq t_1 < t_2 < \cdots < t_m \leq 1.$$

On définit la courbe B-spline de degré n associée à (P_i) et (t_i) comme la courbe $S : [0, 1] \to \mathbb{R}^2$ définie par

$$S(t) = \sum_{i=1}^{m-n-1} b_{i,n}(t) P_i, \quad t \in [0, 1]$$

avec $b_{i,n}$ définie (par récurrence sur n) par

$$b_{i,0}(t) = \begin{cases} 1 & \text{si } t_i \leq t < t_{i+1} \\ 0 & \text{sinon} \end{cases}.$$

et

$$b_{i,n}(t) = \frac{t - t_i}{t_{i+n} - t_i} b_{i,n-1}(t) + \frac{t_{i+n+1} - t}{t_{i+n+1} - t_{i+1}} b_{i+1,n-1}(t).$$

5. Définissez une fonction `function S = bspline(T,P,m,n,L)` qui prend en paramètre la liste $T = (t_i)_{0 \leq i \leq m}$, la liste de points $P = (P_i)_{0 \leq i \leq m}$ sous forme d'une matrice $2 \times (m + 1)$, un entier n et une liste de réels L. La fonction renvoie la valeur de la courbe B-spline de degré n associée à (P_i) et (t_i) sur tous les éléments de L. *Indication : Vous pouvez faire une fonction récursive permettant de calculer les $b_{i,n}(t)$.*

6. Dessinez la courbe B-spline de degré 2 associée à $m = 10$, $P = \begin{pmatrix} 1,6 & 9 & 8 & 1 & 1 & 6 & 9 \\ 9 & 9 & 4 & 7 & 3 & 1 & 3 \end{pmatrix}$ et $T = [0 : 0.14 : 1.35]$. *Indication : Vous devez obtenir une courbe en forme de "e" minuscule.*

4 Calcul de puissance (exercice bonus : $\sim$ 5 points)

L'objectif de cette partie est de construire une fonction puissance qui permet, pour $x \in \mathbb{R}$ et $n \in \mathbb{N}$, de calculer x^n. Évidemment dans toute cette partie il sera interdit d'utiliser la commande Scilab `x^n` pour calculer x^n.

Les trois première questions peuvent être traitées de manière indépendante.

Dans un premier temps on va construire une fonction puissance de manière naïve qui effectuera n (ou $n-1$) multiplications pour calculer x^n.

1. Définissez la fonction `function r=puiss1(x,n)` qui prend en paramètre un réel x et un entier naturel n et qui renvoie x^n. On rappelle qu'il est interdit d'utiliser le symbole ^ dans cette fonction.

On cherche maintenant a améliorer le calcul de x^n en effectuant le moins d'opération possible. Pour cela on s'intéresse à la décomposition en base 2 de n.

On rapelle que tout nombre entier n se décompose de manière unique en

$$n = \sum_{i=0}^{k} c_i 2^i = c_0 + c_1 \times 2 + c_2 \times 2^2 + \cdots + c_k 2^k$$

avec $c_i = 0$ ou $c_i = 1$.

2. Définissez la fonction `function T=decompo2(n)` qui prend en paramètre un entier naturel n et qui renvoie la liste `T=`$[c_0, c_1, \ldots, c_k]$ de sa décomposition en base 2.
Indication : On commence par déterminer c_0 en regardant la parité de n : si n est pair alors $c_0 = 0$ et si n est impair alors $c_0 = 1$. Ensuite pour déterminer c_1 on recommence cette opération avec $n/2$ si n est pair et $(n-1)/2$ si n est impair. On itère ce procéder jusqu'à obtenir $n = 0$.

3. Définissez la fonction `function L=puissListe2(x,k)` qui prend en paramètre une réel x et un entier naturel k et qui renvoie `L=`$[x, x^2, x^{2^2}, \ldots, x^{2^k}]$. Cette fonction ne doit pas faire plus de k multiplications. On rappelle qu'il est interdit d'utiliser le symbole ^ dans cette fonction.
Indication : $x^{2^{i+1}} = x^{2^i} \times x^{2^i}$.

On revient maintenant au problème initial, à savoir calculer x^n avec le moins d'opération possible. Si la décomposition de n en base 2 est

$$n = \sum_{i=0}^{k} c_i 2^i$$

alors on observe que

$$x^n = x^{c_0} \times (x^2)^{c_1} \times (x^{2^2})^{c_2} \times \cdots \times (x^{2^k})^{c_k}.$$

4. A l'aide de l'observation ci-dessus et à l'aide des fonctions définies dans les questions 2 et 3, définissez la fonction `function r=puiss2(x,n)` qui prend en paramètre une réel x et un entier naturel n et qui renvoie `r=`x^n.

5. Combien faut-il de multiplications pour calculer x^{12} à l'aide de la seconde méthode ?

6. Comparez, avec la commande `timer()`, le temps d'exécution des commandes `puiss1(2,10000000)` et `puiss2(2,10000000)`. Quelle fonction est la plus rapide ?

7. Écrire une fonction `function r=puiss3(x,n)` basée sur la fonction `puiss2` mais cette fois sans utiliser les fonctions des question 2 et 3 et sans utiliser de listes. Le but de cette question est de réduire l'utilisation de la mémoire.