

# TP Maple 6

## Dérivées, études locales et EDO

Le logiciel va nous permettre, au-delà des outils graphiques, de calculer des limites, des développements limités et des développements asymptotiques, d'effectuer des calculs de dérivées et de résoudre des équations différentielles (de manière exacte ou approchée).

1	Dérivées totales et partielles	1
1.1	Dériver des expressions : les opérateurs <b>diff</b> et <b>Diff</b>	1
1.2	Dériver des fonctions : l'opérateur <b>D</b>	2
2	Etude locale des fonctions numériques	3
2.1	Limites	3
2.2	Développements de Taylor	3
2.3	Développements asymptotiques	4
2.4	Extraction de la partie principale ou d'un équivalent	4
3	Equations différentielles	5
3.1	La commande <b>dsolve</b>	5
3.2	Système d'équations différentielles	6
3.3	Développement limité d'une solution	7
3.4	Résolution numérique	7
4	Exercices	10

### 1. Dérivées totales et partielles

Comme toujours, on distinguera les fonctions et les expressions. Cela se traduira par deux syntaxes distinctes.

#### 1.1. Dériver des expressions : les opérateurs **diff** et **Diff**

Afin de présenter avec élégance des séquences de calcul différentiel, Maple dispose d'un opérateur *effectif* de différenciation et d'un opérateur *inerte* de différenciation. L'opérateur *effectif* **diff** effectuera et affichera directement les calculs demandés alors que sa version *inerte* **Diff** effectuera les calculs sans les afficher. Il faudra forcer l'affichage du résultat par la commande **value**.

```
> E:=x^3*sin(x): diff(E,x), Diff(E,x); value(Diff(E,x));
```

$$3x^2 \sin(x) + x^3 \cos(x), \quad \frac{d}{dx}(x^3 \sin(x)), \quad \sin(x) + x^3 \cos(x)$$

Bien-sûr, les dérivées successives sont également calculables. Il y a même plusieurs syntaxes synonymes :

```
> diff(E,x,x,x): diff(E,[x,x,x]): diff(E,x$3):
```

La commande `diff(E,[ ])` donne la dérivation à l'ordre zéro, c'est-à-dire  $E$ .

### Dérivées partielles

```
> F:=x/(x^2+y^2): Diff(F,x,y)=diff(F,x,y);
```

$$\frac{\partial^2}{\partial y \partial x} \left( \frac{x}{x^2 + y^2} \right) = -\frac{2y}{(x^2 + y^2)^2} + 8\frac{x^2 y}{(x^2 + y^2)^3}$$

```
> Diff(F,y$2,x)=diff(F,y$2,x);
```

$$\frac{\partial^3}{\partial x \partial^2 y} \left( \frac{x}{x^2 + y^2} \right) = 8\frac{y^2}{(x^2 + y^2)^3} - 48\frac{x^2 y^2}{(x^2 + y^2)^4} - \frac{2}{(x^2 + y^2)^2} + 8\frac{x^2}{(x^2 + y^2)^3}$$

### 1.2. Dériver des fonctions : l'opérateur D

Dans le cas d'une fonction, c'est l'opérateur **D** qui permettra de calculer les dérivées successives.

— Dérivations formelles successives d'une fonction  $f$  —

- ▶ **D(f)** : renvoie la fonction dérivée de  $f$ .
- ▶ **(D@@n)(f)** : renvoie la fonction dérivée  $n$ -ième de  $f$  pour  $n \in \mathbb{N}^*$ .
- ▶ **D[i<sub>1</sub>,i<sub>2</sub>,...,i<sub>n</sub>](f)** retourne la fonction  $\frac{\partial^n f}{\partial x_{i_n} \dots \partial x_{i_2} \partial x_{i_1}}$

```
> f:=x->exp(x)*sin(x): D(f); (D@@2)(f);
```

$$x \rightarrow e^x \sin(x) + e^x \cos(x), \quad x \rightarrow 2e^x \cos(x)$$

```
> g:=(x,y,z)->z*cos(x*y): D[](g); D[1,1,1](g), D[2](g), D[1,2](g);
```

$$g, (x,y,z) \rightarrow z \sin(xy)y^3, (x,y,z) \rightarrow -z \sin(xy)x, (x,y,z) \rightarrow -z \cos(xy)yx - z \sin(xy)$$

```
> D[1,1,1](g)(Pi,1/2,1);
```

$$\frac{1}{8}$$

Signalons que l'on peut aussi dériver une fonction  $f$  en appliquant l'opérateur `diff` à l'expression <sup>1</sup>  $f(x)$ .

1. Ceci n'est bien entendu valable que si la variable  $x$  est libre.

## 2. Etude locale des fonctions numériques

### 2.1. Limites

Le logiciel dispose d'un opérateur inerte (**Limit**) et effectif (**limit**) de calcul des limites. Pour calculer  $\lim_{x \rightarrow a} f(x)$  où  $f(x)$  est une expression, on suivra la syntaxe décrite ci-dessous.

Calcul d'une limite par la commande **limit**

**limit(f(x), x=a, right ou left)**

le troisième argument (pour une limite à droite ou à gauche) étant optionnel.

On écrira  $a = \pm\text{infinity}$  pour calculer des limites en  $\pm\infty$ .

```
> Limit(exp(1/x), x=0, left)=limit(exp(1/x), x=0, left);
Limit(exp(x)/x, x=infinity): %=value(%);
```

$$\lim_{x \rightarrow 0^-} e^{\frac{1}{x}} = 0, \quad \lim_{x \rightarrow +\infty} \left( \frac{e^x}{x} \right) = \infty$$

### 2.2. Développements de Taylor

MAPLE calcule des développements de Taylor dont les restes sont des  $O$  (« grand o »); par exemple

$$\sin(x) = x + O(x^3)$$

Calcul d'un développement limité de  $f(x)$  en un point  $a$  par la commande **taylor**

**taylor(f(x), x = a, n)**

Il faut bien comprendre la signification de  $n$  ci-dessus : il ne s'agit pas de l'ordre du développement calculé mais l'ordre auquel le logiciel développera les fonctions qui constituent l'expression  $f(x)$ . Mais lorsqu'un meilleur résultat est déjà calculé, MAPLE s'en rappelle... Le lecteur méditera l'exemple suivant :

```
> E:=sin(x)/ln(x+1): taylor(E, x=0, 4); taylor(E, x=0, 5); taylor(E, x=0, 4);
```

$$1 + \frac{1}{2}x - \frac{1}{4}x^2 + O(x^3)$$

$$1 + \frac{1}{2}x - \frac{1}{4}x^2 - \frac{1}{24}x^3 + O(x^4)$$

$$1 + \frac{1}{2}x - \frac{1}{4}x^2 - \frac{1}{24}x^3 + O(x^4)$$

### 2.3. Développements asymptotiques

Plus généralement, l'utilisateur obtiendra des développements asymptotiques<sup>2</sup> au moyen de la commande `series`.

Calcul d'un développement asymptotique de  $f(x)$  en un point  $a$  par la commande `series`

`series(f(x), x = a, n)`

La remarque précédente sur la signification de  $n$  est encore valable pour la commande `series`. Ce troisième argument est cependant optionnel. Maple dispose d'une variable d'environnement `Order` qui par défaut vaut 6.

```
> series(1/sin(x), x=0, 4);
```

$$x^{-1} + \frac{1}{6}x + O(x^2)$$

```
> Order:=2:series(sin(x), x=0);
```

$$x + O(x^2)$$

La commande `taylor` n'étant qu'une spécialisation de la commande `series` on pourra également obtenir des *DL* au moyen de `series`. L'avantage de `series` est qu'on peut aussi effectuer des développements en  $\pm\infty$ .

```
> series(1/(1+x), x=infinity, 4);
```

$$\frac{1}{x} - \frac{1}{x^2} + \frac{1}{x^3} + O\left(\frac{1}{x^4}\right)$$

### 2.4. Extraction de la partie principale ou d'un équivalent

Le résultat renvoyé par la commande `series` n'est pas utilisable tel quel pour un tracé ou pour effectuer des calculs algébriques (produits ou sommes de *DL*) car il n'est pas du type *polynom* (au sens large). On pourra cependant *extraire* la partie principale d'un développement en utilisant la commande `convert`.

Extraction de la partie principale d'un développement

`convert (résultat d'une commande series , polynom)`

```
> series(60*ln(1+x), x = 0); convert(%, polynom);
```

$$60x - 30x^2 + 20x^3 - 15x^4 + 12x^5 + O(x^6)$$

$$60x - 30x^2 + 20x^3 - 15x^4 + 12x^5$$

2. Ce type de développement généralise les *DL*. Le lecteur est renvoyé à son cours de Mathématiques.

On obtiendra un équivalent d'une expression au voisinage d'un point en utilisant la command `leadterm`.

Détermination d'un équivalent de  $x$  en  $a$  par `leadterm`

`series(leadterm( $f(x)$ ),  $x = a$ )`

```
> series(leadterm(sqrt(1+sqrt(x))),x=infinity);
```

$$\frac{1}{\left(\frac{1}{x}\right)^{\frac{1}{4}}}$$

### 3. Equations différentielles

Le logiciel peut résoudre explicitement de nombreuses équations différentielles. En cas d'impossibilité, l'utilisateur pourra se rabattre sur des solutions de rechange comme une résolution approchée ou la recherche de solutions développables en série.

#### 3.1. La commande `dsolve`

La commande `dsolve` renvoie les solutions d'une équation différentielle (écrite au moyen de la commande `diff`) en écrivant les constantes sous la forme `_C1`, `_C2`, etc.

```
> Eq:=diff(y(x),x)+y(x)=exp(-x); dsolve(Eq,y(x));
```

$$Eq := \frac{d}{dx}y(x) + y(x) = e^{-x}, \quad y(x) = (x + _C1)e^{-x}$$

On pourra aussi préciser des conditions initiales :

```
> dsolve({Eq,y(0)=0},y(x));
```

$$y(x) = e^{-x}x$$

L'utilisateur prendra garde à ce que les séquences précédents n'ont nullement affecté les variables  $y$  ou  $y(x)$  : la solution n'a été retenue ni sous forme d'une fonction ni sous forme d'une expression.

```
> y(0);
```

$$y(0)$$

S'il souhaite conserver la solution comme une fonction  $y$ , l'utilisateur suivra la syntaxe suivante :

— Résolution formelle d’une équation différentielle —

- ▶ *Solution générale (avec constante) : dsolve(équation)*
- ▶ *Solution à un problème de Cauchy : dsolve({équation, cond. init.})*
- ▶ *Enregistrer la solution comme une fonction :*  

$$\text{sol} := \text{dsolve}(\{\text{équation}, \text{cond. init.}\}, y(x)) : \text{assign}(\text{sol}) : y := \text{unapply}(y(x), x) :$$
- ▶ *Enregistrer la solution comme une expression :*  

$$\text{sol} := \text{dsolve}(\{\text{équation}, \text{cond. init.}\}, y(x)) : y := \text{rhs}(\text{sol}) :$$

```
> eq:=diff(y(x),x)-y(x)=exp(x): sol:=dsolve({eq,y(0)=1},y(x)): assign(sol):
y:=unapply(y(x),x): y(0);
```

1

Signalons que Maple ne peut pas résoudre toutes les équations !...

```
> equ:=diff(y(x),x$2)+x*diff(y(x),x)+(x^2+1)*y(x)=0;

$$\frac{d^2}{dx^2}y(x) + x\left(\frac{d}{dx}y(x)\right) + (x^2 + 1)y(x) = 0$$

> sol:=dsolve({equ,y(0)=1,D(y)(0)=1},y(x));
sol :=
```

Le logiciel reste muet ; il faudra alors envisager d’autres moyens...

### 3.2. Système d’équations différentielles

La syntaxe précédente se généralise sans peine à un système d’équations différentielles de la manière suivante :

— Résolution formelle d’un système d’équations différentielles —

$$\text{dsolve}(\{\text{équation}_1, \text{équation}_2, \dots, \text{équation}_n, \text{conditions initiales}\})$$

On pourra également extraire les solutions en tant que fonctions. Voici l’exemple d’un système linéaire homogène d’ordre deux.

```
> eqq:={diff(x(t),t)+x(t)+2*y(t)=0,diff(y(t),t)+x(t)-y(t)=0,x(0)=0,y(0)=1}:
sol:=dsolve(eqq);
```

$$\text{sol} := \left\{ x(t) = -\frac{1}{3}\sqrt{3}e^{\sqrt{3}t} + \frac{1}{3}\sqrt{3}e^{-\sqrt{3}t}, y(t) = \frac{1}{2}e^{\sqrt{3}t} + \frac{1}{2}e^{-\sqrt{3}t} + \frac{1}{6}\sqrt{3}e^{\sqrt{3}t} - \frac{1}{6}\sqrt{3}e^{-\sqrt{3}t} \right\}$$

```
> x:=unapply(subs(sol,x(t)),t);
```

$$x := t \rightarrow -\frac{1}{3}\sqrt{3}e^{\sqrt{3}t} + \frac{1}{3}\sqrt{3}e^{-\sqrt{3}t}$$

### 3.3. Développement limité d'une solution

Le logiciel peut trouver un *DL* de la solution à une équation différentielle donnée au voisinage d'une condition initiale fixée. L'utilisateur explicitera l'option *series* en troisième argument de la commande *dsolve*.

```
> equ:=diff(y(x),x$2)+x*diff(y(x),x)+(x^2+1)*y(x)=0:
sol:=dsolve({equ,y(0)=1,D(y)(0)=1},y(x),series);
```

$$\text{sol} := y(x) = 1 + x - \frac{1}{2}x^2 - \frac{1}{3}x^3 + \frac{1}{24}x^4 + \frac{1}{60}x^5 + O(x^6)$$

L'ordre du développement<sup>3</sup> est par défaut égal à 6. On pourra le changer en modifiant la valeur de la variable globale **Order** avant d'utiliser la commande *dsolve*.

```
> Order:=8: sol:=dsolve({equ,y(0)=1,D(y)(0)=1},y(x),series);
```

$$\text{sol} := y(x) = 1 + x - \frac{1}{2}x^2 - \frac{1}{3}x^3 + \frac{1}{24}x^4 + \frac{1}{60}x^5 + \frac{7}{720}x^6 + \frac{1}{180}x^7 + O(x^8)$$

### 3.4. Résolution numérique

Lorsque le logiciel s'avérera incapable d'exprimer formellement une solution, on pourra tout de même obtenir une solution approchée obtenue par une méthode numérique choisie par Maple<sup>4</sup>. Il faudra alors préciser l'option *numeric* en troisième argument de la commande *dsolve*.

```
> equ:=diff(y(x),x$2)+x*diff(y(x),x)+(x^2+1)*y(x)=0;
sol:=dsolve({equ,y(0)=1,D(y)(0)=1},y(x),numeric);
```

```
sol := proc(xrkf45)...end proc
```

La variable *sol* se comporte comme une fonction de la variable *x* et donc la valeur en *x* est une liste de la forme \_\_\_\_\_

3. Il s'agit d'un *DL* en  $\mathcal{O}$ , attention au décalage par rapport au cours de Mathématiques !

4. Citons par exemple la méthode de *Runge-Kutta-Fehlberg*, *rkf* en abrégé.

$[x, \text{valeur approchée de } y(x), \text{valeur approchée de } y'(x), \dots, \text{valeur approchée de } y^{(n-1)}(x)]$

dans le cas d'une équation d'ordre  $n$ . Ainsi, dans l'exemple exposé ci-dessus :

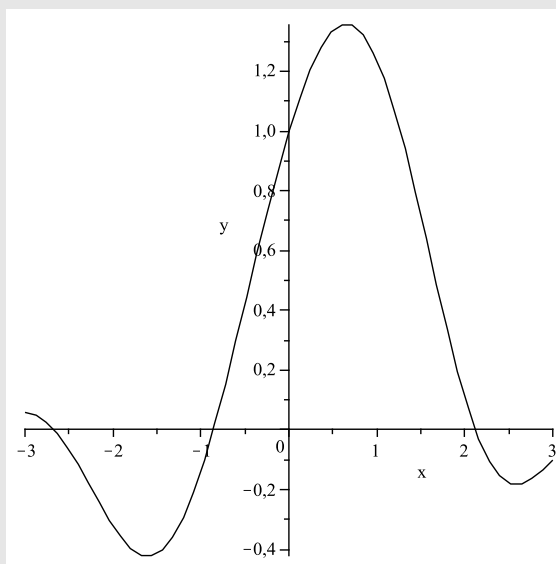
```
> sol(0);
```

$$\left[ x = 0., y(x) = 1., \frac{d}{dx}y(x) = 1. \right]$$

### Représentation graphique

La fonction `odeplot` de la bibliothèque `plots` permettra au lecteur de représenter graphiquement la solution approchée calculée par Maple.

```
> with(plots): odeplot(sol, [x,y(x),color=black], -3..3);
```



On a précisé  $[x, f(x), \dots]$  en deuxième argument pour forcer le logiciel à tracer  $y(x)$  en fonction de  $x$ .

### Etraction de la solution approchée sous forme d'une fonction

Pour cette opération, on commence par forcer le logiciel à créer  $n$  procédures à partir de la procédure définissant `sol`. Il suffit d'indiquer l'option `output=listprocedure` en quatrième argument de `dsolve`. On peut alors extraire la fonction  $y(x)$  par une commande `subs`.

```
> lst:=dsolve({equ,y(0)=1,D(y)(0)=1},y(x),numeric,output=listprocedure);

lst := [ x = proc(x)...end proc, y(x) = proc(x)...end proc,  $\frac{d}{dx}y(x) = \text{proc}(x)\dots\text{end proc}$  ]

> f:=subs(lst,y(x));

f := proc(x)...end proc
```

La fonction ainsi définie est utilisable par exemple pour des tracés, des équations, etc.

```
> fsolve(f(x)=0,x=2..3);
```

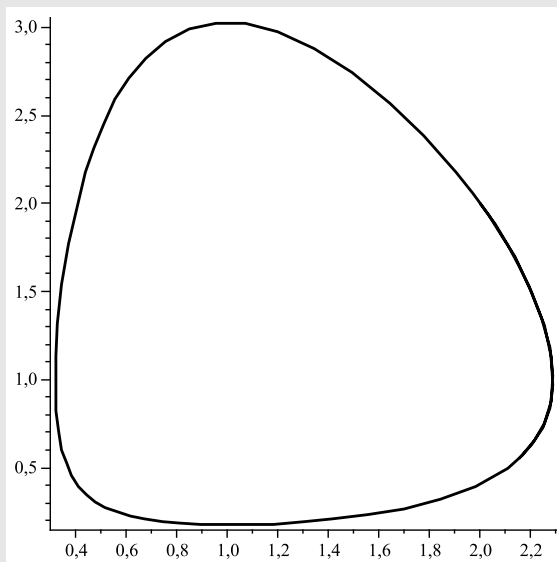
2.120658201

### Systèmes d'équations différentielles

La commande `dsolve` permet également la résolution approchée des systèmes d'équations différentielles. La syntaxe s'inspire largement de ce qui précède et de ce qui a été déjà vu lors de l'étude de la résolution formelle des systèmes d'équations.

Voici l'exemple du système *proie-prédateurs*<sup>5</sup>. On observe classiquement une *oscillation* du système : ce dernier est stable.

```
> restart:equ:={diff(x(t),t)=0.5*x(t)*(y(t)-1),diff(y(t),t)=y(t)*(1-x(t)),x(0)=2,y(0)=2}:
sol:=dsolve(equ,numeric,output=listprocedure):
x:=subs(sol,x(t)):y:=subs(sol,y(t)):
plot([x,y,0..11],color=black,thickness=2);
```



5. les fonctions  $x$  et  $y$  représentent respectivement le nombre de prédateurs et de proies en fonction du temps. On a modélisé leur coexistence en fonction du temps  $t$  par le système différentiel  $\dot{x}(t) = 0.5x(t)(y(t) - 1)$ ,  $\dot{y}(t) = y(t)(1 - x(t))$ .

## 4. Exercices

### Exercice 1.

Pour tout entier naturel  $n$ , on pose

$$\forall x \in \mathbb{R}, f_n(x) = \sin(x) \cos^n(x).$$

1. Représenter sur un même graphique les fonctions  $f_n$  pour  $n$  variant de 0 à 15 sur  $[0, 2\pi]$ .
2. Caculer en utilisant le logiciel le maximum  $u_n$  de  $f_n$  sur  $[0, 2\pi]$ .
3. Déterminer la limite de  $u_n$  quand  $n$  tend vers  $+\infty$  puis un équivalent de  $u_n$ .

### Exercice 2.

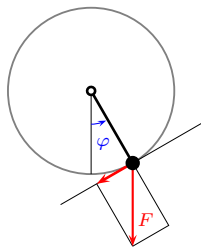
Trouver la limite puis un équivalent du terme général de la suite définie par

$$\forall n \geq 2, u_n = \cos(n^2 \pi \ln(1 - 1/n)).$$

### Exercice 3.

*Solution apériodique du pendule*

On considère l'équation  $\ddot{\varphi} = -U'(\varphi)$  du pendule mathématique avec le potentiel  $U(\varphi) = -\cos \varphi$ .



(Masse et longueur du pendule ainsi que l'accélération de pesanteur  $g$  sont prises égales à 1.)

1. D'après la loi de conservation d'énergie on a  $\frac{\dot{\varphi}^2}{2} - \cos \varphi = E = \text{cte}$ . Montrer que le niveau d'énergie  $E = 1$  correspond aux solutions apériodiques du pendule.
2. Calculer (à la main ou avec Maple) la solution correspondant à la condition initiale  $\varphi(0) = 0, \dot{\varphi}(0) = 2$ .
3. Grâce à Maple représenter des solutions pour divers niveaux d'énergie. Les interpréter et classer selon leur type de mouvement.

### Exercice 4.

Soit

$$(\mathbf{E}) : y'' + \frac{y}{x^2} = 0.$$

Résoudre  $(\mathbf{E})$  au moyen du logiciel sur  $[1, +\infty[$ . Dédire de la forme des solutions une méthode de résolution de  $(\mathbf{E})$ .

**Exercice 5.**

Soit

$$(\mathbf{E}) : x^2 y'' + 4xy' + 2y = \ln(1+x).$$

1. Résoudre  $(\mathbf{E})$  sur  $]0, +\infty[$  en utilisant Maple.
2. Déterminer l'unique solution de  $(\mathbf{E})$  bornée au voisinage 0. On la notera  $f$ .
3. Tracer le graphe de  $f$  sur  $[0, +\infty[$ .