TP Noté

Remarque. Documents donnés en cours et programmes Scilab effectués en TP autorisés

1. Interpolation

La méthode des différences divisées de Newton n'a pas que des avantages. Pour calculer simultanément l'interpolation de plusieurs fonctions pour une même subdivision on peut préférer utiliser la méthode dite « formule barycentrique ». Le but est d'implémenter cette méthode et éventuellement de la comparer.

Soient $n \in \mathbf{N}$ et $n+1$ points du plan $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, d'abscisses x_i deux-à-deux distinctes. On rappelle qu'en introduisant les polynômes

$$L_i(x) = \prod_{\substack{0 \le j \le n \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$$

on montre qu'il existe un et un seul polynôme de degré (au plus) n passant par ces $n+1$ points du plan. On l'appelle le **polynôme d'interpolation de Lagrange** de ces points. Si f est une fonction définie sur un intervalle contenant les points $x_0, \ldots, x_n$ on appelle polynôme d'interpolation de Lagrange de f, l'unique polynôme de degré au plus n passant par les points $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$.

L'expression de p est tout simplement

$$p(x) = \sum_{i=0}^{n} y_i L_i(x).$$

Cette formule n'est pas très pratique ($n+1$ termes à calculer chacun étant le produit de n termes pour chaque évaluation en x), mais il est facile de la transformer. En posant

$$L(x) = \prod_{0 \le i \le n} (x - x_i),$$

$$w_i = \prod_{\substack{0 \le j \le n \\ j \neq i}} \frac{1}{x_i - x_j} \quad \text{pour tout } i \in \{0, \ldots, n\}$$

et en utilisant la relation

$$L_i(x) = L(x) \frac{w_i}{x - x_i},$$

on peut déjà réduire la complexité :

$$p(x) = \sum_{i=0}^{n} y_i L(x) \frac{w_i}{x - x_i} = L(x) \sum_{i=0}^{n} y_i \frac{w_i}{x - x_i}.$$

Une fois les w_i calculés l'évaluation nécessite de l'ordre de $O(n)$ opérations. On peut faire encore mieux en remarquant que

$$1 = \sum_{i=0}^{n} L_i(x) = L(x) \sum_{i=0}^{n} \frac{w_i}{x - x_i}.$$

On obtient alors la formule finale (ouf) appelée formule barycentrique, valable uniquement pour $x \notin \{x_0, \ldots, x_n\}$

$$p(x) = \frac{\displaystyle\sum_{i=0}^{n} \frac{w_i}{x - x_i} y_i}{\displaystyle\sum_{i=0}^{n} \frac{w_i}{x - x_i}}. \tag{1}$$

Pour $x \in \{x_0, \ldots, x_n\}$ nous n'avons pas besoin de formule : $p(x_i) = y_i$!

Les avantages de cette formule sont
— une vectorisation possible avec Scilab
— un calcul simultané de plusieurs interpolation en les mêmes points
— l'ajout d'un point pour l'interpolation possible, ce qui ne nécéssite pas de tout reprendre à zéro
— le calcul des w_i se fait en $O(n^2)$ opérations, une fois pour toute : on peut alors faire l'interpolation en $O(n)$ pour toute fonction (là où les différences divisées nécessitent de refaire un nouveau tableau pour une nouvelle fonction)

Les inconvénients de cette formule sont
— la formule (1) n'est valable que pour $x \neq x_i$. Ceci complique nettement la vectorisation Scilab (mais avec de l'aide on y a arrive). Il faut tester, gérer les divisions par zéro !
— l'optimisation du calcul des w_i nécessite de réfléchir (l'algorithme récursif ne sera pas demandé)

Gestion de `1/0`. Tester les commandes

```
1./0
0/0
ieee(2)
1./0
0/0
```

Dans la suite on aura au préalable imposé `ieee(2)`, ce qui évite une erreur lors d'une division par zéro (sic) !

Calcul des w_i. Écrire une routine Scilab `poids(x)` avec `x` un vecteur unidimensionel (notre subdivision) qui retourne le vecteur des w_i. On se limitera à deux boucles imbriquées et un test sur les indices

```
x en entrée
m=récupération taille de x (m=n+1, par rapport aux indices mathématiques)
w=vecteur de taille m rempli de 1
boucle i de 1 à m
boucle j de 1 à m
si j différent de i faire ce qu'il faut
fin boucle j
fin boucle i
```

La commande `poids(-2:.5:2)'` vous retournera

$$\begin{pmatrix} 0.0063492 & -0.0507937 & 0.1777778 & -0.3555556 & 0.4444444 \\ -0.3555556 & 0.1777778 & -0.0507937 & 0.0063492 & \end{pmatrix}$$

Calcul vectorisé des valeurs du polynômes d'interpolation. Pour plus de facilité écire la routine `bary(x,f,xx)` où `x` est le vecteur de la subdivision, `f` la fonction et `xx` les points en lesquels il faut évaluer p selon la formule (1). Il sera plus difficile (à une boucle et un test près) de donner en argument `y` les valeurs en `x`. La grande difficulté réside dans la gestion/détection des valeurs interdites de x dans la formule (1) et le remplacement de `Nan` par $f(x_i)$ avec le bon i. Mis à part cette détection on calcule dans une seule boucle le numérateur et le dénominateur de $p(x)$.

```
en entrée x,f,xpt : en sortie ypt
m=récupératoin taille de x
w=poids(x)
y=les valeurs de f en x
r=récupération taille de xpt
numera=zeros(r)
denomi=zeros(r)
probleme=zeros(r)  indice à problème
```

```
boucle i de 1 à m
xdiff= différence de xpt et x(i)
probleme(xdiff==0)=1  (détection des valeurs interdites)
probleme est rempli de zéro et là où il y a des 1 cela correspond
à un indice de xx valeur de la subdivision
numera = ce qu'il faut (vectorisé)
denomi = ce qu'il faut (vectorisé)
fin boucle i
ypt= division vectorisée de numera par denomi
jj=find(probleme==1) (récupération des indices à pb)
ypt=f(xpt(jj)) les valeurs à pb sont remplacées
```

Test 1. Prendre $f(x) = \cos(x)$, `deff=("y=f(x)","y=cos(x)")` sur l'intervalle $[-5,5]$ et faire des graphiques pour différentes subdivisions équidistantes de l'intervalle $[-5,5]$.

Comparer la stabilité numérique de cette méthode barycentrique par rapport aux différences divisées de Newton (reprendre son programme). Quelle méthode vous paraît préférable ?

Test 2. Visualiser l'effet de Runge pour $f(x) = 1/(1+x^2)$.

2. Intégration numérique : Weddle-Hardy

Formule élémentaire. La formule élementaire de Weddle-Hardy est la formule de Newton-Cotes fermée pour $n = 6$. Sur l'intervalle $[a,b]$ elle consiste à écrire

$$H(f,a,b) = \frac{b-a}{840} \sum_{i=0}^{6} A_i f(x_i)$$

avec $A_0 = A_6 = 41$, $A_1 = A_5 = 216$, $A_2 = A_4 = 27$, $A_3 = 272$ et $x_i = a + (b-a) * i/6$.

Écrire la routine `hardy(f,a,b)` qui renvoie l'approximation de $\int_a^b f(x)dx$ par la quantité $H(f,a,b)$. Vérifier expérimentalement à l'aide de `intg` implémentée dans Scilab que la formule est exacte sur les polynômes de degré inférieure à 7 (il suffit de tester les puissances de x). Qu'en est-il pour $f(x) = x^8$?

Formule composite. Rendre composite la formule élémentaire de Weddle-Hardy. On écrira une routine Scilab `chardy(f,a,b,n)` qui renvoie

$$\sum_{i=0}^{n-1} H(f, a_i, a_{i+1})$$

où $a_i = a + (b-a)i/n$.

Écrire une deuxième version `chardy1(f,a,b,n)` vectorisée et minimisant le nombre d'opérations effectuées. Il faudra écrire un peu de mathématiques.