

**M66, Modélisation et analyse numérique****TP1 : INTERPOLATION POLYNOMIALE**

Vous êtes invités à télécharger le fichier `tp1_fonction.sci` à partir de moodle, puis de le compléter au fur et à mesure avec les nouvelles procédures. Pour chaque exercices il va falloir créer un fichier, `tp1_exo1.sce`, `tp1_exo2.sce` et `tp1_exo3.sce`, comportant les lignes de code correspondant à la résolution de l'exercice et incluant au début le fichier `tp1_fonction.sci` et les autres initialisations habituelles (`clear,clc,...`).

En cas de blocage, commencez toujours par regarder l'aide ou sur Google !!

**Présentation et indications**

Soient  $n + 1$  points distincts  $(x_i)_{0 \leq i \leq n}$  de  $[a, b] \subset \mathbb{R}$ , et  $f$  une fonction réelle définie sur  $[a, b]$ . Nous cherchons à construire le polynôme d'interpolation de Lagrange  $p_n$  de degré au plus égal à  $n$  tel que  $p_n(x_i) = f(x_i)$  pour  $0 \leq i \leq n$ . On exprime ce polynôme dans sa base de Newton, sous la forme :

$$p_n(x) = f[x_0] + \sum_{i=1}^n f[x_0, \dots, x_i] \prod_{k=0}^{i-1} (x - x_k).$$

L'année dernière, en L2, vous avez déjà travaillé sur ce problème en Scilab. Pour ne pas devoir refaire le même travail, on vous fournit dans le fichier `tp1_fonction.sci` deux fonctions :

- a) **function** `[cfs]=DiffDiv(xdata,ydata)` qui à partir des points  $\{x_0, \dots, x_n\}$  contenus dans `xdata` et des valeurs de la fonctions  $\{f(x_0), \dots, f(x_n)\}$  contenus dans `ydata`, retourne les différences divisée `cfs` =  $\{f[x_0], \dots, f[x_0, \dots, x_n]\}$ .
- b) **function** `[y]=HornerNewton(cfs,xdata,t)` qui à partir des différences divisée `cfs` et les points `xdata`, retourne la valeur du polynôme d'interpolation évalué au point `t`.

## Exercices

### Exercice 1 (Vérification de l'interpolation)

- a) Commencer par créer le fichier `tp1_exo1.sce` qui sera initialisé par les commandes suivantes : `clc`, `clear`, `xdel(winsid())`, `exec('tp1_fonctions.sci')`. Commenter ces commandes.
- b) Créer dans ce même fichier la fonction polynomiale  $f_1(x) = x^2 + 2x$ , nommée `f1`. Puis vérifier qu'elle fonctionne.
- c) Dans le fichier `tp1_fonction.sci` créer la procédure

```
function PlotFunction(f,a,prec)
```

qui trace la fonction `f` sur l'intervalle `[-a,a]` avec un pas de précision `prec`. Tester cette procédure (dans `tp1_exo1.sce`) avec la fonction `f1`.

- d) Dans le fichier `tp1_fonction.sci` créer la procédure

```
function PlotInterpUnif(f,a,k,prec)
```

qui trace, avec un pas de précision `prec`, le polynôme d'interpolation uniforme<sup>1</sup> de degré au plus `k` de `f` sur l'intervalle `[-a,a]`.

- e) Sur la même figure tracer la fonctions  $f_1$  sur  $[-3,3]$  et ces polynômes d'interpolation uniforme  $p_k$  de degré majoré par  $k = 0, \dots, 10$ . Que constatez-vous ?
- f) Écrire un test qui imprime le degré du polynôme  $p_{10}$ . Pour déterminer le degré on peut se servir des fonctions `find`, `max`, et pour l'impression de `disp` et `sprintf`.  
Le résultat est-il cohérent avec la théorie ? Explication.
- g) En utilisant la fonction `norm` calculer et afficher les erreurs d'interpolation  $\|f_1 - p_2\|_\infty$  et  $\|f_1 - p_{10}\|_\infty$ . Commenter.

### Exercice 2 (Phénomène de Runge)

Soit  $f \in \mathcal{C}^{n+1}([a, b])$ . On rappelle le résultat :

$$\forall x \in [a, b], \exists \zeta \in [a, b] ; f(x) - p_n(x) = \frac{f^{(n+1)}(\zeta)}{(n+1)!} \prod_{i=0}^n (x - x_i).$$

- a) Dans le fichier `tp1_fonction.sci` créer la procédure

```
function PlotErrInterpUnif(f,a,deg,prec)
```

qui, après avoir déterminé le polynôme d'interpolation uniforme  $p_k$ , calcule l'erreur d'interpolation  $\|f - p_k\|_\infty$  sur `[-a,a]` en utilisant la fonction `norm` avec un pas de précision `prec`, pour les degrés  $k$  prescrits dans le vecteurs `deg`. Puis tracer ces erreurs (en échelle log) en fonctions du degré.

---

<sup>1</sup>c'est-à-dire, dont les points d'interpolations sont équidistants.

Par exemple la commande `PlotErrInterpUnif(f,3,[10 :20],.1)` doit calculer et tracer les erreurs d'approximation (calculées avec un pas de précision `.1`) de `f` sur `[-3,3]` pour les polynômes  $p_{10}, \dots, p_{20}$ .

- b) Créer la fonction  $f_2(x) = \sin(\pi x)$ , appelé `f2`, dans le fichier `tp1_exo2.sce`. N'oublier pas d'initialiser ce fichier au début, comme vous l'avez fait pour `tp1_exo1.sce`.
- c) Tracer cette fonction sur  $[-2, 2]$  ainsi que ses polynômes interpolations pour les degrés  $k = 20, \dots, 30$  sur le même graphique.
- d) Tracer l'erreur d'interpolation uniforme de  $f_2$  pour les degrés  $[0, 1, \dots, 50]$ . Est-ce prévisible ? Quelle est le résultat théorique qu'on peut annoncer.
- e) Soit  $f_3 = \frac{1}{1+x^2}$ . Refaire les même calculs que pour  $f_2$ .
- f) Puis élargir l'intervalle de  $[-2, 2]$  à  $[-5, 5]$ . Que constate t-on ? Comment l'expliquer ?

### Exercice 3 (Abscisses de Tchebychev)

Étant donné un entier  $n \in \mathbb{N}^*$  et un intervalle  $[a, b]$ , on rappelle que les  $n$  abscisses de Tchebychev dans  $[a, b]$  sont donnés par :

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \hat{x}_i, \text{ où } \hat{x}_i = \cos\left(\frac{(2i+1)}{2n}\pi\right), \quad i = 0, \dots, n-1.$$

- a) Dans le fichier `tp1_fonction.sci` créer la procédure

```
function [v]=tcheb(a,b,n)
```

qui renvoie les `n` abscisses de Tchebychev dans `[a,b]`.

- b) Modifier (copier/coller + renommer + modifier) les fonctions

`PlotInterpUnif`  $\longrightarrow$  `PlotInterpTcheb`

`PlotErrInterpUnif`  $\longrightarrow$  `PlotErrInterpTcheb`

de sorte que l'interpolation se fasse au niveau des abscisses de Tchebychev.

- c) Étudier comme dans l'exercice précédent la convergence des interpolations de Tchebychev pour la fonction  $f_3$ . Que constatez-vous ?