Université de Lille 1 Sciences et Technologies — UFR de Mathématiques

2014-2015, Licence 3ème année

parcours Mathématiques

M66, Modélisation et analyse numérique

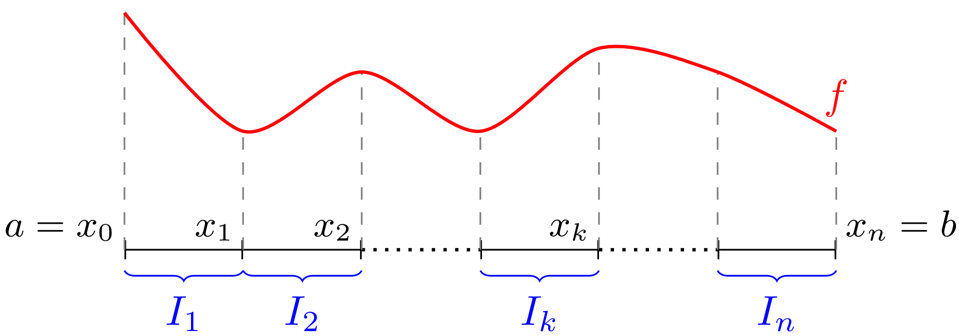
TP2 : Interpolation par morceaux et splines

Vous êtes invité à créer le fichier `tp2_fonction.sci` qui sera complété au fur et à mesure avec les nouvelles procédures. Pour chaque exercice, il va falloir créer un fichier, `tp2_exo0.sce`, ...,`tp2_exo3.sce`, comportant les lignes de code correspondant à la résolution de l'exercice et incluant au début le fichier `tp2_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

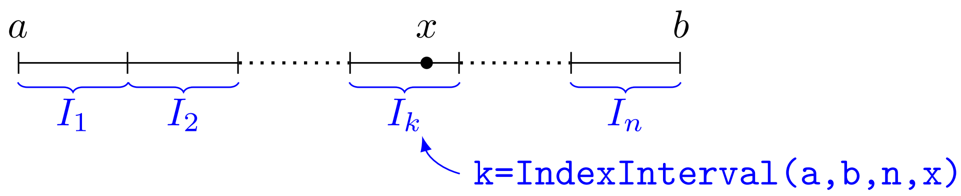
On se propose ici de procéder à l'interpolation d'une fonction f définie sur un intervalle $I = [a, b]$ par utilisation de fonctions polynomiales par morceaux. On découpe donc l'intervalle en n sous-intervalles de même longueur $h = (b-a)/n$ définis par $I_i = [x_{i-1}, x_i]$, avec $x_i = a+ih$, $1 \leq i \leq n$. Sur chaque I_i, on approchera f par un polynôme π_i de degré fixé (ce degré va varier selon les exercices).



Exercices

Exercice 0 (Détermination de l'intervalle)

Avant de procéder aux différents interpolations, on va créer une fonction utilitaire. Cette fonction nous permettra étant donné un $x \in [a, b[$ de déterminer le sous intervalle d'appartenance.



Pour cela, créez dans le fichier `tp2_fonction.sci` une fonction `[k]=IndexInterval(a,b,n,x)`. Cette fonction doit satisfaire les conditions suivantes :

— Étant donnés les bornes `a` et `b`, le nombre de sous intervalles `n` et un vecteur `x`, elle doit retourner un vecteur `k`, de la même taille que `x`, qui contient les indices des intervalles contenant les coordonnées de `x`.

— Pour des raisons techniques, on demande en plus que si une coordonnée de `x` est inférieure à `a`, la valeur correspondante rendue doit être `1`, et si elle est supérieure à `b`, elle doit être `n`.

Indications :

a) Avant de débuter la création de la fonction, testez et commentez le code suivant

```
test = [-1.3 -0.5 0.7 1.3 2 3.7];
disp(ceil(test));
disp(max(test,1));
disp(min(test,2));
```

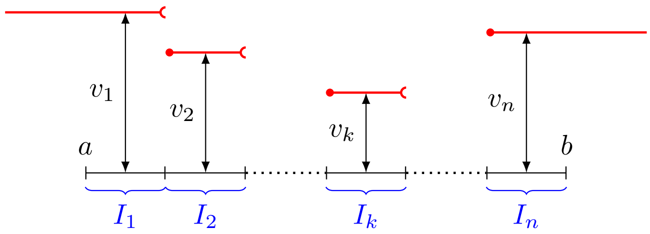
b) Créez d'abord la fonction pour qu'elle fonctionne avec une valeur unique `x`. Puis testez-la par exemple avec avec `IndexInterval(0,1,3,.5)` (le résultat doit être `2`).

c) Une fois la fonction faite pour fonctionner avec un vecteur `x`, vérifiez que l'instruction `IndexInterval(0,1,3,[-1,0,.5,1,2])` renvoie `1. 1. 2. 3. 3.`.

Exercice 1 (Interpolation constante par morceaux)

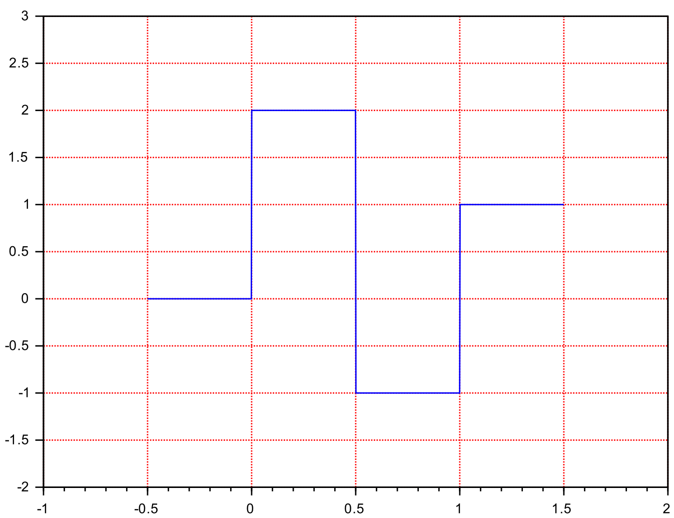
Dans cet exercice, on va approcher une fonction f par une fonction constante par morceaux.

a) Une fonction constante par morceaux sur n sous-intervalles de $I = [a, b]$, de longueur constante, peut être encodée par un vecteur $v = (v_1, \ldots, v_n)$:



Créez la fonction `[y] = ConstPiecewise(a,b,v,x)` qui, étant données les bornes `a`, `b` et les valeurs des hauteurs des paliers `v`, renvoie les valeurs `y` de la fonction constante par morceaux aux points `x`.

Puis pour tester son fonctionnement, dessinez (en utilisant 1000 points) la fonction constante par morceaux sur $[-\frac{1}{2}, \frac{3}{2}]$ qui a pour hauteur des paliers $v = (0, 2, -1, 1)$. Profitez de l'occasion pour rajouter, en plus de la légende, une grille `xgrid(5, 1, 7)` visible sur la boîte de coordonnées `axes.data_bounds = [-1,2,-2,3]`, où `axes=gca()`. Le résultat devrait ressembler à ça :



b) On se propose maintenant de construire une fonction s_0 constante sur chaque intervalle I_i telle que $s_{0|_{I_i}}(x) = f(x_{i-\frac{1}{2}})$, où $x_{i-\frac{1}{2}} = (x_{i-1} + x_i)/2$. En supposant que $f \in C^1(I)$, on rappelle que :

$$||f - s_0||_\infty \leq \frac{h}{2} M_1,$$

où M_1 est un majorant de f' sur I.

(1) Pour cela, créez une fonction `[v,t] = InterpConst(a,b,n,f)` qui étant donnée une fonction `f` renvoie dans `t` les valeurs des $x_{i-\frac{1}{2}}$ pour $i = 1, \ldots, n$, et en `v` les valeurs de f en ces points.

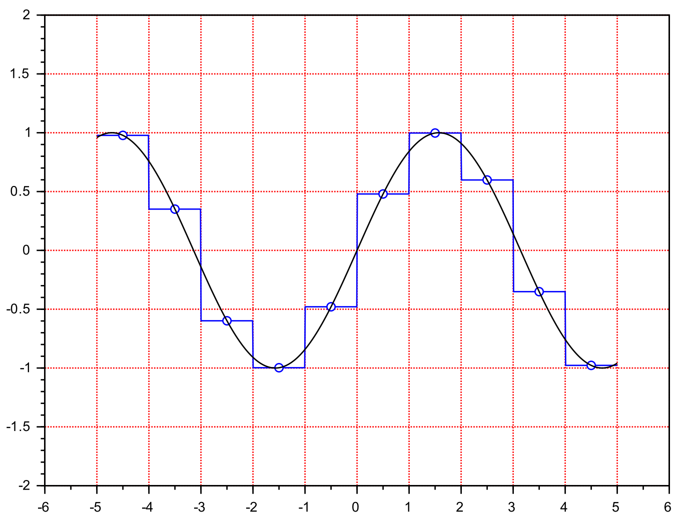
(2) Soit $f = \sin(x)$ sur $[-5, 5]$. En utilisant les fonctions précédemment définies, `InterpConst` et `ConstPiecewise`, dessinez (en utilisant 1000 points) sur un même graphique :

— la fonction f,

— la fonction d'interpolation constante par morceaux s_0,

— les points d'interpolation $(x_{i-\frac{1}{2}}, f(x_{i-\frac{1}{2}}))$.

Le résultat devrait ressembler à ça :

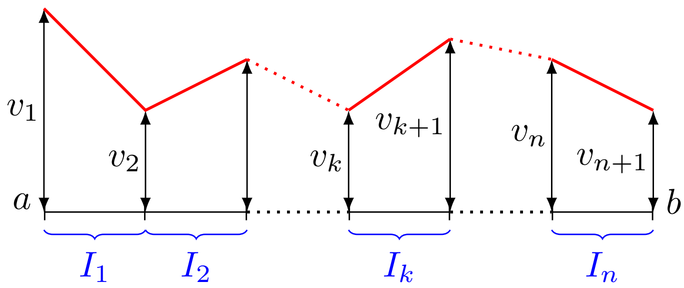


(3) Pour `n=2,4,...,2^10` calculez l'erreur d'interpolation (en norme infinie). Puis sur un même graphique représentez (en échelle logarithmique) cette erreur et la majoration théorique de l'erreur.

Exercice 2 (Interpolation affine par morceaux)

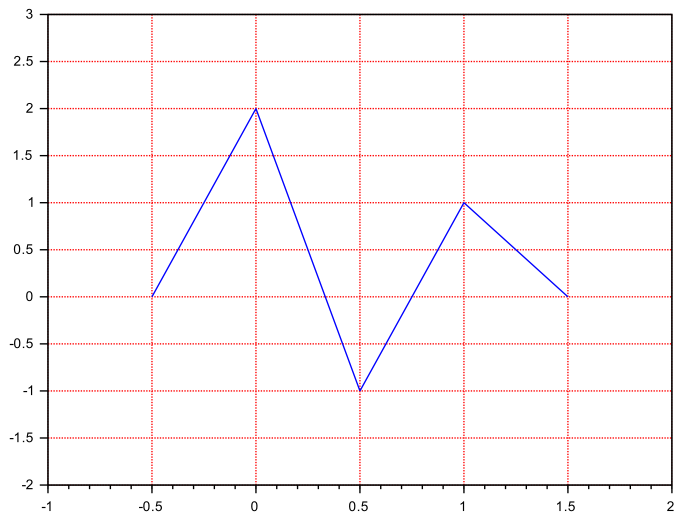
Dans cet exercice, on va approcher une fonction f par une fonction affine par morceaux.

a) Une fonction affine par morceaux sur n sous-intervalles de $I = [a, b]$ de longueur constante, peut être encodée par un vecteur $v = (v_1, \ldots, v_{n+1})$:



Créez la fonction `[y] = AffinePiecewise(a,b,v,x)` qui, étant données les bornes `a`, `b` et les valeurs aux nœuds `v`, renvoie les valeurs `y` de la fonction affine par morceaux aux points `x`.

Puis pour tester son fonctionnement, dessinez (en utilisant 1000 points) la fonction affine par morceaux sur $[-\frac{1}{2}, \frac{3}{2}]$ qui a pour valeurs aux nœuds $v = (0, 2, -1, 1, 0)$. Le résultat devrait ressembler à ça :



b) On se propose maintenant de construire la fonction s_1 affine sur chaque intervalle I_i et qui coïncide avec f aux points x_i et x_{i+1} :

$$s_{1|_{I_i}}(x) = \frac{(x_{i+1} - x)f(x_i) + (x - x_i)f(x_{i+1})}{h}.$$

En supposant que $f \in C^2(I)$, on rappelle que :

$$||f - s_1||_\infty \leq \frac{h^2}{8} M_2,$$

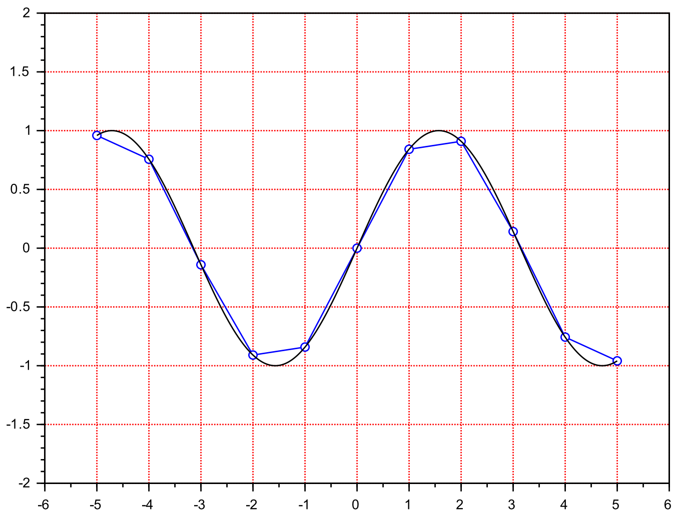
où M_2 est un majorant de f'' sur I.

On va suivre le même schéma que dans l'exercice précédent.

(1) Créez une fonction `[v,t] = InterpAffine(a,b,n,f)` qui étant donnée une fonction `f` renvoie dans `t` les valeur des x_i pour $i = 0, \ldots, n$, et en `v` les valeurs de f en ces points.

(2) Soit $f = \sin(x)$ sur $[-5, 5]$. En utilisant les fonctions précédemment définies, `InterpAffine` et `AffinePiecewise`, dessinez (en utilisant 1000 points) sur un même graphique :
- la fonction f,
- la fonction d'interpolation affine par morceaux s_1 sur 10 intervalles,
- les points d'interpolation $(x_i, f(x_i))$.

Le résultat devrait ressembler à ça :



(3) Pour `n=2,4,...,2^10` calculez l'erreur d'interpolation (en norme infinie). Puis sur un même graphique, représentez (en échelle logarithmique) cette erreur et la majoration théorique de l'erreur.

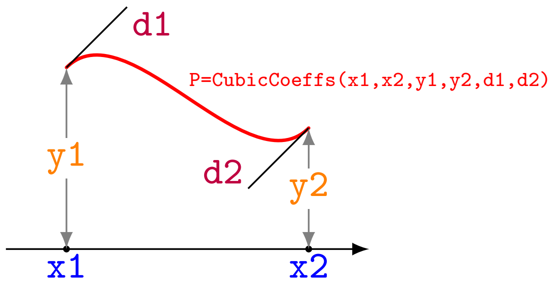
Exercice 3 (Interpolation par splines)

Dans cet exercice, on va approcher une fonction f par une spline cubique (fonction C^2 polynomiale de degré 3 par morceaux).

a) Dans cette question on va déterminer les coefficients d'un polynôme cubique à partir des données d'interpolation d'Hermite. Pour commencer, écrire une procédure

`[p] = CubicCoeffs(x1,x2,y1,y2,d1,d2)`

qui détermine les coefficients `p` du polynôme $P(x) = p_1 + p_2 x + p_3 x^2 + p_4 x^3$ qui a pour valeurs `y1` en `x1` et `y2` en `x2`, et dont les dérivées dans ces deux points sont respectivement `d1` et `d2`. Pour vérifier que votre procédure fonctionne, testez-la avec `CubicCoeffs(0,1,2,3,4,5)` et le résultat devrait être `2. 4. - 10. 7.` (éventuellement en colonne).



b) Écrire la procédure `[y] = CubicEval(p,x)` qui, étant donnés les coefficients `p` d'un polynôme et un vecteur `x`, évalue ce polynôme en les coordonnées de `x`. Pour vérifier que votre procédure fonctionne, testez-la avec `CubicEval([1:4],[-1:1])` et le résultat devrait être `- 2. 1. 10.`.

c) En utilisant les procédures `CubicCoeffs` et `CubicEval`, dessinez la base d'Hermite sur $[0,1]$, c'est-à-dire les 4 polynômes qui ont une seule des valeurs $P(0)$, $P(1)$, $P'(0)$, $P'(1)$ non nulle, égale à 1.

d) En utilisant les procédure `CubicCoeffs` et `CubicEval`, dessinez les polynômes P sur $[0,1]$ tels que $P(0) = 0$, $P(1) = 1$, $P'(1) = -7$ et $P'(0)$ prend les valeurs entières de -10 à 10.

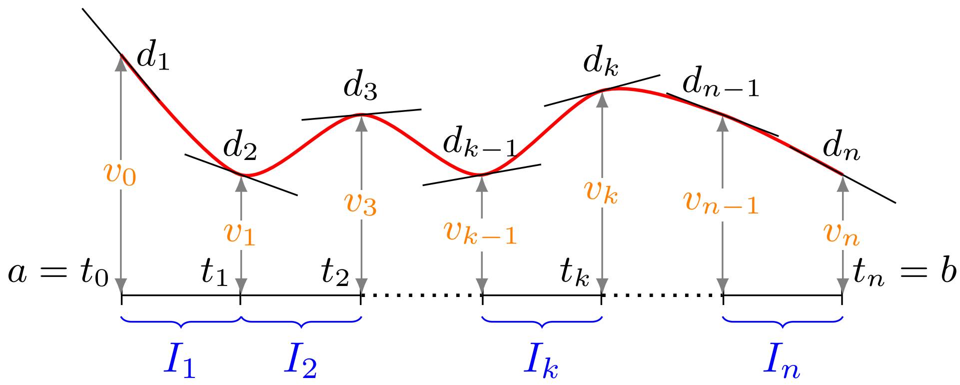
e) Il est temps maintenant de créer la procédure qui évalue la spline cubique d'interpolation. Pour cela créez la procédure `[y] = CubicSplin(a,b,v,x)` qui a pour entrées :

— Les bornes `a` et `b` de l'intervalle d'interpolation.

— Les valeurs `v=[v(1), v(2), ..., v(n+1)]` d'interpolation aux $n+1$ nœuds équi-distribués.

— Les points `x` dans lesquels il faut faire l'évaluation.

Cette procédure doit renvoyer les valeurs `y` de la spline naturelle aux points `x`.



La procédure `CubicSplin` devra faire dans l'ordre :

— Créez le vecteur `t` des $n+1$ nœuds équi-distribués d'interpolation.

— En utilisant `d = splin(t,v,"natural")`, déterminez les valeurs des dérivées aux nœuds d'interpolation.

— En utilisant `IndexInterval` de l'exercice **1**, déterminez les intervalles dans lesquels se trouvent les points `x`.

— En utilisant `CubicCoeffs` et `CubicEval`, déterminez les valeurs des `y` à partir des informations précédemment obtenues.

Pour vérifier que votre procédure fonctionne, testez-la avec

```
CubicSplin(0,1,[0,2,1],[0:.2:1])
```

et le résultat devrait être

```
0. 1.052 1.816 2.016 1.652 1.
```

f) Reprendre la question 2.(b) de l'exercice **2**, pour représenter la fonction $\sin(x)$ sur $[-5, 5]$, ainsi que la spline cubique naturelle qui l'interpole sur 10 intervalles de longueur constante. Pour cela normalement il suffit de remplacer la fonction `AffinePiecewise` par `CubicSplin`.

g) Modifiez la fonction `CubicSplin` (ou créez une nouvelle fonction `CubicSplinClamped`) qui permet à la place de `splin(t,v,"natural")` d'évoquer `d = splin(t,v,"clamped",bd)` pour obtenir une interpolation par spline tendue avec des dérivées aux bords `bd(1)` en `a`, et `bd(2)` en `b`.

Puis rajouter le graphe du spline cubique tendu d'interpolation de $\sin(x)$ sur le graphique de la question précédente.

h) Pour `n=2,4,...,2^10` calculez l'erreur d'interpolation par splines cubiques naturelles et tendues (en norme infinie). Puis représentez-les sur un même graphique (en échelle logarithmique).

i) Dessinez la base cardinale des splines cubiques sur les nœuds $\{0, 1, 2, 3\}$.