TD 7 Approximations polynomiales

I Développements limités

a. Formule de Taylor-Young

Allez, je ne vous fais pas l'offense de vous la rappeler...Nous allons créer une procédure MAPLE qui donnera la partie polynomiale d'un DL en utilisant cette formule :

Entrées : une expression f(x) et un ordre n
Initialisation :
Der⟵ f (sous forme de fonction !)
Pol⟵ 0
début
  pour *k allant de 0 à n* **faire**
    Pol⟵ Pol+Der(0)$\times\frac{x^k}{k!}$
    Der⟵ fonction dérivée de Der
fin
retourner Pol

Algorithme 1 : Partie polynomiale d'un DL

Yaka écrire une procédure `DL :=proc(f,n)` en MAPLE...

> **Remarque 1 : syntaxe MAPLE**
>
> MAPLE a une commande pour ça. Il suffit d'utiliser `taylor(expression,var=point,ordre(en option))`
> L'ordre par défaut est 6. Attention ! MAPLE utilise des O et non des o.
> Pour les développements asymptotiques, on utilise plutôt : `asympt(expression, variable)`

Utiliser la procédure `DL` pour avoir un ordre de grandeur de l'approximation pour différentes fonctions et différentes valeurs de x.
Par exemple :

```
evalf(unapply(DL(ln(1+x),5),x)(0.001)-ln(1+0.001),20)
evalf(unapply(DL(ln(1+x),5),x)(1.1)-ln(1+1.1))
```

Essayez plutôt de présenter les résultats dans un tableau avec plusieurs valeurs de x en colonnes et plusieurs fonctions en lignes.

b. Visualisation de l'approximation locale par une fonction polynôme

Créez une procédure `graph :=proc(f,xmax,ymax,p)` dans laquelle vous construirez une séquence de `plot(DL(f,i))` pour i variant de 1 à p et vous ajouterez la représentation graphique de f avec une couleur différente.

Pour visualiser de manière dynamique l'approximation, on utilisera `display` de la bibliothèque `plots` avec l'option `insequence=true` qui permet d'animer le graphique. Pour cela, il faut cliquer sur le graphique : des icônes apparaissent. Par exemple, regardez et commentez ce que donne :

```
graph(x->cos(x),15,1,50);
graph(x->ln(1+x),3,3,50);
```

II Polynôme d'interpolation de Lagrange

a. À quoi ça sert ?

D'un point de vue abstrait, on pourrait se contenter de dire que si on ne connaît que quelques points de la courbe représentative d'une fonction inconnue, on cherche à approcher cette fonction par une fonction polynomiale passant par ces points. Intuitivement, on peut penser que plus on aura de points de contrôles, meilleures sera l'approximation mais les mathématiques vont venir une nouvelle fois contredire notre intuition.
D'un point de vue plus concret, on voudrait reconstituer un organe humain à partir de quelques mesures prises en imagerie médicale ou reconstituer une couche géologique à partir de mesures sismiques : on est alors amené à reconstituer une formes à partir de nuages de points, en 3D cette fois.
Ces problèmes sont le quotidien de l'ingénieur. Nous nous contenterons bien sûr ici d'un survol de quelques méthodes très simples dans des cas théoriques eux aussi assez triviaux.

b. Cas général

On considère n points de coordonnées $(x_1, y_1) \dots (x_n, y_n)$ On veut trouver un polynôme P tel que $P(x_i) = y_i$ pour tout $i \in [\![1\,;\,n]\!]$.
Une petite activité mathématique consiste à prouver que l'unique polynôme de degré $n-1$, solution du problème, est défini par

$$P(x) = \sum_{k=1}^{n} y_k \mathcal{L}_k(x) \quad \text{avec} \quad \mathcal{L}_k(x) = \frac{\prod_{i=1, i\neq k}^{i=n} (x - x_i)}{\prod_{i=1, i\neq k}^{i=n} (x_k - x_i)}$$

Déterminez une procédure `Lagrange :=proc(X,Y,x)` où X est la liste des abscisses, Y celle des ordonnées et x la variable.
Le problème, c'est ce $i \neq k$. Il faut donc user d'astuce...
On construit un polynôme intermédiaire : $L \leftarrow \prod_{i=1}^{n}(x - x_i)$
On enlève le terme en trop en divisant par $(x - x_k)$: $L \leftarrow L/(x - x_k)$
Pour cela on utilise la commande `quo(poly1,poly2,variable)` qui calcule le quotient de deux polynômes.
On divise par la même chose mais en substituant x par x_k : $L \leftarrow L/subs(x = x_k, L)$
Vous ferez bon usage des fonctions MAPLE suivantes :
- `product(expression(k),k=debut..fin);`
- `subs(ancien=nouveau,Liste);`
- `expand(Polynome);`
- `sort(Polynome).`

Ça aura cette forme :

```
Lagrange:=proc(X,Y,t)
 local k,L,P,Produit;
 P:=0:
 Produit:=product(t-X[i],i=1..nops(X));
 for k to nops(X) do
   .
   .
   .
 od:
 sort(expand(P));
 end:
```

Par exemple avec :

```
Lagrange([1,2,3,4,5],[7,-8,9,-10,11],x)
```

on obtient :

Réponse du logiciel

$$6x^4 - \frac{214}{3}x^3 + 294x^2 - \frac{1463}{3}x + 266$$

Vérifiez que $P(1) = 7$, $P(2) = -8$, etc.

c. Application à l'approximation polynomiale d'une fonction

Déterminez une procédure `LagrangeFonc:=proc(f,X,t)` où f est une fonction de $\mathbb{R}$ dans $\mathbb{R}$ et qui renvoie le polynôme d'interpolation de Lagrange associé aux points de coordonnées $(x_1, f(x_1)), \ldots (x_n, f(x_n))$.
Il suffit d'utiliser astucieusement la procédure précédente.
Voyez ce que ça donne par exemple avec

```
LagrangeFonction(t->1/(1+t^2),[1,2,3,4,5],x)
```

d. Visualisation

On veut à présent visualiser l'approximation du graphe d'une fonction par des polynômes de Lagrange successifs en augmentant les points de contrôle.
On va par exemple prendre une subdivision régulière de l'intervalle et on augmentera le nombre de points de contrôles pour observer si l'approximation s'en trouve améliorée.
Construisez une animation de l'approximation de la fonction $x \mapsto \frac{1}{1+x^2}$ sur $[-5; 5]$ avec comme séries successives d'abscisses de points de contrôle des subdivisions régulières de l'intervalle $[-5\,;\,5]$.
Faites de même pour $x \mapsto \ln(1+x)$ en adaptant les intervalles.

On a bien sûr envie de généraliser. Comme je suis gentil, voici une solution

```
LagrangeGraphe:=proc(g,n,a,b,ymin,ymax)
local i,s,p;
s:=seq(plots[display](plot(LagrangeFonction(g,[a+(b-a)*i/p$i=0..p],t),t=a..b),
       plot(g(t),t=a..b,color=blue)),p=1..n):
plots[display](s,insequence=true,view=[a..b,ymin..ymax]);
end:
```

Ce qui donne

```
LagrangeGraphe(t->1/(1+t^2),40,-5,5,t,-0.2,1.2);
LagrangeGraphe(t->ln(1+t),20,0,5,t,-2,3);
```

Bref, ce n'est pas aussi magique qu'il n'y paraissait : plus on augmente le nombre de points de contrôle, plus les « effets de bord » deviennent gênants.
Vous prouverez peut-être un jour que si f est définie sur un intervalle I, si la suite des dérivées successives de f est uniformément bornée et si l'on fait tendre le nombre de points de contrôle vers $+\infty$, alors la suite des interpolateurs de Lagrange de f converge *uniformément* vers f sur tout segment $[a\,;\,b]$ inclus dans I lorsque n tend vers $+\infty$. Vous comprendrez l'an prochain toute l'importance du mot *uniformément* qui permet d'éviter ces désagréables « effets de bord ». Ça n'a malgré tout que peu d'intérêt car ces conditions sont trop restrictives et de toute façon, la formule de Taylor-Lagrange nous donne ce qu'il faut pour de telles fonctions.

e. Polynômes de Tchebychev

Il existe néanmoins un résultat positif pour les fonctions de classe $\mathcal{C}^1$ sur $[-1\,;\,1]$: il suffit de prendre pour abscisses des points de contrôle les racines des polynômes de Tchebychev et alors la convergence est uniforme. Oui, mais qu'est-ce qu'un polynôme de Tchebychev ?

Polynôme de Tchebychev

On définit les polynômes de Tchebychev d'ordre n par

$$T_n(x) = \cos\left(n\,\mathrm{Arccos}(x)\right)$$

En fait, ce sont les polynômes qui vérifient $T_n(\cos(t)) = \cos(nt)$.
On peut les déterminer à la main ou utiliser `orthopoly[T](n,x)` qui donne $T_n(x)$.

Observez par exemple :

```
T(5,x);
S:=solve(T(5,x)=0,x);
evalf(S);
```

Étudiez alors quelques exemples (les fonctions doivent être définies sur $[-1\,;\,1]$) mais avec comme abscisses les racines des polynômes de Tchebychev pour différentes valeurs de n. Qu'observez-vous ? On construira une procédure `LagrangeTcheb:=proc(g,n,ymin,ymax)`

```
LagrangeTcheb:=proc(g,n,ymin,ymax)
 local i,s,p;
 s:=seq( plots[display](
    plot ( LagrangeFonction(g,[evalf(solve(T(p,x)=0,x))],t),t=-1..1),
    plot(g(t),t=-1..1,color=blue)),
    p=1..n):
 plots[display](s,insequence=true,view=[-1..1,ymin..ymax]);
 end:
```

Il y a bien sûr des approximations plus intéressantes

- par les polynômes de Bernstein définis par $B_n(x) = \sum_{k=0}^{n} \binom{n}{k} f\left(\frac{k}{n}\right) x^k (1-x)^{n-k}$ pour des fonctions continues sur un segment
- par des polynômes trigonométriques dans des cas encore plus généraux comme vous le verrez en étudiant les séries de Fourier l'an prochain.