

# Chap. II : Analyse avec Maple

Laurent Poinsot

25 septembre 2009

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Plan

## 1 Calculs sur les nombres entiers et réels

- Calculs sur les nombres réels
- Calculs sur les entiers
- Les nombres premiers
- Analyse combinatoire
- Sommation
- Produit

## 2 Les nombres complexes

## 3 Fonctions numériques

- Dérivation
- Limites
- Intégration
- Développements limités

## 4 Expressions symboliques

- Polynômes
- Fractions rationnelles

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Plan

- 1** Calculs sur les nombres entiers et réels
  - Calculs sur les nombres réels
  - Calculs sur les entiers
  - Les nombres premiers
  - Analyse combinatoire
  - Sommation
  - Produit
- 2** Les nombres complexes
- 3** Fonctions numériques
  - Dérivation
  - Limites
  - Intégration
  - Développements limités
- 4** Expressions symboliques
  - Polynômes
  - Fractions rationnelles

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Plan

- 1 Calculs sur les nombres entiers et réels
  - Calculs sur les nombres réels
  - Calculs sur les entiers
  - Les nombres premiers
  - Analyse combinatoire
  - Sommation
  - Produit
- 2 Les nombres complexes
- 3 Fonctions numériques
  - Dérivation
  - Limites
  - Intégration
  - Développements limités
- 4 Expressions symboliques
  - Polynômes
  - Fractions rationnelles

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Plan

## 1 Calculs sur les nombres entiers et réels

- Calculs sur les nombres réels
- Calculs sur les entiers
- Les nombres premiers
- Analyse combinatoire
- Sommation
- Produit

## 2 Les nombres complexes

## 3 Fonctions numériques

- Dérivation
- Limites
- Intégration
- Développements limités

## 4 Expressions symboliques

- Polynômes
- Fractions rationnelles

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

## 1 Calculs sur les nombres entiers et réels

### ■ Calculs sur les nombres réels

- Calculs sur les entiers
- Les nombres premiers
- Analyse combinatoire
- Sommation
- Produit

## 2 Les nombres complexes

## 3 Fonctions numériques

- Dérivation
- Limites
- Intégration
- Développements limités

## 4 Expressions symboliques

- Polynômes
- Fractions rationnelles

Sous Maple, la priorité des opérations est bien évidemment la même qu'en mathématiques. Il ne faut pas hésiter à mettre des parenthèses afin d'éviter toute ambiguïté.

Par ex., on définit une fraction rationnelle à l'aide de la ligne de commande suivante :

`> (2 * x^2 + 3 * x + 1)/(2 * x + 1);`

$$\frac{2x^2 + 3x + 1}{2x + 1}$$

Sous Maple, la priorité des opérations est bien évidemment la même qu'en mathématiques. Il ne faut pas hésiter à mettre des parenthèses afin d'éviter toute ambiguïté.

Par ex., on définit une fraction rationnelle à l'aide de la ligne de commande suivante :

`> (2 * x^2 + 3 * x + 1)/(2 * x + 1);`

$$\frac{2x^2 + 3x + 1}{2x + 1}$$

Sous Maple, la priorité des opérations est bien évidemment la même qu'en mathématiques. Il ne faut pas hésiter à mettre des parenthèses afin d'éviter toute ambiguïté.

Par ex., on définit une fraction rationnelle à l'aide de la ligne de commande suivante :

```
> (2 * x^2 + 3 * x + 1)/(2 * x + 1);
```

$$\frac{2x^2 + 3x + 1}{2x + 1}$$

Sous Maple, la priorité des opérations est bien évidemment la même qu'en mathématiques. Il ne faut pas hésiter à mettre des parenthèses afin d'éviter toute ambiguïté.

Par ex., on définit une fraction rationnelle à l'aide de la ligne de commande suivante :

`> (2 * x^2 + 3 * x + 1)/(2 * x + 1);`

$$\frac{2x^2 + 3x + 1}{2x + 1}$$

Sous Maple, la priorité des opérations est bien évidemment la même qu'en mathématiques. Il ne faut pas hésiter à mettre des parenthèses afin d'éviter toute ambiguïté.

Par ex., on définit une fraction rationnelle à l'aide de la ligne de commande suivante :

$> (2 * x^2 + 3 * x + 1)/(2 * x + 1);$

$$\frac{2x^2 + 3x + 1}{2x + 1}$$

Le calcul sur les nombres réels fait intervenir les opérateurs arithmétiques et algébriques classiques. Par exemple, pour effectuer une soustraction, une multiplication et une division, on écrira :

> 3-2 ;

1

> 4\*23 ;

92

>17/4 ;

$\frac{17}{4}$

Maple manipule les fractions :

> 1/2+3/8 ;

$\frac{7}{8}$

Le calcul sur les nombres réels fait intervenir les opérateurs arithmétiques et algébriques classiques. Par exemple, pour effectuer une soustraction, une multiplication et une division, on écrira :

> 3-2 ;

1

> 4\*23 ;

92

>17/4 ;

$\frac{17}{4}$

Maple manipule les fractions :

> 1/2+3/8 ;

$\frac{7}{8}$

Le calcul sur les nombres réels fait intervenir les opérateurs arithmétiques et algébriques classiques. Par exemple, pour effectuer une soustraction, une multiplication et une division, on écrira :

> 3-2 ;

1

> 4\*23 ;

92

>17/4 ;

$\frac{17}{4}$

Maple manipule les fractions :

> 1/2+3/8 ;

$\frac{7}{8}$

Le calcul sur les nombres réels fait intervenir les opérateurs arithmétiques et algébriques classiques. Par exemple, pour effectuer une soustraction, une multiplication et une division, on écrira :

> 3-2 ;

1

> 4\*23 ;

92

>17/4 ;

$\frac{17}{4}$

Maple manipule les fractions :

> 1/2+3/8 ;

$\frac{7}{8}$

Le calcul sur les nombres réels fait intervenir les opérateurs arithmétiques et algébriques classiques. Par exemple, pour effectuer une soustraction, une multiplication et une division, on écrira :

> 3-2 ;

1

> 4\*23 ;

92

>17/4 ;

$\frac{17}{4}$

Maple manipule les fractions :

> 1/2+3/8 ;

$\frac{7}{8}$

Le calcul sur les nombres réels fait intervenir les opérateurs arithmétiques et algébriques classiques. Par exemple, pour effectuer une soustraction, une multiplication et une division, on écrira :

> 3-2 ;

1

> 4\*23 ;

92

>17/4 ;

$\frac{17}{4}$

Maple manipule les fractions :

> 1/2+3/8 ;

$\frac{7}{8}$

Le calcul sur les nombres réels fait intervenir les opérateurs arithmétiques et algébriques classiques. Par exemple, pour effectuer une soustraction, une multiplication et une division, on écrira :

$$> 3-2 ;$$

1

$$> 4*23 ;$$

92

$$> 17/4 ;$$

$$\frac{17}{4}$$

Maple manipule les fractions :

$$> 1/2+3/8 ;$$

$$\frac{7}{8}$$

Le calcul sur les nombres réels fait intervenir les opérateurs arithmétiques et algébriques classiques. Par exemple, pour effectuer une soustraction, une multiplication et une division, on écrira :

> 3-2 ;

1

> 4\*23 ;

92

>17/4 ;

$\frac{17}{4}$

Maple manipule les fractions :

> 1/2+3/8 ;

$\frac{7}{8}$

Le calcul sur les nombres réels fait intervenir les opérateurs arithmétiques et algébriques classiques. Par exemple, pour effectuer une soustraction, une multiplication et une division, on écrira :

> 3-2 ;

1

> 4\*23 ;

92

>17/4 ;

$\frac{17}{4}$

Maple manipule les fractions :

> 1/2+3/8 ;

$\frac{7}{8}$

Le calcul sur les nombres réels fait intervenir les opérateurs arithmétiques et algébriques classiques. Par exemple, pour effectuer une soustraction, une multiplication et une division, on écrira :

> 3-2 ;

1

> 4\*23 ;

92

>17/4 ;

$\frac{17}{4}$

Maple manipule les fractions :

> 1/2+3/8 ;

$\frac{7}{8}$

Le calcul sur les nombres réels fait intervenir les opérateurs arithmétiques et algébriques classiques. Par exemple, pour effectuer une soustraction, une multiplication et une division, on écrira :

> 3-2 ;

1

> 4\*23 ;

92

>17/4 ;

$\frac{17}{4}$

Maple manipule les fractions :

> 1/2+3/8 ;

$\frac{7}{8}$

Maple connaît aussi les fonctions classiques telles que la racine carrée :

```
> sqrt(2);
```

$$\sqrt{2}$$

Pour obtenir une valeur approchée, on peut utiliser la commande `evalf` :

```
> evalf(%);
```

$$1.414213562$$

On peut aussi obliger Maple à reconnaître l'entier 2 comme un réel en ajoutant un point après le 2 :

```
> sqrt(2.);
```

$$1.414213562$$

Maple connaît aussi les fonctions classiques telles que la racine carrée :

```
> sqrt(2);
```

$$\sqrt{2}$$

Pour obtenir une valeur approchée, on peut utiliser la commande `evalf` :

```
> evalf(%);
```

$$1.414213562$$

On peut aussi obliger Maple à reconnaître l'entier 2 comme un réel en ajoutant un point après le 2 :

```
> sqrt(2.);
```

$$1.414213562$$

Maple connaît aussi les fonctions classiques telles que la racine carrée :

```
> sqrt(2);
```

$$\sqrt{2}$$

Pour obtenir une valeur approchée, on peut utiliser la commande `evalf` :

```
> evalf(%);
```

1.414213562

On peut aussi obliger Maple à reconnaître l'entier 2 comme un réel en ajoutant un point après le 2 :

```
> sqrt(2.);
```

1.414213562

Maple connaît aussi les fonctions classiques telles que la racine carrée :

```
> sqrt(2);
```

$$\sqrt{2}$$

Pour obtenir une valeur approchée, on peut utiliser la commande `evalf` :

```
> evalf(%);
```

1.414213562

On peut aussi obliger Maple à reconnaître l'entier 2 comme un réel en ajoutant un point après le 2 :

```
> sqrt(2.);
```

1.414213562

Maple connaît aussi les fonctions classiques telles que la racine carrée :

```
> sqrt(2);
```

$$\sqrt{2}$$

Pour obtenir une valeur approchée, on peut utiliser la commande `evalf` :

```
> evalf(%);
```

1.414213562

On peut aussi obliger Maple à reconnaître l'entier 2 comme un réel en ajoutant un point après le 2 :

```
> sqrt(2.);
```

1.414213562

Maple connaît aussi les fonctions classiques telles que la racine carrée :

```
> sqrt(2);
```

$$\sqrt{2}$$

Pour obtenir une valeur approchée, on peut utiliser la commande `evalf` :

```
> evalf(%);
```

1.414213562

On peut aussi obliger Maple à reconnaître l'entier 2 comme un réel en ajoutant un point après le 2 :

```
> sqrt(2.);
```

1.414213562

Maple connaît aussi les fonctions classiques telles que la racine carrée :

```
> sqrt(2);
```

$$\sqrt{2}$$

Pour obtenir une valeur approchée, on peut utiliser la commande `evalf` :

```
> evalf(%);
```

1.414213562

On peut aussi obliger Maple à reconnaître l'entier 2 comme un réel en ajoutant un point après le 2 :

```
> sqrt(2.);
```

1.414213562

Maple connaît aussi les fonctions classiques telles que la racine carrée :

```
> sqrt(2) ;
```

$$\sqrt{2}$$

Pour obtenir une valeur approchée, on peut utiliser la commande `evalf` :

```
> evalf(%) ;
```

1.414213562

On peut aussi obliger Maple à reconnaître l'entier 2 comme un réel en ajoutant un point après le 2 :

```
> sqrt(2.) ;
```

1.414213562

Maple connaît aussi les fonctions classiques telles que la racine carrée :

```
> sqrt(2);
```

$$\sqrt{2}$$

Pour obtenir une valeur approchée, on peut utiliser la commande `evalf` :

```
> evalf(%);
```

1.414213562

On peut aussi obliger Maple à reconnaître l'entier 2 comme un réel en ajoutant un point après le 2 :

```
> sqrt(2.);
```

1.414213562

La commande `evalf` permet également de passer d'une fraction à un nombre décimal :

```
> evalf(5/2);
```

2.5

On peut aussi effectuer l'opération inverse :

```
> convert(2.5, fraction);
```

$\frac{5}{2}$

La commande `evalf` permet également de passer d'une fraction à un nombre décimal :

```
> evalf(5/2) ;
```

2.5

On peut aussi effectuer l'opération inverse :

```
> convert(2.5, fraction) ;
```

$\frac{5}{2}$

La commande `evalf` permet également de passer d'une fraction à un nombre décimal :

```
> evalf(5/2);
```

2.5

On peut aussi effectuer l'opération inverse :

```
> convert(2.5, fraction);
```

$\frac{5}{2}$

La commande `evalf` permet également de passer d'une fraction à un nombre décimal :

```
> evalf(5/2);
```

2.5

On peut aussi effectuer l'opération inverse :

```
> convert(2.5, fraction);
```

$\frac{5}{2}$

La commande `evalf` permet également de passer d'une fraction à un nombre décimal :

```
> evalf(5/2);
```

2.5

On peut aussi effectuer l'opération inverse :

```
> convert(2.5, fraction);
```

$\frac{5}{2}$

La commande `evalf` permet également de passer d'une fraction à un nombre décimal :

```
> evalf(5/2) ;
```

2.5

On peut aussi effectuer l'opération inverse :

```
> convert(2.5, fraction) ;
```

$\frac{5}{2}$

# Valeur absolue

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

On obtient la valeur absolue à l'aide de la commande `abs` :

```
> abs ( expression ) ;
```

Cette même fonction servira à obtenir le module d'un nombre complexe.

```
> abs (-5) ;
```

5

```
> abs (6) ;
```

6

# Valeur absolue

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

On obtient la valeur absolue à l'aide de la commande `abs` :

```
> abs (expression) ;
```

Cette même fonction servira à obtenir le module d'un nombre complexe.

```
> abs (-5) ;
```

5

```
> abs (6) ;
```

6

# Valeur absolue

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

On obtient la valeur absolue à l'aide de la commande `abs` :

```
> abs (expression) ;
```

Cette même fonction servira à obtenir le module d'un nombre complexe.

```
> abs (-5) ;
```

5

```
> abs (6) ;
```

6

# Valeur absolue

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

On obtient la valeur absolue à l'aide de la commande `abs` :

```
> abs (expression) ;
```

Cette même fonction servira à obtenir le module d'un nombre complexe.

```
> abs (-5) ;
```

5

```
> abs (6) ;
```

6

# Valeur absolue

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

On obtient la valeur absolue à l'aide de la commande `abs` :

```
> abs (expression) ;
```

Cette même fonction servira à obtenir le module d'un nombre complexe.

```
> abs (-5) ;
```

5

```
> abs (6) ;
```

6

# Valeur absolue

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

On obtient la valeur absolue à l'aide de la commande `abs` :

```
> abs (expression) ;
```

Cette même fonction servira à obtenir le module d'un nombre complexe.

```
> abs (-5) ;
```

5

```
> abs (6) ;
```

6

# Valeur absolue

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

On obtient la valeur absolue à l'aide de la commande `abs` :

```
> abs (expression) ;
```

Cette même fonction servira à obtenir le module d'un nombre complexe.

```
> abs (-5) ;
```

5

```
> abs (6) ;
```

6

# Minimum et maximum

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

On peut aussi chercher le minimum ou le maximum d'une suite de nombres à l'aide des commandes `min` et `max` :

```
> min(1, 2, 6, -3) ;
```

-3

```
> max(1, 2, 6, -3) ;
```

6

# Minimum et maximum

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

On peut aussi chercher le minimum ou le maximum d'une suite de nombres à l'aide des commandes `min` et `max` :

```
> min(1, 2, 6, -3) ;
```

-3

```
> max(1, 2, 6, -3) ;
```

6

# Minimum et maximum

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

On peut aussi chercher le minimum ou le maximum d'une suite de nombres à l'aide des commandes `min` et `max` :

```
> min(1, 2, 6, -3) ;
```

-3

```
> max(1, 2, 6, -3) ;
```

6

# Minimum et maximum

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

On peut aussi chercher le minimum ou le maximum d'une suite de nombres à l'aide des commandes `min` et `max` :

```
> min(1, 2, 6, -3) ;
```

-3

```
> max(1, 2, 6, -3) ;
```

6

# Minimum et maximum

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

On peut aussi chercher le minimum ou le maximum d'une suite de nombres à l'aide des commandes `min` et `max` :

```
> min(1, 2, 6, -3) ;
```

-3

```
> max(1, 2, 6, -3) ;
```

6

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

## 1 Calculs sur les nombres entiers et réels

- Calculs sur les nombres réels

### ■ Calculs sur les entiers

- Les nombres premiers
- Analyse combinatoire
- Sommation
- Produit

## 2 Les nombres complexes

## 3 Fonctions numériques

- Dérivation
- Limites
- Intégration
- Développements limités

## 4 Expressions symboliques

- Polynômes
- Fractions rationnelles

# Division euclidienne

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Maple permet de réaliser des opérations usuelles sur les nombres entiers :

La commande `iquo` permet d'obtenir le quotient de la division euclidienne de  $a$  par  $b$  :

```
> iquo(a,b);
```

La commande `irem` permet d'obtenir le reste de cette même division euclidienne :

```
> irem(a,b);
```

Par exemple,

```
> iquo(14,5);
```

2

```
> irem(14,5);
```

4

# Division euclidienne

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Maple permet de réaliser des opérations usuelles sur les nombres entiers :

La commande `iquo` permet d'obtenir le quotient de la division euclidienne de  $a$  par  $b$  :

```
> iquo(a,b);
```

La commande `irem` permet d'obtenir le reste de cette même division euclidienne :

```
> irem(a,b);
```

Par exemple,

```
> iquo(14,5);
```

2

```
> irem(14,5);
```

4

# Division euclidienne

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Maple permet de réaliser des opérations usuelles sur les nombres entiers :

La commande `iquo` permet d'obtenir le quotient de la division euclidienne de  $a$  par  $b$  :

```
> iquo(a,b) ;
```

La commande `irem` permet d'obtenir le reste de cette même division euclidienne :

```
> irem(a,b) ;
```

Par exemple,

```
> iquo(14,5) ;
```

2

```
> irem(14,5) ;
```

4

# Division euclidienne

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Maple permet de réaliser des opérations usuelles sur les nombres entiers :

La commande `iquo` permet d'obtenir le quotient de la division euclidienne de  $a$  par  $b$  :

```
> iquo(a,b) ;
```

La commande `irem` permet d'obtenir le reste de cette même division euclidienne :

```
> irem(a,b) ;
```

Par exemple,

```
> iquo(14,5) ;
```

2

```
> irem(14,5) ;
```

4

# Division euclidienne

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Maple permet de réaliser des opérations usuelles sur les nombres entiers :

La commande `iquo` permet d'obtenir le quotient de la division euclidienne de  $a$  par  $b$  :

```
> iquo(a,b) ;
```

La commande `irem` permet d'obtenir le reste de cette même division euclidienne :

```
> irem(a,b) ;
```

Par exemple,

```
> iquo(14,5) ;
```

2

```
> irem(14,5) ;
```

4

# Division euclidienne

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Maple permet de réaliser des opérations usuelles sur les nombres entiers :

La commande `iquo` permet d'obtenir le quotient de la division euclidienne de  $a$  par  $b$  :

```
> iquo(a,b) ;
```

La commande `irem` permet d'obtenir le reste de cette même division euclidienne :

```
> irem(a,b) ;
```

Par exemple,

```
> iquo(14,5) ;
```

2

```
> irem(14,5) ;
```

4

# Division euclidienne

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Maple permet de réaliser des opérations usuelles sur les nombres entiers :

La commande `iquo` permet d'obtenir le quotient de la division euclidienne de  $a$  par  $b$  :

```
> iquo(a,b) ;
```

La commande `irem` permet d'obtenir le reste de cette même division euclidienne :

```
> irem(a,b) ;
```

Par exemple,

```
> iquo(14,5) ;
```

2

```
> irem(14,5) ;
```

4

# Division euclidienne

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Maple permet de réaliser des opérations usuelles sur les nombres entiers :

La commande `iquo` permet d'obtenir le quotient de la division euclidienne de  $a$  par  $b$  :

```
> iquo(a,b) ;
```

La commande `irem` permet d'obtenir le reste de cette même division euclidienne :

```
> irem(a,b) ;
```

Par exemple,

```
> iquo(14,5) ;
```

2

```
> irem(14,5) ;
```

4

# Division euclidienne

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Maple permet de réaliser des opérations usuelles sur les nombres entiers :

La commande `iquo` permet d'obtenir le quotient de la division euclidienne de  $a$  par  $b$  :

```
> iquo(a,b) ;
```

La commande `irem` permet d'obtenir le reste de cette même division euclidienne :

```
> irem(a,b) ;
```

Par exemple,

```
> iquo(14,5) ;
```

2

```
> irem(14,5) ;
```

4

# Division euclidienne

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Maple permet de réaliser des opérations usuelles sur les nombres entiers :

La commande `iquo` permet d'obtenir le quotient de la division euclidienne de  $a$  par  $b$  :

```
> iquo(a,b) ;
```

La commande `irem` permet d'obtenir le reste de cette même division euclidienne :

```
> irem(a,b) ;
```

Par exemple,

```
> iquo(14,5) ;
```

2

```
> irem(14,5) ;
```

4

Maple permet aussi de décomposer un nombre en un produit de ses facteurs premiers :

```
> ifactor(35748) ;
```

$$(2)^2(3)^3(331)$$

Maple permet aussi de décomposer un nombre en un produit de ses facteurs premiers :

```
> ifactor(35748) ;
```

$$(2)^2(3)^3(331)$$

Maple permet aussi de décomposer un nombre en un produit de ses facteurs premiers :

```
> ifactor(35748) ;
```

$$(2)^2(3)^3(331)$$

La commande `igcd` calcule le plus grand commun diviseur (pgcd) des entiers  $n_1, \dots, n_k$  :

```
> igcd( $n_1, \dots, n_k$ ) ;
```

La commande `ilcm` calcule le plus petit commun multiple (ppcm) des entiers  $n_1, \dots, n_k$  :

```
> ilcm( $n_1, \dots, n_k$ ) ;
```

Par exemple :

```
> igcd(15, 12) ;
```

3

```
> ilcm(112, 15) ;
```

1680

La commande `igcd` calcule le plus grand commun diviseur (pgcd) des entiers  $n_1, \dots, n_k$  :

```
> igcd( $n_1, \dots, n_k$ ) ;
```

La commande `ilcm` calcule le plus petit commun multiple (ppcm) des entiers  $n_1, \dots, n_k$  :

```
> ilcm( $n_1, \dots, n_k$ ) ;
```

Par exemple :

```
> igcd(15, 12) ;
```

3

```
> ilcm(112, 15) ;
```

1680

La commande `igcd` calcule le plus grand commun diviseur (pgcd) des entiers  $n_1, \dots, n_k$  :

```
> igcd( $n_1, \dots, n_k$ ) ;
```

La commande `ilcm` calcule le plus petit commun multiple (ppcm) des entiers  $n_1, \dots, n_k$  :

```
> ilcm( $n_1, \dots, n_k$ ) ;
```

Par exemple :

```
> igcd(15, 12) ;
```

3

```
> ilcm(112, 15) ;
```

1680

La commande `igcd` calcule le plus grand commun diviseur (pgcd) des entiers  $n_1, \dots, n_k$  :

```
> igcd( $n_1, \dots, n_k$ ) ;
```

La commande `ilcm` calcule le plus petit commun multiple (ppcm) des entiers  $n_1, \dots, n_k$  :

```
> ilcm( $n_1, \dots, n_k$ ) ;
```

Par exemple :

```
> igcd(15, 12) ;
```

3

```
> ilcm(112, 15) ;
```

1680

La commande `igcd` calcule le plus grand commun diviseur (pgcd) des entiers  $n_1, \dots, n_k$  :

```
> igcd( $n_1, \dots, n_k$ ) ;
```

La commande `ilcm` calcule le plus petit commun multiple (ppcm) des entiers  $n_1, \dots, n_k$  :

```
> ilcm( $n_1, \dots, n_k$ ) ;
```

Par exemple :

```
> igcd(15, 12) ;
```

3

```
> ilcm(112, 15) ;
```

1680

La commande `igcd` calcule le plus grand commun diviseur (pgcd) des entiers  $n_1, \dots, n_k$  :

```
> igcd( $n_1, \dots, n_k$ ) ;
```

La commande `ilcm` calcule le plus petit commun multiple (ppcm) des entiers  $n_1, \dots, n_k$  :

```
> ilcm( $n_1, \dots, n_k$ ) ;
```

Par exemple :

```
> igcd(15, 12) ;
```

3

```
> ilcm(112, 15) ;
```

1680

La commande `igcd` calcule le plus grand commun diviseur (pgcd) des entiers  $n_1, \dots, n_k$  :

```
> igcd( $n_1, \dots, n_k$ ) ;
```

La commande `ilcm` calcule le plus petit commun multiple (ppcm) des entiers  $n_1, \dots, n_k$  :

```
> ilcm( $n_1, \dots, n_k$ ) ;
```

Par exemple :

```
> igcd(15, 12) ;
```

3

```
> ilcm(112, 15) ;
```

1680

La commande `igcd` calcule le plus grand commun diviseur (pgcd) des entiers  $n_1, \dots, n_k$  :

```
> igcd( $n_1, \dots, n_k$ ) ;
```

La commande `ilcm` calcule le plus petit commun multiple (ppcm) des entiers  $n_1, \dots, n_k$  :

```
> ilcm( $n_1, \dots, n_k$ ) ;
```

Par exemple :

```
> igcd(15, 12) ;
```

3

```
> ilcm (112, 15) ;
```

1680

La commande `igcd` calcule le plus grand commun diviseur (pgcd) des entiers  $n_1, \dots, n_k$  :

```
> igcd( $n_1, \dots, n_k$ ) ;
```

La commande `ilcm` calcule le plus petit commun multiple (ppcm) des entiers  $n_1, \dots, n_k$  :

```
> ilcm( $n_1, \dots, n_k$ ) ;
```

Par exemple :

```
> igcd(15, 12) ;
```

3

```
> ilcm (112, 15) ;
```

1680

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

## 1 Calculs sur les nombres entiers et réels

- Calculs sur les nombres réels
- Calculs sur les entiers
- **Les nombres premiers**
- Analyse combinatoire
- Sommation
- Produit

## 2 Les nombres complexes

## 3 Fonctions numériques

- Dérivation
- Limites
- Intégration
- Développements limités

## 4 Expressions symboliques

- Polynômes
- Fractions rationnelles

On connaît l'importance des nombres premiers en arithmétique. Maple permet de manipuler facilement les nombres premiers :

La fonction `isprime` permet de savoir si un nombre est ou n'est pas premier :

```
> isprime(nombre) ;
```

On obtient le  $n$ -ème nombre premier à l'aide de la commande `ithprime` :

```
> ithprime(n) ;
```

Par exemple, pour savoir si 18 ou 17 sont premiers :

```
> isprime(18) ;
```

*false*

```
> isprime(17) ;
```

*true*

On connaît l'importance des nombres premiers en arithmétique. Maple permet de manipuler facilement les nombres premiers :

La fonction `isprime` permet de savoir si un nombre est ou n'est pas premier :

```
> isprime(nombre) ;
```

On obtient le n-ème nombre premier à l'aide de la commande `ithprime` :

```
> ithprime(n) ;
```

Par exemple, pour savoir si 18 ou 17 sont premiers :

```
> isprime(18) ;
```

*false*

```
> isprime(17) ;
```

*true*

On connaît l'importance des nombres premiers en arithmétique. Maple permet de manipuler facilement les nombres premiers :

La fonction `isprime` permet de savoir si un nombre est ou n'est pas premier :

```
> isprime(nombre) ;
```

On obtient le n-ème nombre premier à l'aide de la commande `ithprime` :

```
> ithprime(n) ;
```

Par exemple, pour savoir si 18 ou 17 sont premiers :

```
> isprime(18) ;
```

*false*

```
> isprime(17) ;
```

*true*

On connaît l'importance des nombres premiers en arithmétique. Maple permet de manipuler facilement les nombres premiers :

La fonction `isprime` permet de savoir si un nombre est ou n'est pas premier :

```
> isprime(nombre) ;
```

On obtient le  $n$ -ème nombre premier à l'aide de la commande `ithprime` :

```
> ithprime(n) ;
```

Par exemple, pour savoir si 18 ou 17 sont premiers :

```
> isprime(18) ;
```

*false*

```
> isprime(17) ;
```

*true*

On connaît l'importance des nombres premiers en arithmétique. Maple permet de manipuler facilement les nombres premiers :

La fonction `isprime` permet de savoir si un nombre est ou n'est pas premier :

```
> isprime(nombre) ;
```

On obtient le n-ème nombre premier à l'aide de la commande `ithprime` :

```
> ithprime(n) ;
```

Par exemple, pour savoir si 18 ou 17 sont premiers :

```
> isprime(18) ;
```

*false*

```
> isprime(17) ;
```

*true*

On connaît l'importance des nombres premiers en arithmétique. Maple permet de manipuler facilement les nombres premiers :

La fonction `isprime` permet de savoir si un nombre est ou n'est pas premier :

```
> isprime(nombre) ;
```

On obtient le n-ème nombre premier à l'aide de la commande `ithprime` :

```
> ithprime(n) ;
```

Par exemple, pour savoir si 18 ou 17 sont premiers :

```
> isprime(18) ;
```

*false*

```
> isprime(17) ;
```

*true*

On connaît l'importance des nombres premiers en arithmétique. Maple permet de manipuler facilement les nombres premiers :

La fonction `isprime` permet de savoir si un nombre est ou n'est pas premier :

```
> isprime(nombre) ;
```

On obtient le n-ème nombre premier à l'aide de la commande `ithprime` :

```
> ithprime(n) ;
```

Par exemple, pour savoir si 18 ou 17 sont premiers :

```
> isprime(18) ;
```

*false*

```
> isprime(17) ;
```

*true*

On connaît l'importance des nombres premiers en arithmétique. Maple permet de manipuler facilement les nombres premiers :

La fonction `isprime` permet de savoir si un nombre est ou n'est pas premier :

```
> isprime(nombre) ;
```

On obtient le n-ème nombre premier à l'aide de la commande `ithprime` :

```
> ithprime(n) ;
```

Par exemple, pour savoir si 18 ou 17 sont premiers :

```
> isprime(18) ;
```

*false*

```
> isprime(17) ;
```

*true*

On connaît l'importance des nombres premiers en arithmétique. Maple permet de manipuler facilement les nombres premiers :

La fonction `isprime` permet de savoir si un nombre est ou n'est pas premier :

```
> isprime(nombre) ;
```

On obtient le n-ème nombre premier à l'aide de la commande `ithprime` :

```
> ithprime(n) ;
```

Par exemple, pour savoir si 18 ou 17 sont premiers :

```
> isprime(18) ;
```

*false*

```
> isprime(17) ;
```

*true*

On connaît l'importance des nombres premiers en arithmétique. Maple permet de manipuler facilement les nombres premiers :

La fonction `isprime` permet de savoir si un nombre est ou n'est pas premier :

```
> isprime(nombre) ;
```

On obtient le n-ème nombre premier à l'aide de la commande `ithprime` :

```
> ithprime(n) ;
```

Par exemple, pour savoir si 18 ou 17 sont premiers :

```
> isprime(18) ;
```

*false*

```
> isprime(17) ;
```

*true*

On peut également obtenir le 25-ème nombre premier :

```
> ithprime(25);  
97
```

Enfin, le système permet d'aller d'un nombre premier au suivant à l'aide de la fonction `nextprime` :

```
> nextprime(7);  
11
```

```
> nextprime(17);  
19
```

On peut également obtenir le 25-ème nombre premier :

```
> ithprime(25);  
97
```

Enfin, le système permet d'aller d'un nombre premier au suivant à l'aide de la fonction `nextprime` :

```
> nextprime(7);  
11  
  
> nextprime(17);  
19
```

On peut également obtenir le 25-ème nombre premier :

```
> ithprime(25);  
97
```

Enfin, le système permet d'aller d'un nombre premier au suivant à l'aide de la fonction `nextprime` :

```
> nextprime(7);  
11  
  
> nextprime(17);  
19
```

On peut également obtenir le 25-ème nombre premier :

```
> ithprime(25);  
97
```

Enfin, le système permet d'aller d'un nombre premier au suivant à l'aide de la fonction `nextprime` :

```
> nextprime(7);  
11  
  
> nextprime(17);  
19
```

On peut également obtenir le 25-ème nombre premier :

```
> ithprime(25);  
97
```

Enfin, le système permet d'aller d'un nombre premier au suivant à l'aide de la fonction `nextprime` :

```
> nextprime(7);  
11
```

```
> nextprime(17);  
19
```

On peut également obtenir le 25-ème nombre premier :

```
> ithprime(25);  
97
```

Enfin, le système permet d'aller d'un nombre premier au suivant à l'aide de la fonction `nextprime` :

```
> nextprime(7);  
11
```

```
> nextprime(17);  
19
```

On peut également obtenir le 25-ème nombre premier :

```
> ithprime(25);  
97
```

Enfin, le système permet d'aller d'un nombre premier au suivant à l'aide de la fonction `nextprime` :

```
> nextprime(7);  
11
```

```
> nextprime(17);  
19
```

On peut également obtenir le 25-ème nombre premier :

```
> ithprime(25);  
97
```

Enfin, le système permet d'aller d'un nombre premier au suivant à l'aide de la fonction `nextprime` :

```
> nextprime(7);  
11
```

```
> nextprime(17);  
19
```

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

## 1 Calculs sur les nombres entiers et réels

- Calculs sur les nombres réels
- Calculs sur les entiers
- Les nombres premiers
- **Analyse combinatoire**
- Sommation
- Produit

## 2 Les nombres complexes

## 3 Fonctions numériques

- Dérivation
- Limites
- Intégration
- Développements limités

## 4 Expressions symboliques

- Polynômes
- Fractions rationnelles

Une autre fonction fréquemment employée est la **factorielle**.

```
> 10 ! ;
```

3628800

On génère un entier de manière aléatoire avec la fonction `rand` :

```
> rand() ;
```

Pour générer aléatoirement un entier entre *min* et *max* on utilise d'abord :

```
> nom := rand(min..max) ;
```

On obtient ensuite ce nombre par :

```
> nom() ;
```

Une autre fonction fréquemment employée est la **factorielle**.

```
> 10 ! ;
```

```
3628800
```

On génère un entier de manière aléatoire avec la fonction `rand` :

```
> rand() ;
```

Pour générer aléatoirement un entier entre *min* et *max* on utilise d'abord :

```
> nom := rand(min..max) ;
```

On obtient ensuite ce nombre par :

```
> nom() ;
```

Une autre fonction fréquemment employée est la **factorielle**.

```
> 10 ! ;
```

3628800

On génère un entier de manière aléatoire avec la fonction `rand` :

```
> rand() ;
```

Pour générer aléatoirement un entier entre *min* et *max* on utilise d'abord :

```
> nom := rand(min..max) ;
```

On obtient ensuite ce nombre par :

```
> nom() ;
```

Une autre fonction fréquemment employée est la **factorielle**.

```
> 10 ! ;
```

3628800

On génère un entier de manière aléatoire avec la fonction `rand` :

```
> rand() ;
```

Pour générer aléatoirement un entier entre *min* et *max* on utilise d'abord :

```
> nom := rand(min..max) ;
```

On obtient ensuite ce nombre par :

```
> nom() ;
```

Une autre fonction fréquemment employée est la **factorielle**.

```
> 10 ! ;
```

3628800

On génère un entier de manière aléatoire avec la fonction `rand` :

```
> rand() ;
```

Pour générer aléatoirement un entier entre *min* et *max* on utilise d'abord :

```
> nom := rand(min..max) ;
```

On obtient ensuite ce nombre par :

```
> nom() ;
```

Une autre fonction fréquemment employée est la **factorielle**.

```
> 10 ! ;
```

3628800

On génère un entier de manière aléatoire avec la fonction `rand` :

```
> rand() ;
```

Pour générer aléatoirement un entier entre *min* et *max* on utilise d'abord :

```
> nom := rand(min..max) ;
```

On obtient ensuite ce nombre par :

```
> nom() ;
```

Une autre fonction fréquemment employée est la **factorielle**.

```
> 10 ! ;
```

3628800

On génère un entier de manière aléatoire avec la fonction `rand` :

```
> rand() ;
```

Pour générer aléatoirement un entier entre *min* et *max* on utilise d'abord :

```
> nom := rand(min..max) ;
```

On obtient ensuite ce nombre par :

```
> nom() ;
```

Une autre fonction fréquemment employée est la **factorielle**.

```
> 10 ! ;
```

3628800

On génère un entier de manière aléatoire avec la fonction `rand` :

```
> rand() ;
```

Pour générer aléatoirement un entier entre *min* et *max* on utilise d'abord :

```
> nom := rand(min..max) ;
```

On obtient ensuite ce nombre par :

```
> nom() ;
```

Une autre fonction fréquemment employée est la **factorielle**.

```
> 10 ! ;
```

3628800

On génère un entier de manière aléatoire avec la fonction `rand` :

```
> rand() ;
```

Pour générer aléatoirement un entier entre *min* et *max* on utilise d'abord :

```
> nom := rand(min..max) ;
```

On obtient ensuite ce nombre par :

```
> nom() ;
```

Les coefficients binomiaux sont donnés par la fonction  
binomial :

```
> binomial(n,k) ;
```

Par exemple :

```
> binomial(6,3) ;
```

20

Le package **combinat** apporte bon nombre de fonctions  
supplémentaires pour l'analyse combinatoire que nous ne  
détaillerons pas ici.

```
> with(combinat) ;
```

Les coefficients binomiaux sont donnés par la fonction  
binomial :

```
> binomial(n, k) ;
```

Par exemple :

```
> binomial(6, 3) ;
```

20

Le package **combinat** apporte bon nombre de fonctions  
supplémentaires pour l'analyse combinatoire que nous ne  
détaillerons pas ici.

```
> with(combinat) ;
```

Les coefficients binomiaux sont donnés par la fonction  
binomial :

```
> binomial(n, k) ;
```

Par exemple :

```
> binomial(6, 3) ;
```

20

Le package **combinat** apporte bon nombre de fonctions supplémentaires pour l'analyse combinatoire que nous ne détaillerons pas ici.

```
> with(combinat) ;
```

Les coefficients binomiaux sont donnés par la fonction  
binomial :

```
> binomial(n, k) ;
```

Par exemple :

```
> binomial(6, 3) ;
```

20

Le package **combinat** apporte bon nombre de fonctions supplémentaires pour l'analyse combinatoire que nous ne détaillerons pas ici.

```
> with(combinat) ;
```

Les coefficients binomiaux sont donnés par la fonction  
binomial :

```
> binomial(n, k) ;
```

Par exemple :

```
> binomial(6, 3) ;
```

20

Le package **combinat** apporte bon nombre de fonctions supplémentaires pour l'analyse combinatoire que nous ne détaillerons pas ici.

```
> with(combinat) ;
```

Les coefficients binomiaux sont donnés par la fonction  
binomial :

```
> binomial(n, k) ;
```

Par exemple :

```
> binomial(6, 3) ;
```

20

Le package **combinat** apporte bon nombre de fonctions  
supplémentaires pour l'analyse combinatoire que nous ne  
détaillerons pas ici.

```
> with(combinat) ;
```

Les coefficients binomiaux sont donnés par la fonction  
binomial :

```
> binomial(n, k) ;
```

Par exemple :

```
> binomial(6, 3) ;
```

20

Le package **combinat** apporte bon nombre de fonctions  
supplémentaires pour l'analyse combinatoire que nous ne  
détaillerons pas ici.

```
> with(combinat) ;
```

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

## 1 Calculs sur les nombres entiers et réels

- Calculs sur les nombres réels
- Calculs sur les entiers
- Les nombres premiers
- Analyse combinatoire
- **Sommation**
- Produit

## 2 Les nombres complexes

## 3 Fonctions numériques

- Dérivation
- Limites
- Intégration
- Développements limités

## 4 Expressions symboliques

- Polynômes
- Fractions rationnelles

Maple permet de calculer des sommes à l'aide de la fonction `sum`. Ainsi pour somme une expression  $u(i)$  pour  $i$  variant de  $a$  à  $b$  on écrit :

```
> sum(u(i), i : a..b) ;
```

Par exemple,

```
> sum(i ^ 2, i = 1..100) ; # Calcul de la somme
```

$$\sum_{i=1}^{100} i^2$$

338350

Maple permet aussi de calculer des sommes indéfinies, c'est-à-dire des sommes dont la borne supérieure n'est pas fixée.

```
> sum(i ^ 2, i = 1..n) ;
```

$$\frac{1}{3}(n+1)^3 - \frac{1}{2}(n+1)^2 + \frac{1}{6}n + \frac{1}{6}$$

Maple permet de calculer des sommes à l'aide de la fonction `sum`. Ainsi pour sommer une expression  $u(i)$  pour  $i$  variant de  $a$  à  $b$  on écrit :

```
> sum(u(i), i : a..b) ;
```

Par exemple,

```
> sum(i ^ 2, i = 1..100) ; # Calcul de la somme
```

$$\sum_{i=1}^{100} i^2$$

338350

Maple permet aussi de calculer des sommes indéfinies, c'est-à-dire des sommes dont la borne supérieure n'est pas fixée.

```
> sum(i ^ 2, i = 1..n) ;
```

$$\frac{1}{3}(n+1)^3 - \frac{1}{2}(n+1)^2 + \frac{1}{6}n + \frac{1}{6}$$

Maple permet de calculer des sommes à l'aide de la fonction `sum`. Ainsi pour somme une expression  $u(i)$  pour  $i$  variant de  $a$  à  $b$  on écrit :

```
> sum(u(i), i : a..b) ;
```

Par exemple,

```
> sum(i ^ 2, i = 1..100) ; # Calcul de la somme
```

$$\sum_{i=1}^{100} i^2$$

338350

Maple permet aussi de calculer des sommes indéfinies, c'est-à-dire des sommes dont la borne supérieure n'est pas fixée.

```
> sum(i ^ 2, i = 1..n) ;
```

$$\frac{1}{3}(n+1)^3 - \frac{1}{2}(n+1)^2 + \frac{1}{6}n + \frac{1}{6}$$

Maple permet de calculer des sommes à l'aide de la fonction `sum`. Ainsi pour sommer une expression  $u(i)$  pour  $i$  variant de  $a$  à  $b$  on écrit :

```
> sum(u(i), i : a..b) ;
```

Par exemple,

```
> sum(i ^ 2, i = 1..100) ; # Calcul de la somme
```

$$\sum_{i=1}^{100} i^2$$

338350

Maple permet aussi de calculer des sommes indéfinies, c'est-à-dire des sommes dont la borne supérieure n'est pas fixée.

```
> sum(i ^ 2, i = 1..n) ;
```

$$\frac{1}{3}(n+1)^3 - \frac{1}{2}(n+1)^2 + \frac{1}{6}n + \frac{1}{6}$$

Maple permet de calculer des sommes à l'aide de la fonction `sum`. Ainsi pour sommer une expression  $u(i)$  pour  $i$  variant de  $a$  à  $b$  on écrit :

```
> sum(u(i), i : a..b) ;
```

Par exemple,

```
> sum(i ^ 2, i = 1..100) ; # Calcul de la somme
```

$$\sum_{i=1}^{100} i^2$$

338350

Maple permet aussi de calculer des sommes indéfinies, c'est-à-dire des sommes dont la borne supérieure n'est pas fixée.

```
> sum(i ^ 2, i = 1..n) ;
```

$$\frac{1}{3}(n+1)^3 - \frac{1}{2}(n+1)^2 + \frac{1}{6}n + \frac{1}{6}$$

Maple permet de calculer des sommes à l'aide de la fonction `sum`. Ainsi pour somme une expression  $u(i)$  pour  $i$  variant de  $a$  à  $b$  on écrit :

```
> sum(u(i), i : a..b) ;
```

Par exemple,

```
> sum(i ^ 2, i = 1..100) ; # Calcul de la somme
```

$$\sum_{i=1}^{100} i^2$$

338350

Maple permet aussi de calculer des sommes indéfinies, c'est-à-dire des sommes dont la borne supérieure n'est pas fixée.

```
> sum(i ^ 2, i = 1..n) ;
```

$$\frac{1}{3}(n+1)^3 - \frac{1}{2}(n+1)^2 + \frac{1}{6}n + \frac{1}{6}$$

Maple permet de calculer des sommes à l'aide de la fonction `sum`. Ainsi pour somme une expression  $u(i)$  pour  $i$  variant de  $a$  à  $b$  on écrit :

```
> sum(u(i), i : a..b) ;
```

Par exemple,

```
> sum(i ^ 2, i = 1..100) ; # Calcul de la somme
```

$$\sum_{i=1}^{100} i^2$$

338350

Maple permet aussi de calculer des sommes indéfinies, c'est-à-dire des sommes dont la borne supérieure n'est pas fixée.

```
> sum(i ^ 2, i = 1..n) ;
```

$$\frac{1}{3}(n+1)^3 - \frac{1}{2}(n+1)^2 + \frac{1}{6}n + \frac{1}{6}$$

Maple permet de calculer des sommes à l'aide de la fonction `sum`. Ainsi pour somme une expression  $u(i)$  pour  $i$  variant de  $a$  à  $b$  on écrit :

```
> sum(u(i), i : a..b) ;
```

Par exemple,

```
> sum(i ^ 2, i = 1..100) ; # Calcul de la somme
```

$$\sum_{i=1}^{100} i^2$$

338350

Maple permet aussi de calculer des sommes indéfinies, c'est-à-dire des sommes dont la borne supérieure n'est pas fixée.

```
> sum(i ^ 2, i = 1..n) ;
```

$$\frac{1}{3}(n+1)^3 - \frac{1}{2}(n+1)^2 + \frac{1}{6}n + \frac{1}{6}$$

Maple permet de calculer des sommes à l'aide de la fonction `sum`. Ainsi pour somme une expression  $u(i)$  pour  $i$  variant de  $a$  à  $b$  on écrit :

```
> sum(u(i), i : a..b) ;
```

Par exemple,

```
> sum(i ^ 2, i = 1..100) ; # Calcul de la somme
```

$$\sum_{i=1}^{100} i^2$$

338350

Maple permet aussi de calculer des sommes indéfinies, c'est-à-dire des sommes dont la borne supérieure n'est pas fixée.

```
> sum(i ^ 2, i = 1..n) ;
```

$$\frac{1}{3}(n+1)^3 - \frac{1}{2}(n+1)^2 + \frac{1}{6}n + \frac{1}{6}$$

On peut aussi calculer des sommes de séries :

```
> sum(1/k ^ 2, k=1..infinity); # Calcul de
```

$$\sum_{k=1}^{\infty} \frac{1}{k^2}$$

$$\frac{\pi^2}{6}$$

Pour obtenir une forme inerte, on utilise la commande Sum  
comme suit :

```
> Sum(1/k ^ 2, k=1..infinity) =  
sum(1/k ^ 2, k=1..infinity);
```

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

On peut aussi calculer des sommes de séries :

> sum(1/k ^ 2, k=1..infinity); # Calcul de

$$\sum_{k=1}^{\infty} \frac{1}{k^2}$$

$$\frac{\pi^2}{6}$$

Pour obtenir une forme inerte, on utilise la commande Sum  
comme suit :

> Sum(1/k ^ 2, k=1..infinity) =  
sum(1/k ^ 2, k=1..infinity);

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

On peut aussi calculer des sommes de séries :

> sum(1/k ^ 2, k=1..infinity); # Calcul de

$$\sum_{k=1}^{\infty} \frac{1}{k^2}$$

$$\frac{\pi^2}{6}$$

Pour obtenir une forme inerte, on utilise la commande Sum  
comme suit :

> Sum(1/k ^ 2, k=1..infinity) =  
sum(1/k ^ 2, k=1..infinity);

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

On peut aussi calculer des sommes de séries :

> sum(1/k ^ 2, k=1..infinity); # Calcul de

$$\sum_{k=1}^{\infty} \frac{1}{k^2}$$

$$\frac{\pi^2}{6}$$

Pour obtenir une forme inerte, on utilise la commande Sum  
comme suit :

> Sum(1/k ^ 2, k=1..infinity) =  
sum(1/k ^ 2, k=1..infinity);

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

On peut aussi calculer des sommes de séries :

> sum(1/k ^ 2, k=1..infinity); # Calcul de

$$\sum_{k=1}^{\infty} \frac{1}{k^2}$$

$$\frac{\pi^2}{6}$$

Pour obtenir une forme inerte, on utilise la commande Sum  
comme suit :

> Sum(1/k ^ 2, k=1..infinity) =  
sum(1/k ^ 2, k=1..infinity);

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

On peut aussi calculer des sommes de séries :

> sum(1/k ^ 2, k=1..infinity); # Calcul de

$$\sum_{k=1}^{\infty} \frac{1}{k^2}$$

$$\frac{\pi^2}{6}$$

Pour obtenir une forme inerte, on utilise la commande Sum  
comme suit :

> Sum(1/k ^ 2, k=1..infinity) =  
sum(1/k ^ 2, k=1..infinity);

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

On peut aussi calculer des sommes de séries :

> sum(1/k ^ 2, k=1..infinity); # Calcul de

$$\sum_{k=1}^{\infty} \frac{1}{k^2}$$

$$\frac{\pi^2}{6}$$

Pour obtenir une forme inerte, on utilise la commande Sum  
comme suit :

> Sum(1/k ^ 2, k=1..infinity) =  
sum(1/k ^ 2, k=1..infinity);

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

Si une série diverge, Maple renvoie alors l'infini.

```
> sum(1/k, k=1..infinity);
```

$\infty$

Si une série diverge, Maple renvoie alors l'infini.

```
> sum(1/k, k=1..infinity) ;
```

$\infty$

Si une série diverge, Maple renvoie alors l'infini.

```
> sum(1/k, k=1..infinity) ;
```

$\infty$

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

## 1 Calculs sur les nombres entiers et réels

- Calculs sur les nombres réels
- Calculs sur les entiers
- Les nombres premiers
- Analyse combinatoire
- Sommation
- **Produit**

## 2 Les nombres complexes

## 3 Fonctions numériques

- Dérivation
- Limites
- Intégration
- Développements limités

## 4 Expressions symboliques

- Polynômes
- Fractions rationnelles

De même que l'on vient de réaliser des sommes, Maple permet également de calculer des produits à l'aide de la fonction `product`.

Ainsi pour calculer le produit des  $u(i)$  pour  $i$  variant de  $a$  à  $b$  :

```
> product (u(i), i=a..b) ;
```

Par exemple,

```
> product (i ^ 2, i = 1..100) ; # Calcul de  $\prod_{i=1}^{100} i^2$ 
```

1316819440000

De même que l'on vient de réaliser des sommes, Maple permet également de calculer des produits à l'aide de la fonction `product`.

Ainsi pour calculer le produit des  $u(i)$  pour  $i$  variant de  $a$  à  $b$  :

```
> product (u(i), i=a..b) ;
```

Par exemple,

```
> product (i ^ 2, i = 1..100) ; # Calcul de  $\prod_{i=1}^{100} i^2$ 
```

1316819440000

De même que l'on vient de réaliser des sommes, Maple permet également de calculer des produits à l'aide de la fonction `product`.

Ainsi pour calculer le produit des  $u(i)$  pour  $i$  variant de  $a$  à  $b$  :

```
> product (u (i) , i=a..b) ;
```

Par exemple,

```
> product (i ^ 2, i = 1..100) ; # Calcul de  $\prod_{i=1}^{100} i^2$ 
```

1316819440000

De même que l'on vient de réaliser des sommes, Maple permet également de calculer des produits à l'aide de la fonction `product`.

Ainsi pour calculer le produit des  $u(i)$  pour  $i$  variant de  $a$  à  $b$  :

```
> product (u (i) , i=a..b) ;
```

Par exemple,

```
> product (i ^ 2, i = 1..100) ; # Calcul de  $\prod_{i=1}^{100} i^2$ 
```

1316819440000

De même que l'on vient de réaliser des sommes, Maple permet également de calculer des produits à l'aide de la fonction `product`.

Ainsi pour calculer le produit des  $u(i)$  pour  $i$  variant de  $a$  à  $b$  :

```
> product (u (i) , i=a..b) ;
```

Par exemple,

```
> product (i ^ 2, i = 1..100) ; # Calcul de  $\prod_{i=1}^{100} i^2$ 
```

1316819440000

De même que l'on vient de réaliser des sommes, Maple permet également de calculer des produits à l'aide de la fonction `product`.

Ainsi pour calculer le produit des  $u(i)$  pour  $i$  variant de  $a$  à  $b$  :

```
> product (u (i) , i=a..b) ;
```

Par exemple,

```
> product (i ^ 2, i = 1..100) ; # Calcul de  $\prod_{i=1}^{100} i^2$ 
```

1316819440000

Comme pour la fonction somme, on obtient la forme inerte en entrant la fonction avec un "P" majuscule :

```
> Product(k, k=1..n) ;
```

$$\prod_{k=1}^n k$$

Comme pour la fonction somme, on obtient la forme inerte en entrant la fonction avec un "P" majuscule :

> Product (k, k=1..n) ;

$$\prod_{k=1}^n k$$

Comme pour la fonction somme, on obtient la forme inerte en entrant la fonction avec un "P" majuscule :

> Product (k, k=1..n) ;

$$\prod_{k=1}^n k$$

Le nombre complexe  $i$  est représenté par  $I$  dans Maple.

```
> I ^ 2 ;
```

$-1$

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

```
> (4+3*I) + (7-2*I) ;
```

$11 + I$

```
> (1+I) * (2-3*I) ;
```

$5 - I$

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions  $\text{Re}$  et  $\text{Im}$  :

```
> Re(1+2*I) ;
```

$1$

```
> Im(1+2*I) ;
```

$2$

Le nombre complexe  $i$  est représenté par  $I$  dans Maple.

```
> I ^ 2 ;
```

$-1$

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

```
> (4+3*I) + (7-2*I) ;
```

$11 + I$

```
> (1+I) * (2-3*I) ;
```

$5 - I$

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions  $\text{Re}$  et  $\text{Im}$  :

```
> Re(1+2*I) ;
```

$1$

```
> Im(1+2*I) ;
```

$2$

Le nombre complexe  $i$  est représenté par `I` dans Maple.

```
> I ^ 2 ;
```

`-1`

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

```
> (4+3*I) + (7-2*I) ;
```

`11 + I`

```
> (1+I) * (2-3*I) ;
```

`5 - I`

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions `Re` et `Im` :

```
> Re(1+2*I) ;
```

`1`

```
> Im(1+2*I) ;
```

`2`

Le nombre complexe  $i$  est représenté par  $I$  dans Maple.

```
> I ^ 2 ;
```

$-1$

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

```
> (4+3*I) + (7-2*I) ;
```

$11 + I$

```
> (1+I) * (2-3*I) ;
```

$5 - I$

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions  $\text{Re}$  et  $\text{Im}$  :

```
> Re(1+2*I) ;
```

$1$

```
> Im(1+2*I) ;
```

$2$

Le nombre complexe  $i$  est représenté par  $I$  dans Maple.

```
> I ^ 2 ;
```

$-1$

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

```
> (4+3*I) + (7-2*I) ;
```

$11 + I$

```
> (1+I) * (2-3*I) ;
```

$5 - I$

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions  $\text{Re}$  et  $\text{Im}$  :

```
> Re(1+2*I) ;
```

$1$

```
> Im(1+2*I) ;
```

$2$

Le nombre complexe  $i$  est représenté par  $I$  dans Maple.

```
> I ^ 2 ;
```

$-1$

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

```
> (4+3*I) + (7-2*I) ;
```

$11 + I$

```
> (1+I) * (2-3*I) ;
```

$5 - I$

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions  $\text{Re}$  et  $\text{Im}$  :

```
> Re(1+2*I) ;
```

$1$

```
> Im(1+2*I) ;
```

$2$

Le nombre complexe  $i$  est représenté par `I` dans Maple.

```
> I ^ 2 ;
```

`-1`

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

```
> (4+3*I) + (7-2*I) ;
```

`11 + I`

```
> (1+I) * (2-3*I) ;
```

`5 - I`

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions `Re` et `Im` :

```
> Re(1+2*I) ;
```

`1`

```
> Im(1+2*I) ;
```

`2`

Le nombre complexe  $i$  est représenté par `I` dans Maple.

```
> I ^ 2 ;
```

`-1`

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

```
> (4+3*I) + (7-2*I) ;
```

`11 + I`

```
> (1+I) * (2-3*I) ;
```

`5 - I`

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions `Re` et `Im` :

```
> Re(1+2*I) ;
```

`1`

```
> Im(1+2*I) ;
```

`2`

Le nombre complexe  $i$  est représenté par `I` dans Maple.

```
> I ^ 2 ;
```

`-1`

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

```
> (4+3*I) + (7-2*I) ;
```

`11 + I`

```
> (1+I) * (2-3*I) ;
```

`5 - I`

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions `Re` et `Im` :

```
> Re(1+2*I) ;
```

`1`

```
> Im(1+2*I) ;
```

`2`

Le nombre complexe  $i$  est représenté par `I` dans Maple.

```
> I ^ 2 ;
```

`-1`

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

```
> (4+3*I) + (7-2*I) ;
```

`11 + I`

```
> (1+I) * (2-3*I) ;
```

`5 - I`

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions `Re` et `Im` :

```
> Re(1+2*I) ;
```

`1`

```
> Im(1+2*I) ;
```

`2`

Le nombre complexe  $i$  est représenté par  $I$  dans Maple.

```
> I ^ 2 ;
```

$-1$

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

```
> (4+3*I) + (7-2*I) ;
```

$11 + I$

```
> (1+I) * (2-3*I) ;
```

$5 - I$

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions  $\text{Re}$  et  $\text{Im}$  :

```
> Re (1+2*I) ;
```

$1$

```
> Im (1+2*I) ;
```

$2$

Le nombre complexe  $i$  est représenté par  $I$  dans Maple.

>  $I^2$  ;

$-1$

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

>  $(4+3*I) + (7-2*I)$  ;

$11 + I$

>  $(1+I) * (2-3*I)$  ;

$5 - I$

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions  $\text{Re}$  et  $\text{Im}$  :

>  $\text{Re}(1+2*I)$  ;

$1$

>  $\text{Im}(1+2*I)$  ;

Le nombre complexe  $i$  est représenté par  $I$  dans Maple.

>  $I^2$  ;

$-1$

À partir de là, toutes les opérations vues avec les nombres réels peuvent être réalisées avec des nombres complexes :

>  $(4+3*I) + (7-2*I)$  ;

$11 + I$

>  $(1+I) * (2-3*I)$  ;

$5 - I$

On peut isoler partie réelle et partie imaginaire à l'aide des fonctions  $\text{Re}$  et  $\text{Im}$  :

>  $\text{Re}(1+2*I)$  ;

$1$

>  $\text{Im}(1+2*I)$  ;

$2$

On peut également calculer le conjugué :

```
> conjugate(4-3*I);
```

$$4 + 3I$$

L'écriture d'un nombre sous sa forme polaire se fait tout aussi aisément :

```
> polar(2,Pi/6); # Représente le nombre  $2e^{j\frac{\pi}{6}}$ 
```

$$\text{polar}\left(2, \frac{1}{6}\pi\right)$$

On peut alors passer à la forme cartésienne de ce nombre :

```
> evalc(%);
```

$$\sqrt{3} + I$$

On peut également calculer le conjugué :

> conjugate(4-3\*I) ;

$$4 + 3I$$

L'écriture d'un nombre sous sa forme polaire se fait tout aussi aisément :

> polar(2,Pi/6) ; # Représente le nombre  $2e^{j\frac{\pi}{6}}$

$$\text{polar}\left(2, \frac{1}{6}\pi\right)$$

On peut alors passer à la forme cartésienne de ce nombre :

> evalc(%);

$$\sqrt{3} + I$$

On peut également calculer le conjugué :

```
> conjugate(4-3*I) ;
```

$$4 + 3I$$

L'écriture d'un nombre sous sa forme polaire se fait tout aussi aisément :

```
> polar(2,Pi/6) ; # Représente le nombre  $2e^{j\frac{\pi}{6}}$ 
```

$$\text{polar}\left(2, \frac{1}{6}\pi\right)$$

On peut alors passer à la forme cartésienne de ce nombre :

```
> evalc(%);
```

$$\sqrt{3} + I$$

On peut également calculer le conjugué :

> conjugate(4-3\*I) ;

$$4 + 3I$$

L'écriture d'un nombre sous sa forme polaire se fait tout aussi aisément :

> polar(2,Pi/6) ; # Représente le nombre  $2e^{j\frac{\pi}{6}}$

$$\text{polar}\left(2, \frac{1}{6}\pi\right)$$

On peut alors passer à la forme cartésienne de ce nombre :

> evalc(%);

$$\sqrt{3} + I$$

On peut également calculer le conjugué :

> conjugate(4-3\*I) ;

$$4 + 3I$$

L'écriture d'un nombre sous sa forme polaire se fait tout aussi aisément :

> polar(2, Pi/6) ; # Représente le nombre  $2e^{j\frac{\pi}{6}}$

$$\text{polar}\left(2, \frac{1}{6}\pi\right)$$

On peut alors passer à la forme cartésienne de ce nombre :

> evalc(%);

$$\sqrt{3} + I$$

On peut également calculer le conjugué :

```
> conjugate(4-3*I) ;
```

$$4 + 3I$$

L'écriture d'un nombre sous sa forme polaire se fait tout aussi aisément :

```
> polar(2, Pi/6) ; # Représente le nombre  $2e^{j\frac{\pi}{6}}$ 
```

$$\text{polar}\left(2, \frac{1}{6}\pi\right)$$

On peut alors passer à la forme cartésienne de ce nombre :

```
> evalc(%);
```

$$\sqrt{3} + I$$

On peut également calculer le conjugué :

```
> conjugate(4-3*I) ;
```

$$4 + 3I$$

L'écriture d'un nombre sous sa forme polaire se fait tout aussi aisément :

```
> polar(2, Pi/6) ; # Représente le nombre  $2e^{j\frac{\pi}{6}}$ 
```

$$\text{polar}\left(2, \frac{1}{6}\pi\right)$$

On peut alors passer à la forme cartésienne de ce nombre :

```
> evalc(%);
```

$$\sqrt{3} + I$$

On peut également calculer le conjugué :

```
> conjugate(4-3*I) ;
```

$$4 + 3I$$

L'écriture d'un nombre sous sa forme polaire se fait tout aussi aisément :

```
> polar(2, Pi/6) ; # Représente le nombre  $2e^{i\frac{\pi}{6}}$ 
```

$$\text{polar}\left(2, \frac{1}{6}\pi\right)$$

On peut alors passer à la forme cartésienne de ce nombre :

```
> evalc(%);
```

$$\sqrt{3} + I$$

On peut également calculer le conjugué :

> conjugate(4-3\*I) ;

$$4 + 3I$$

L'écriture d'un nombre sous sa forme polaire se fait tout aussi aisément :

> polar(2, Pi/6) ; # Représente le nombre  $2e^{j\frac{\pi}{6}}$

$$\text{polar}\left(2, \frac{1}{6}\pi\right)$$

On peut alors passer à la forme cartésienne de ce nombre :

> evalc(%);

$$\sqrt{3} + I$$

On peut également calculer le conjugué :

> conjugate(4-3\*I) ;

$$4 + 3I$$

L'écriture d'un nombre sous sa forme polaire se fait tout aussi aisément :

> polar(2, Pi/6) ; # Représente le nombre  $2e^{i\frac{\pi}{6}}$

$$\text{polar}\left(2, \frac{1}{6}\pi\right)$$

On peut alors passer à la forme cartésienne de ce nombre :

> evalc(%);

$$\sqrt{3} + I$$

On peut vérifier qu'il s'agit bien du même nombre. On calcule son module :

```
> abs(%);
```

2

Puis on détermine son argument :

```
> argument(%%);
```

$\frac{\pi}{6}$

On peut vérifier qu'il s'agit bien du même nombre. On calcule son module :

```
> abs (%) ;
```

2

Puis on détermine son argument :

```
> argument (%) ;
```

$\frac{\pi}{6}$

On peut vérifier qu'il s'agit bien du même nombre. On calcule son module :

```
> abs (%) ;
```

2

Puis on détermine son argument :

```
> argument (%%) ;
```

$\frac{\pi}{6}$

On peut vérifier qu'il s'agit bien du même nombre. On calcule son module :

```
> abs (%) ;
```

2

Puis on détermine son argument :

```
> argument (%%) ;
```

$\frac{\pi}{6}$

On peut vérifier qu'il s'agit bien du même nombre. On calcule son module :

```
> abs (%) ;
```

2

Puis on détermine son argument :

```
> argument (%%) ;
```

$\frac{\pi}{6}$

On peut vérifier qu'il s'agit bien du même nombre. On calcule son module :

```
> abs (%) ;
```

2

Puis on détermine son argument :

```
> argument (%%) ;
```

$\frac{\pi}{6}$

On peut vérifier qu'il s'agit bien du même nombre. On calcule son module :

```
> abs (%) ;
```

2

Puis on détermine son argument :

```
> argument (%%) ;
```

$\frac{\pi}{6}$

Avant de présenter les fonctions, il convient de revenir sur cette notion. Commençons par entrer une expression fonctionnelle en mémoire :

```
> a := x ^ 2 + 2*x+1 ;
```

$$a := x^2 + 2x + 1$$

On peut penser - on le verra à tort - que  $a(1)$  peut renvoyer 4 ( $= 1^2 + 2 + 1$ ). Mais en fait, on a

```
> a(1) ;
```

$$x(1)^2 + 2x(1) + 1$$

Avant de présenter les fonctions, il convient de revenir sur cette notion. Commençons par entrer une expression fonctionnelle en mémoire :

```
> a := x ^ 2 + 2*x+1 ;
```

$$a := x^2 + 2x + 1$$

On peut penser - on le verra à tort - que  $a(1)$  peut renvoyer 4 ( $= 1^2 + 2 + 1$ ). Mais en fait, on a

```
> a(1) ;
```

$$x(1)^2 + 2x(1) + 1$$

Avant de présenter les fonctions, il convient de revenir sur cette notion. Commençons par entrer une expression fonctionnelle en mémoire :

```
> a := x ^ 2 + 2*x+1 ;
```

$$a := x^2 + 2x + 1$$

On peut penser - on le verra à tort - que  $a(1)$  peut renvoyer 4 ( $= 1^2 + 2 + 1$ ). Mais en fait, on a

```
> a(1) ;
```

$$x(1)^2 + 2x(1) + 1$$

Avant de présenter les fonctions, il convient de revenir sur cette notion. Commençons par entrer une expression fonctionnelle en mémoire :

```
> a := x ^ 2 + 2*x+1 ;
```

$$a := x^2 + 2x + 1$$

On peut penser - on le verra à tort - que  $a(1)$  peut renvoyer 4 ( $= 1^2 + 2 + 1$ ). Mais en fait, on a

```
> a(1) ;
```

$$x(1)^2 + 2x(1) + 1$$

Avant de présenter les fonctions, il convient de revenir sur cette notion. Commençons par entrer une expression fonctionnelle en mémoire :

```
> a := x ^ 2 + 2*x+1 ;
```

$$a := x^2 + 2x + 1$$

On peut penser - on le verra à tort - que  $a(1)$  peut renvoyer 4 ( $= 1^2 + 2 + 1$ ). Mais en fait, on a

```
> a(1) ;
```

$$x(1)^2 + 2x(1) + 1$$

Avant de présenter les fonctions, il convient de revenir sur cette notion. Commençons par entrer une expression fonctionnelle en mémoire :

```
> a := x ^ 2 + 2*x+1 ;
```

$$a := x^2 + 2x + 1$$

On peut penser - on le verra à tord - que  $a(1)$  peut renvoyer 4 ( $= 1^2 + 2 + 1$ ). Mais en fait, on a

```
> a(1) ;
```

$$x(1)^2 + 2x(1) + 1$$

Avant de présenter les fonctions, il convient de revenir sur cette notion. Commençons par entrer une expression fonctionnelle en mémoire :

```
> a := x ^ 2 + 2*x+1 ;
```

$$a := x^2 + 2x + 1$$

On peut penser - on le verra à tort - que  $a(1)$  peut renvoyer 4 ( $= 1^2 + 2 + 1$ ). Mais en fait, on a

```
> a(1) ;
```

$$x(1)^2 + 2x(1) + 1$$

Avant de présenter les fonctions, il convient de revenir sur cette notion. Commençons par entrer une expression fonctionnelle en mémoire :

```
> a := x ^ 2 + 2*x+1 ;
```

$$a := x^2 + 2x + 1$$

On peut penser - on le verra à tort - que  $a(1)$  peut renvoyer 4 ( $= 1^2 + 2 + 1$ ). Mais en fait, on a

```
> a(1) ;
```

$$x(1)^2 + 2x(1) + 1$$

Pour définir une fonction en Maple, il faut employer la syntaxe suivante :

```
> f := x -> x^2 + 2*x + 1 ;
```

$$f := x \rightarrow x^2 + 2x + 1$$

Puis tapons `> f(1) ;`

4

On peut utiliser la fonction `unapply` pour définir une fonction à partir d'une expression :

```
> fonction := unapply(expression, var) ;
```

Par exemple, on crée la même fonction  $f$  par :

```
> g := unapply(a, x) ;
```

$$g := x \rightarrow x^2 + 2x + 1$$

Pour définir une fonction en Maple, il faut employer la syntaxe suivante :

```
> f := x -> x^2 + 2*x + 1 ;
```

$$f := x \rightarrow x^2 + 2x + 1$$

Puis tapons `> f(1) ;`

4

On peut utiliser la fonction `unapply` pour définir une fonction à partir d'une expression :

```
> fonction := unapply(expression, var) ;
```

Par exemple, on crée la même fonction  $f$  par :

```
> g := unapply(a, x) ;
```

$$g := x \rightarrow x^2 + 2x + 1$$

Pour définir une fonction en Maple, il faut employer la syntaxe suivante :

```
> f := x -> x^2 + 2*x + 1 ;
```

$$f := x \rightarrow x^2 + 2x + 1$$

Puis tapons `> f(1) ;`

4

On peut utiliser la fonction `unapply` pour définir une fonction à partir d'une expression :

```
> fonction := unapply(expression, var) ;
```

Par exemple, on crée la même fonction  $f$  par :

```
> g := unapply(a, x) ;
```

$$g := x \rightarrow x^2 + 2x + 1$$

Pour définir une fonction en Maple, il faut employer la syntaxe suivante :

```
> f := x -> x^2 + 2*x + 1 ;
```

$$f := x \rightarrow x^2 + 2x + 1$$

Puis tapons :> f(1) ;

4

On peut utiliser la fonction `unapply` pour définir une fonction à partir d'une expression :

```
> fonction := unapply(expression, var) ;
```

Par exemple, on crée la même fonction  $f$  par :

```
> g := unapply(a, x) ;
```

$$g := x \rightarrow x^2 + 2x + 1$$

Pour définir une fonction en Maple, il faut employer la syntaxe suivante :

```
> f := x -> x^2 + 2*x + 1 ;
```

$$f := x \rightarrow x^2 + 2x + 1$$

Puis tapons `> f(1) ;`

4

On peut utiliser la fonction `unapply` pour définir une fonction à partir d'une expression :

```
> fonction := unapply(expression, var) ;
```

Par exemple, on crée la même fonction  $f$  par :

```
> g := unapply(a, x) ;
```

$$g := x \rightarrow x^2 + 2x + 1$$

Pour définir une fonction en Maple, il faut employer la syntaxe suivante :

```
> f := x -> x^2 + 2*x + 1 ;
```

$$f := x \rightarrow x^2 + 2x + 1$$

Puis tapons `> f(1) ;`

4

On peut utiliser la fonction `unapply` pour définir une fonction à partir d'une expression :

```
> fonction := unapply(expression, var) ;
```

Par exemple, on crée la même fonction  $f$  par :

```
> g := unapply(a, x) ;
```

$$g := x \rightarrow x^2 + 2x + 1$$

Pour définir une fonction en Maple, il faut employer la syntaxe suivante :

```
> f := x -> x^2 + 2*x + 1 ;
```

$$f := x \rightarrow x^2 + 2x + 1$$

Puis tapons `> f(1) ;`

4

On peut utiliser la fonction `unapply` pour définir une fonction à partir d'une expression :

```
> fonction := unapply(expression, var) ;
```

Par exemple, on crée la même fonction  $f$  par :

```
> g := unapply(a, x) ;
```

$$g := x \rightarrow x^2 + 2x + 1$$

Pour définir une fonction en Maple, il faut employer la syntaxe suivante :

```
> f := x -> x^2 + 2*x + 1 ;
```

$$f := x \rightarrow x^2 + 2x + 1$$

Puis tapons `> f(1) ;`

4

On peut utiliser la fonction `unapply` pour définir une fonction à partir d'une expression :

```
> fonction := unapply(expression, var) ;
```

Par exemple, on crée la même fonction  $f$  par :

```
> g := unapply(a, x) ;
```

$$g := x \rightarrow x^2 + 2x + 1$$

Pour définir une fonction en Maple, il faut employer la syntaxe suivante :

```
> f := x -> x^2 + 2*x + 1 ;
```

$$f := x \rightarrow x^2 + 2x + 1$$

Puis tapons `> f(1) ;`

4

On peut utiliser la fonction `unapply` pour définir une fonction à partir d'une expression :

```
> fonction := unapply(expression, var) ;
```

Par exemple, on crée la même fonction  $f$  par :

```
> g := unapply(a, x) ;
```

$$g := x \rightarrow x^2 + 2x + 1$$

Pour définir une fonction en Maple, il faut employer la syntaxe suivante :

```
> f := x -> x^2 + 2*x + 1 ;
```

$$f := x \rightarrow x^2 + 2x + 1$$

Puis tapons `> f(1) ;`

4

On peut utiliser la fonction `unapply` pour définir une fonction à partir d'une expression :

```
> fonction := unapply(expression, var) ;
```

Par exemple, on crée la même fonction  $f$  par :

```
> g := unapply(a, x) ;
```

$$g := x \rightarrow x^2 + 2x + 1$$

Pour définir une fonction en Maple, il faut employer la syntaxe suivante :

```
> f := x -> x^2 + 2*x + 1 ;
```

$$f := x \rightarrow x^2 + 2x + 1$$

Puis tapons `> f(1) ;`

4

On peut utiliser la fonction `unapply` pour définir une fonction à partir d'une expression :

```
> fonction := unapply(expression, var) ;
```

Par exemple, on crée la même fonction  $f$  par :

```
> g := unapply(a, x) ;
```

$$g := x \rightarrow x^2 + 2x + 1$$

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

## 1 Calculs sur les nombres entiers et réels

- Calculs sur les nombres réels
- Calculs sur les entiers
- Les nombres premiers
- Analyse combinatoire
- Sommation
- Produit

## 2 Les nombres complexes

## 3 Fonctions numériques

- Dérivation
- Limites
- Intégration
- Développements limités

## 4 Expressions symboliques

- Polynômes
- Fractions rationnelles

On n'emploie pas la même syntaxe selon que l'objet que l'on doit dériver est une fonction mathématique ou un expression (fonctionnelle).

Par exemple, pour une **expression** :

```
> diff(2*x^2+3, x) ;
```

4x

Comme pour la fonction `sum`, le fait d'entrer `Diff` permet d'obtenir une forme inerte :

```
> Diff(2*x^2+3, x) ;
```

$\frac{\partial}{\partial x}(2x^2 + 3)$

On n'emploie pas la même syntaxe selon que l'objet que l'on doit dériver est une fonction mathématique ou un expression (fonctionnelle).

Par exemple, pour une **expression** :

```
> diff(2*x^2+3, x) ;
```

4x

Comme pour la fonction `sum`, le fait d'entrer `Diff` permet d'obtenir une forme inerte :

```
> Diff(2*x^2+3, x) ;
```

$\frac{\partial}{\partial x}(2x^2 + 3)$

On n'emploie pas la même syntaxe selon que l'objet que l'on doit dériver est une fonction mathématique ou un expression (fonctionnelle).

Par exemple, pour une **expression** :

```
> diff(2*x^2+3, x) ;
```

4x

Comme pour la fonction `sum`, le fait d'entrer `Diff` permet d'obtenir une forme inerte :

```
> Diff(2*x^2+3, x) ;
```

$\frac{\partial}{\partial x}(2x^2 + 3)$

On n'emploie pas la même syntaxe selon que l'objet que l'on doit dériver est une fonction mathématique ou un expression (fonctionnelle).

Par exemple, pour une **expression** :

```
> diff(2*x^2+3, x) ;
```

$4x$

Comme pour la fonction `sum`, le fait d'entrer `Diff` permet d'obtenir une forme inerte :

```
> Diff(2*x^2+3, x) ;
```

$\frac{\partial}{\partial x}(2x^2 + 3)$

On n'emploie pas la même syntaxe selon que l'objet que l'on doit dériver est une fonction mathématique ou un expression (fonctionnelle).

Par exemple, pour une **expression** :

```
> diff(2*x^2+3, x) ;
```

$4x$

Comme pour la fonction `sum`, le fait d'entrer `Diff` permet d'obtenir une forme inerte :

```
> Diff(2*x^2+3, x) ;
```

$\frac{\partial}{\partial x}(2x^2 + 3)$

On n'emploie pas la même syntaxe selon que l'objet que l'on doit dériver est une fonction mathématique ou un expression (fonctionnelle).

Par exemple, pour une **expression** :

```
> diff(2*x^2+3, x) ;
```

$4x$

Comme pour la fonction `sum`, le fait d'entrer `Diff` permet d'obtenir une forme inerte :

```
> Diff(2*x^2+3, x) ;
```

$\frac{\partial}{\partial x}(2x^2 + 3)$

On n'emploie pas la même syntaxe selon que l'objet que l'on doit dériver est une fonction mathématique ou un expression (fonctionnelle).

Par exemple, pour une **expression** :

```
> diff(2*x^2+3, x) ;
```

$4x$

Comme pour la fonction `sum`, le fait d'entrer `Diff` permet d'obtenir une forme inerte :

```
> Diff(2*x^2+3, x) ;
```

$\frac{\partial}{\partial x}(2x^2 + 3)$

On peut dériver par rapport à plusieurs variables ou plusieurs fois par rapport à la même variable :

```
> Diff(Diff(sin(x*y), x), y) =  
diff(diff(sin(x*y), x), y);
```

$$\frac{\partial^2}{\partial y \partial x} \sin(xy) = -\sin(xy)xy + \cos(xy)$$

On a donc dérivé  $\sin(xy)$  par rapport à  $x$  puis par rapport à  $y$ .

```
> diff(2*x^3+3*x^2+4, x, x, x); # On dérive  
trois fois par rapport à x
```

On peut dériver par rapport à plusieurs variables ou plusieurs fois par rapport à la même variable :

```
> Diff(Diff(sin(x*y), x), y) =  
diff(diff(sin(x*y), x), y);
```

$$\frac{\partial^2}{\partial y \partial x} \sin(xy) = -\sin(xy)xy + \cos(xy)$$

On a donc dérivé  $\sin(xy)$  par rapport à  $x$  puis par rapport à  $y$ .

```
> diff(2*x^3+3*x^2+4, x, x, x); # On dérive  
trois fois par rapport à x
```

On peut dériver par rapport à plusieurs variables ou plusieurs fois par rapport à la même variable :

```
> Diff(Diff(sin(x*y), x), y) =  
diff(diff(sin(x*y), x), y);
```

$$\frac{\partial^2}{\partial y \partial x} \sin(xy) = -\sin(xy)xy + \cos(xy)$$

On a donc dérivé  $\sin(xy)$  par rapport à  $x$  puis par rapport à  $y$ .

```
> diff(2*x^3+3*x^2+4, x, x, x); # On dérive  
trois fois par rapport à x
```

On peut dériver par rapport à plusieurs variables ou plusieurs fois par rapport à la même variable :

```
> Diff(Diff(sin(x*y), x), y) =  
diff(diff(sin(x*y), x), y);
```

$$\frac{\partial^2}{\partial y \partial x} \sin(xy) = -\sin(xy)xy + \cos(xy)$$

On a donc dérivé  $\sin(xy)$  par rapport à  $x$  puis par rapport à  $y$ .

```
> diff(2*x^3+3*x^2+4, x, x, x); # On dérive  
trois fois par rapport à x
```

On peut dériver par rapport à plusieurs variables ou plusieurs fois par rapport à la même variable :

```
> Diff(Diff(sin(x*y), x), y) =  
diff(diff(sin(x*y), x), y);
```

$$\frac{\partial^2}{\partial y \partial x} \sin(xy) = -\sin(xy)xy + \cos(xy)$$

On a donc dérivé  $\sin(xy)$  par rapport à  $x$  puis par rapport à  $y$ .

```
> diff(2*x^3+3*x^2+4, x, x, x); # On dérive  
trois fois par rapport à x
```

On peut dériver par rapport à plusieurs variables ou plusieurs fois par rapport à la même variable :

```
> Diff(Diff(sin(x*y), x), y) =  
diff(diff(sin(x*y), x), y);
```

$$\frac{\partial^2}{\partial y \partial x} \sin(xy) = -\sin(xy)xy + \cos(xy)$$

On a donc dérivé  $\sin(xy)$  par rapport à  $x$  puis par rapport à  $y$ .

```
> diff(2*x^3+3*x^2+4, x, x, x); # On dérive  
trois fois par rapport à x
```

On peut dériver par rapport à plusieurs variables ou plusieurs fois par rapport à la même variable :

```
> Diff(Diff(sin(x*y), x), y) =  
diff(diff(sin(x*y), x), y) ;
```

$$\frac{\partial^2}{\partial y \partial x} \sin(xy) = -\sin(xy)xy + \cos(xy)$$

On a donc dérivé  $\sin(xy)$  par rapport à  $x$  puis par rapport à  $y$ .

```
> diff(2*x^3+3*x^2+4, x, x, x) ; # On dérive  
trois fois par rapport à x
```

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

L'instruction `diff(diff(sin(x*y), x), y)` est donc  
identique à `diff(sin(x*y), x, y)`.

Il existe aussi une autre méthode pour dériver plusieurs fois par rapport à la même variable, à l'aide de l'opérateur \$ :

```
> diff(2*x^3+3*x^2+4,x$3);# On dérive trois  
fois en X
```

12

Il existe aussi une autre méthode pour dériver plusieurs fois par rapport à la même variable, à l'aide de l'opérateur \$ :

```
> diff(2*x^3+3*x^2+4, x$3) ; # On dérive trois  
fois en X
```

12

Il existe aussi une autre méthode pour dériver plusieurs fois par rapport à la même variable, à l'aide de l'opérateur \$ :

```
> diff(2*x^3+3*x^2+4, x$3) ; # On dérive trois  
fois en X
```

12

Il existe aussi une autre méthode pour dériver plusieurs fois par rapport à la même variable, à l'aide de l'opérateur \$ :

```
> diff(2*x^3+3*x^2+4, x$3) ; # On dérive trois  
fois en X
```

12

Pour dériver une fonction  $f$  on utilise l'opérateur  $D$  :

>  $D(f)$  ;

Si  $f$  est une fonction de plusieurs variables  $x_1, \dots, x_m$ ,

>  $D[i_1, \dots, i_k](f)$  ;

permet d'effectuer la dérivation  $\frac{\partial^k}{\partial x_{i_1} \dots \partial x_{i_k}} f$ .

Exemple :

>  $f := x \rightarrow x^2 + 2 * x + 1$  ;

>  $D(f)(x)$  ;

$$2x + 2$$

On peut alors très simplement calculer le nombre dérivé en un point :

>  $D(f)(2)$  ;

$$6$$

Pour dériver une fonction  $f$  on utilise l'opérateur  $D$  :

>  $D(f)$  ;

Si  $f$  est une fonction de plusieurs variables  $x_1, \dots, x_m$ ,

>  $D[i_1, \dots, i_k](f)$  ;

permet d'effectuer la dérivation  $\frac{\partial^k}{\partial x_{i_1} \dots \partial x_{i_k}} f$ .

Exemple :

>  $f := x \rightarrow x^2 + 2 * x + 1$  ;

>  $D(f)(x)$  ;

$$2x + 2$$

On peut alors très simplement calculer le nombre dérivé en un point :

>  $D(f)(2)$  ;

$$6$$

Pour dériver une fonction  $f$  on utilise l'opérateur  $D$  :

>  $D(f)$  ;

Si  $f$  est une fonction de plusieurs variables  $x_1, \dots, x_m$ ,

>  $D[i_1, \dots, i_k](f)$  ;

permet d'effectuer la dérivation  $\frac{\partial^k}{\partial x_{i_1} \dots \partial x_{i_k}} f$ .

Exemple :

>  $f := x \rightarrow x^2 + 2 * x + 1$  ;

>  $D(f)(x)$  ;

$2x + 2$

On peut alors très simplement calculer le nombre dérivé en un point :

>  $D(f)(2)$  ;

6

Pour dériver une fonction  $f$  on utilise l'opérateur  $D$  :

>  $D(f)$  ;

Si  $f$  est une fonction de plusieurs variables  $x_1, \dots, x_m$ ,

>  $D[i_1, \dots, i_k](f)$  ;

permet d'effectuer la dérivation  $\frac{\partial^k}{\partial x_{i_1} \dots \partial x_{i_k}} f$ .

Exemple :

>  $f := x \rightarrow x^2 + 2 * x + 1$  ;

>  $D(f)(x)$  ;

$2x + 2$

On peut alors très simplement calculer le nombre dérivé en un point :

>  $D(f)(2)$  ;

6

Pour dériver une fonction  $f$  on utilise l'opérateur  $D$  :

>  $D(f)$  ;

Si  $f$  est une fonction de plusieurs variables  $x_1, \dots, x_m$ ,

>  $D[i_1, \dots, i_k](f)$  ;

permet d'effectuer la dérivation  $\frac{\partial^k}{\partial x_{i_1} \dots \partial x_{i_k}} f$ .

Exemple :

>  $f := x \rightarrow x^2 + 2 * x + 1$  ;

>  $D(f)(x)$  ;

$2x + 2$

On peut alors très simplement calculer le nombre dérivé en un point :

>  $D(f)(2)$  ;

6

Pour dériver une fonction  $f$  on utilise l'opérateur  $D$  :

>  $D(f)$  ;

Si  $f$  est une fonction de plusieurs variables  $x_1, \dots, x_m$ ,

>  $D[i_1, \dots, i_k](f)$  ;

permet d'effectuer la dérivation  $\frac{\partial^k}{\partial x_{i_1} \dots \partial x_{i_k}} f$ .

Exemple :

>  $f := x \rightarrow x^2 + 2 * x + 1$  ;

>  $D(f)(x)$  ;

$2x + 2$

On peut alors très simplement calculer le nombre dérivé en un point :

>  $D(f)(2)$  ;

6

Pour dériver une fonction  $f$  on utilise l'opérateur  $D$  :

>  $D(f)$  ;

Si  $f$  est une fonction de plusieurs variables  $x_1, \dots, x_m$ ,

>  $D[i_1, \dots, i_k](f)$  ;

permet d'effectuer la dérivation  $\frac{\partial^k}{\partial x_{i_1} \dots \partial x_{i_k}} f$ .

Exemple :

>  $f := x \rightarrow x^2 + 2 * x + 1$  ;

>  $D(f)(x)$  ;

$2x + 2$

On peut alors très simplement calculer le nombre dérivé en un point :

>  $D(f)(2)$  ;

6

Pour dériver une fonction  $f$  on utilise l'opérateur  $D$  :

>  $D(f)$  ;

Si  $f$  est une fonction de plusieurs variables  $x_1, \dots, x_m$ ,

>  $D[i_1, \dots, i_k](f)$  ;

permet d'effectuer la dérivation  $\frac{\partial^k}{\partial x_{i_1} \dots \partial x_{i_k}} f$ .

Exemple :

>  $f := x \rightarrow x^2 + 2 * x + 1$  ;

>  $D(f)(x)$  ;

$2x + 2$

On peut alors très simplement calculer le nombre dérivé en un point :

>  $D(f)(2)$  ;

6

Pour dériver une fonction  $f$  on utilise l'opérateur  $D$  :

>  $D(f)$  ;

Si  $f$  est une fonction de plusieurs variables  $x_1, \dots, x_m$ ,

>  $D[i_1, \dots, i_k](f)$  ;

permet d'effectuer la dérivation  $\frac{\partial^k}{\partial x_{i_1} \dots \partial x_{i_k}} f$ .

Exemple :

>  $f := x \rightarrow x^2 + 2 * x + 1$  ;

>  $D(f)(x)$  ;

$$2x + 2$$

On peut alors très simplement calculer le nombre dérivé en un point :

>  $D(f)(2)$  ;

6

Pour dériver une fonction  $f$  on utilise l'opérateur  $D$  :

>  $D(f)$  ;

Si  $f$  est une fonction de plusieurs variables  $x_1, \dots, x_m$ ,

>  $D[i_1, \dots, i_k](f)$  ;

permet d'effectuer la dérivation  $\frac{\partial^k}{\partial x_{i_1} \dots \partial x_{i_k}} f$ .

Exemple :

>  $f := x \rightarrow x^2 + 2 * x + 1$  ;

>  $D(f)(x)$  ;

$$2x + 2$$

On peut alors très simplement calculer le nombre dérivé en un point :

>  $D(f)(2)$  ;

6

Pour dériver une fonction  $f$  on utilise l'opérateur  $D$  :

>  $D(f)$  ;

Si  $f$  est une fonction de plusieurs variables  $x_1, \dots, x_m$ ,

>  $D[i_1, \dots, i_k](f)$  ;

permet d'effectuer la dérivation  $\frac{\partial^k}{\partial x_{i_1} \dots \partial x_{i_k}} f$ .

Exemple :

>  $f := x \rightarrow x^2 + 2 * x + 1$  ;

>  $D(f)(x)$  ;

$$2x + 2$$

On peut alors très simplement calculer le nombre dérivé en un point :

>  $D(f)(2)$  ;

Pour dériver une fonction  $f$  on utilise l'opérateur  $D$  :

>  $D(f)$  ;

Si  $f$  est une fonction de plusieurs variables  $x_1, \dots, x_m$ ,

>  $D[i_1, \dots, i_k](f)$  ;

permet d'effectuer la dérivation  $\frac{\partial^k}{\partial x_{i_1} \dots \partial x_{i_k}} f$ .

Exemple :

>  $f := x \rightarrow x^2 + 2 * x + 1$  ;

>  $D(f)(x)$  ;

$$2x + 2$$

On peut alors très simplement calculer le nombre dérivé en un point :

>  $D(f)(2)$  ;

6

Pour dériver plusieurs fois :

```
> D@@2 (f) (x) ; # Pour calculer  $f''(x)$ 
```

2

Pour dériver plusieurs fois :

```
> D@@2 (f) (x) ; # Pour calculer  $f''(x)$ 
```

2

Pour dériver plusieurs fois :

> D@@2 (f) (x) ;# Pour calculer  $f''(x)$

2

Pour dériver plusieurs fois :

> D@@2 (f) (x) ;# Pour calculer  $f''(x)$

2

Intéressons-nous aux fonctions de plusieurs variables :

> g := (x, y) -> 2\*x^2\*y^3 + 3\*y\*x + x\*sin(y) + y + 1 ;

$$g := (x, y) \rightarrow 2x^2y^3 + 3yx + x \sin(y) + y + 1$$

> D[1](g)(x, y) ; # Correspond à  $\frac{\partial}{\partial x}g(x, y)$

$$4xy^3 + 3y + \sin(y)$$

> D[1, 2](g)(x, y) ; # Correspond à  $\frac{\partial^2}{\partial y \partial x}g(x, y)$

$$12xy^2 + 3 + \cos(y)$$

> D[1\$2, 2](g)(x, y) ; # Correspond à  $\frac{\partial^3}{\partial y \partial x^2}g(x, y)$

$$12y^2$$

Intéressons-nous aux fonctions de plusieurs variables :

> g := (x, y) -> 2\*x^2\*y^3 + 3\*y\*x + x\*sin(y) + y + 1 ;

$$g := (x, y) \rightarrow 2x^2y^3 + 3yx + x \sin(y) + y + 1$$

> D[1](g)(x, y) ; # Correspond à  $\frac{\partial}{\partial x}g(x, y)$

$$4xy^3 + 3y + \sin(y)$$

> D[1, 2](g)(x, y) ; # Correspond à  $\frac{\partial^2}{\partial y \partial x}g(x, y)$

$$12xy^2 + 3 + \cos(y)$$

> D[1\$2, 2](g)(x, y) ; # Correspond à  $\frac{\partial^3}{\partial y \partial x^2}g(x, y)$

$$12y^2$$

Intéressons-nous aux fonctions de plusieurs variables :

> g := (x, y) -> 2\*x^2\*y^3 + 3\*y\*x + x\*sin(y) + y + 1 ;

$$g := (x, y) \rightarrow 2x^2y^3 + 3yx + x \sin(y) + y + 1$$

> D[1](g)(x, y) ; # Correspond à  $\frac{\partial}{\partial x}g(x, y)$

$$4xy^3 + 3y + \sin(y)$$

> D[1,2](g)(x, y) ; # Correspond à  $\frac{\partial^2}{\partial y \partial x}g(x, y)$

$$12xy^2 + 3 + \cos(y)$$

> D[1\$2,2](g)(x, y) ; # Correspond à  $\frac{\partial^3}{\partial y \partial x^2}g(x, y)$

$$12y^2$$

Intéressons-nous aux fonctions de plusieurs variables :

> g := (x, y) -> 2\*x^2\*y^3 + 3\*y\*x + x\*sin(y) + y + 1 ;

$$g := (x, y) \rightarrow 2x^2y^3 + 3yx + x \sin(y) + y + 1$$

> D[1](g)(x, y) ; # Correspond à  $\frac{\partial}{\partial x}g(x, y)$

$$4xy^3 + 3y + \sin(y)$$

> D[1,2](g)(x, y) ; # Correspond à  $\frac{\partial^2}{\partial y \partial x}g(x, y)$

$$12xy^2 + 3 + \cos(y)$$

> D[1\$2,2](g)(x, y) ; # Correspond à  $\frac{\partial^3}{\partial y \partial x^2}g(x, y)$

$$12y^2$$

Intéressons-nous aux fonctions de plusieurs variables :

> g := (x, y) -> 2\*x^2\*y^3 + 3\*y\*x + x\*sin(y) + y + 1 ;

$$g := (x, y) \rightarrow 2x^2y^3 + 3yx + x \sin(y) + y + 1$$

> D[1](g)(x, y) ; # Correspond à  $\frac{\partial}{\partial x}g(x, y)$

$$4xy^3 + 3y + \sin(y)$$

> D[1,2](g)(x, y) ; # Correspond à  $\frac{\partial^2}{\partial y \partial x}g(x, y)$

$$12xy^2 + 3 + \cos(y)$$

> D[1\$2,2](g)(x, y) ; # Correspond à  $\frac{\partial^3}{\partial y \partial x^2}g(x, y)$

$$12y^2$$

Intéressons-nous aux fonctions de plusieurs variables :

> g := (x, y) -> 2\*x^2\*y^3 + 3\*y\*x + x\*sin(y) + y + 1 ;

$$g := (x, y) \rightarrow 2x^2y^3 + 3yx + x \sin(y) + y + 1$$

> D[1](g)(x, y) ; # Correspond à  $\frac{\partial}{\partial x}g(x, y)$

$$4xy^3 + 3y + \sin(y)$$

> D[1,2](g)(x, y) ; # Correspond à  $\frac{\partial^2}{\partial y \partial x}g(x, y)$

$$12xy^2 + 3 + \cos(y)$$

> D[1\$2,2](g)(x, y) ; # Correspond à  $\frac{\partial^3}{\partial y \partial x^2}g(x, y)$

$$12y^2$$

Intéressons-nous aux fonctions de plusieurs variables :

> g := (x, y) -> 2\*x^2\*y^3 + 3\*y\*x + x\*sin(y) + y + 1 ;

$$g := (x, y) \rightarrow 2x^2y^3 + 3yx + x \sin(y) + y + 1$$

> D[1](g)(x, y) ; # Correspond à  $\frac{\partial}{\partial x}g(x, y)$

$$4xy^3 + 3y + \sin(y)$$

> D[1, 2](g)(x, y) ; # Correspond à  $\frac{\partial^2}{\partial y \partial x}g(x, y)$

$$12xy^2 + 3 + \cos(y)$$

> D[1\$2, 2](g)(x, y) ; # Correspond à  $\frac{\partial^3}{\partial y \partial x^2}g(x, y)$

$$12y^2$$

Intéressons-nous aux fonctions de plusieurs variables :

> g := (x, y) -> 2\*x^2\*y^3 + 3\*y\*x + x\*sin(y) + y + 1 ;

$$g := (x, y) \rightarrow 2x^2y^3 + 3yx + x \sin(y) + y + 1$$

> D[1](g)(x, y) ; # Correspond à  $\frac{\partial}{\partial x}g(x, y)$

$$4xy^3 + 3y + \sin(y)$$

> D[1,2](g)(x, y) ; # Correspond à  $\frac{\partial^2}{\partial y \partial x}g(x, y)$

$$12xy^2 + 3 + \cos(y)$$

> D[1\$2,2](g)(x, y) ; # Correspond à  $\frac{\partial^3}{\partial y \partial x^2}g(x, y)$

$$12y^2$$

Intéressons-nous aux fonctions de plusieurs variables :

> g := (x, y) -> 2\*x^2\*y^3 + 3\*y\*x + x\*sin(y) + y + 1 ;

$$g := (x, y) \rightarrow 2x^2y^3 + 3yx + x \sin(y) + y + 1$$

> D[1](g)(x, y) ; # Correspond à  $\frac{\partial}{\partial x}g(x, y)$

$$4xy^3 + 3y + \sin(y)$$

> D[1,2](g)(x, y) ; # Correspond à  $\frac{\partial^2}{\partial y \partial x}g(x, y)$

$$12xy^2 + 3 + \cos(y)$$

> D[1\$2,2](g)(x, y) ; # Correspond à  $\frac{\partial^3}{\partial y \partial x^2}g(x, y)$

$$12y^2$$

Intéressons-nous aux fonctions de plusieurs variables :

> g := (x, y) -> 2\*x^2\*y^3 + 3\*y\*x + x\*sin(y) + y + 1 ;

$$g := (x, y) \rightarrow 2x^2y^3 + 3yx + x \sin(y) + y + 1$$

> D[1](g)(x, y) ; # Correspond à  $\frac{\partial}{\partial x}g(x, y)$

$$4xy^3 + 3y + \sin(y)$$

> D[1, 2](g)(x, y) ; # Correspond à  $\frac{\partial^2}{\partial y \partial x}g(x, y)$

$$12xy^2 + 3 + \cos(y)$$

> D[1\$2, 2](g)(x, y) ; # Correspond à  $\frac{\partial^3}{\partial y \partial x^2}g(x, y)$

$$12y^2$$

Intéressons-nous aux fonctions de plusieurs variables :

> g := (x, y) -> 2\*x^2\*y^3 + 3\*y\*x + x\*sin(y) + y + 1 ;

$$g := (x, y) \rightarrow 2x^2y^3 + 3yx + x \sin(y) + y + 1$$

> D[1](g)(x, y) ; # Correspond à  $\frac{\partial}{\partial x}g(x, y)$

$$4xy^3 + 3y + \sin(y)$$

> D[1, 2](g)(x, y) ; # Correspond à  $\frac{\partial^2}{\partial y \partial x}g(x, y)$

$$12xy^2 + 3 + \cos(y)$$

> D[1\$2, 2](g)(x, y) ; # Correspond à  $\frac{\partial^3}{\partial y \partial x^2}g(x, y)$

$$12y^2$$

Intéressons-nous aux fonctions de plusieurs variables :

> g := (x, y) -> 2\*x^2\*y^3 + 3\*y\*x + x\*sin(y) + y + 1 ;

$$g := (x, y) \rightarrow 2x^2y^3 + 3yx + x \sin(y) + y + 1$$

> D[1](g)(x, y) ; # Correspond à  $\frac{\partial}{\partial x}g(x, y)$

$$4xy^3 + 3y + \sin(y)$$

> D[1, 2](g)(x, y) ; # Correspond à  $\frac{\partial^2}{\partial y \partial x}g(x, y)$

$$12xy^2 + 3 + \cos(y)$$

> D[1\$2, 2](g)(x, y) ; # Correspond à  $\frac{\partial^3}{\partial y \partial x^2}g(x, y)$

$$12y^2$$

Enfin si on ne précise pas le point  $(x, y)$  alors on obtient la fonction dérivée :

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

$$(x, y) \rightarrow 12xy^2 + 3 + \cos(y)$$

Enfin si on ne précise pas le point  $(x, y)$  alors on obtient la fonction dérivée :

> D[1, 2] (g) ;

$$(x, y) \rightarrow 12xy^2 + 3 + \cos(y)$$

Enfin si on ne précise pas le point  $(x, y)$  alors on obtient la fonction dérivée :

$> D[1, 2](g) ;$

$$(x, y) \rightarrow 12xy^2 + 3 + \cos(y)$$

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

## 1 Calculs sur les nombres entiers et réels

- Calculs sur les nombres réels
- Calculs sur les entiers
- Les nombres premiers
- Analyse combinatoire
- Sommation
- Produit

## 2 Les nombres complexes

## 3 Fonctions numériques

- Dérivation
- **Limites**
- Intégration
- Développements limités

## 4 Expressions symboliques

- Polynômes
- Fractions rationnelles

On calcule la limite d'une expression  $u$  dépendant d'une variable  $\text{var}$  quand  $\text{var}$  tend vers  $v$  par :

```
> limit(u, var=v, direction);
```

La `direction` est un argument optionnel (voir après).

Exemple :

```
limit(exp(x), x=infinity);
```

$\infty$

Parfois il est nécessaire de spécifier si l'on veut une limite à gauche ou à droite (c'est l'argument `direction`) :

```
> limit(1/x, x=0);
```

*undefined*

Il faut alors spécifier une direction : `right` ou `left`.

```
> limit(1/x, x=0, left);
```

$-\infty$

On calcule la limite d'une expression  $u$  dépendant d'une variable  $var$  quand  $var$  tend vers  $v$  par :

```
> limit(u, var=v, direction);
```

La `direction` est un argument optionnel (voir après).

Exemple :

```
limit(exp(x), x=infinity);
```

$\infty$

Parfois il est nécessaire de spécifier si l'on veut une limite à gauche ou à droite (c'est l'argument `direction`) :

```
> limit(1/x, x=0);
```

*undefined*

Il faut alors spécifier une direction : `right` ou `left`.

```
> limit(1/x, x=0, left);
```

$-\infty$

On calcule la limite d'une expression  $u$  dépendant d'une variable  $var$  quand  $var$  tend vers  $v$  par :

```
> limit(u, var=v, direction);
```

La direction est un argument optionnel (voir après).

Exemple :

```
limit(exp(x), x=infinity);
```

$\infty$

Parfois il est nécessaire de spécifier si l'on veut une limite à gauche ou à droite (c'est l'argument `direction`) :

```
> limit(1/x, x=0);
```

*undefined*

Il faut alors spécifier une direction : `right` ou `left`.

```
> limit(1/x, x=0, left);
```

$-\infty$

On calcule la limite d'une expression  $u$  dépendant d'une variable  $var$  quand  $var$  tend vers  $v$  par :

```
> limit(u, var=v, direction);
```

La **direction** est un argument optionnel (voir après).

Exemple :

```
limit(exp(x), x=infinity);
```

$\infty$

Parfois il est nécessaire de spécifier si l'on veut une limite à gauche ou à droite (c'est l'argument **direction**) :

```
> limit(1/x, x=0);
```

*undefined*

Il faut alors spécifier une direction : **right** ou **left**.

```
> limit(1/x, x=0, left);
```

$-\infty$

On calcule la limite d'une expression  $u$  dépendant d'une variable  $var$  quand  $var$  tend vers  $v$  par :

```
> limit(u, var=v, direction);
```

La `direction` est un argument optionnel (voir après).

Exemple :

```
limit(exp(x), x=infinity);
```

$\infty$

Parfois il est nécessaire de spécifier si l'on veut une limite à gauche ou à droite (c'est l'argument `direction`) :

```
> limit(1/x, x=0);
```

*undefined*

Il faut alors spécifier une direction : `right` ou `left`.

```
> limit(1/x, x=0, left);
```

$-\infty$

On calcule la limite d'une expression  $u$  dépendant d'une variable  $var$  quand  $var$  tend vers  $v$  par :

```
> limit(u, var=v, direction);
```

La `direction` est un argument optionnel (voir après).

Exemple :

```
limit(exp(x), x=infinity);
```

$\infty$

Parfois il est nécessaire de spécifier si l'on veut une limite à gauche ou à droite (c'est l'argument `direction`) :

```
> limit(1/x, x=0);
```

*undefined*

Il faut alors spécifier une direction : `right` ou `left`.

```
> limit(1/x, x=0, left);
```

$-\infty$

On calcule la limite d'une expression  $u$  dépendant d'une variable  $var$  quand  $var$  tend vers  $v$  par :

```
> limit(u, var=v, direction);
```

La `direction` est un argument optionnel (voir après).

Exemple :

```
limit(exp(x), x=infinity);
```

$\infty$

Parfois il est nécessaire de spécifier si l'on veut une limite à gauche ou à droite (c'est l'argument `direction`) :

```
> limit(1/x, x=0);
```

*undefined*

Il faut alors spécifier une direction : `right` ou `left`.

```
> limit(1/x, x=0, left);
```

$-\infty$

On calcule la limite d'une expression  $u$  dépendant d'une variable  $var$  quand  $var$  tend vers  $v$  par :

```
> limit(u, var=v, direction);
```

La `direction` est un argument optionnel (voir après).

Exemple :

```
limit(exp(x), x=infinity);
```

$\infty$

Parfois il est nécessaire de spécifier si l'on veut une limite à gauche ou à droite (c'est l'argument `direction`) :

```
> limit(1/x, x=0);
```

*undefined*

Il faut alors spécifier une direction : `right` ou `left`.

```
> limit(1/x, x=0, left);
```

$-\infty$

On calcule la limite d'une expression  $u$  dépendant d'une variable  $var$  quand  $var$  tend vers  $v$  par :

```
> limit(u, var=v, direction);
```

La `direction` est un argument optionnel (voir après).

Exemple :

```
limit(exp(x), x=infinity);
```

$\infty$

Parfois il est nécessaire de spécifier si l'on veut une limite à gauche ou à droite (c'est l'argument `direction`) :

```
> limit(1/x, x=0);
```

*undefined*

Il faut alors spécifier une direction : `right` ou `left`.

```
> limit(1/x, x=0, left);
```

$-\infty$

On calcule la limite d'une expression  $u$  dépendant d'une variable  $var$  quand  $var$  tend vers  $v$  par :

```
> limit(u, var=v, direction);
```

La `direction` est un argument optionnel (voir après).

Exemple :

```
limit(exp(x), x=infinity);
```

$\infty$

Parfois il est nécessaire de spécifier si l'on veut une limite à gauche ou à droite (c'est l'argument `direction`) :

```
> limit(1/x, x=0);
```

*undefined*

Il faut alors spécifier une direction : `right` ou `left`.

```
> limit(1/x, x=0, left);
```

$-\infty$

On calcule la limite d'une expression  $u$  dépendant d'une variable  $var$  quand  $var$  tend vers  $v$  par :

```
> limit(u, var=v, direction) ;
```

La `direction` est un argument optionnel (voir après).

Exemple :

```
limit(exp(x), x=infinity) ;
```

$\infty$

Parfois il est nécessaire de spécifier si l'on veut une limite à gauche ou à droite (c'est l'argument `direction`) :

```
> limit(1/x, x=0) ;
```

*undefined*

Il faut alors spécifier une direction : `right` ou `left`.

```
> limit(1/x, x=0, left) ;
```

$-\infty$

On calcule la limite d'une expression  $u$  dépendant d'une variable  $var$  quand  $var$  tend vers  $v$  par :

```
> limit(u, var=v, direction) ;
```

La `direction` est un argument optionnel (voir après).

Exemple :

```
limit(exp(x), x=infinity) ;
```

$\infty$

Parfois il est nécessaire de spécifier si l'on veut une limite à gauche ou à droite (c'est l'argument `direction`) :

```
> limit(1/x, x=0) ;
```

*undefined*

Il faut alors spécifier une direction : `right` ou `left`.

```
> limit(1/x, x=0, left) ;
```

$-\infty$

On obtient la forme inerte en écrivant `Limit`. Par exemple,

```
> Limit(ln(x)/x, x=infinity) =  
limit(ln(x)/x, x=infinity);
```

$$\lim_{x \rightarrow \infty} \frac{\ln(x)}{x} = 0$$

On obtient la forme inerte en écrivant `Limit`. Par exemple,

```
> Limit(ln(x)/x, x=infinity) =  
limit(ln(x)/x, x=infinity);
```

$$\lim_{x \rightarrow \infty} \frac{\ln(x)}{x} = 0$$

On obtient la forme inerte en écrivant `Limit`. Par exemple,

```
> Limit(ln(x)/x, x=infinity) =  
limit(ln(x)/x, x=infinity) ;
```

$$\lim_{x \rightarrow \infty} \frac{\ln(x)}{x} = 0$$

On obtient la forme inerte en écrivant `Limit`. Par exemple,

```
> Limit(ln(x)/x, x=infinity) =  
limit(ln(x)/x, x=infinity) ;
```

$$\lim_{x \rightarrow \infty} \frac{\ln(x)}{x} = 0$$

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

- 1 Calculs sur les nombres entiers et réels
  - Calculs sur les nombres réels
  - Calculs sur les entiers
  - Les nombres premiers
  - Analyse combinatoire
  - Sommation
  - Produit
- 2 Les nombres complexes
- 3 Fonctions numériques**
  - Dérivation
  - Limites
  - Intégration**
  - Développements limités
- 4 Expressions symboliques
  - Polynômes
  - Fractions rationnelles

Pour intégrer une expression, on utilise la fonction `int`.

Cette même fonction va également permettre de trouver une primitive de cette expression lorsque l'on n'indiquera pas de bornes d'intégration. Il y a donc deux syntaxes possibles pour `int` :

Pour trouver une primitive d'une expression dépendant de la variable `var` :

```
int (expression, var) ;
```

Pour intégrer une expression dépendant de la variable `var` entre `a` et `b` :

```
int (expression, var=a..b) ;
```

Par exemple pour calculer une primitive :

```
> int (2*x^2+3*cos(x)+ln(x/(x+1)), x) ;
```

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

Rem. : Maple ne rajoute pas de constante d'intégration.

Pour intégrer une expression, on utilise la fonction `int`. Cette même fonction va également permettre de trouver une primitive de cette expression lorsque l'on n'indiquera pas de bornes d'intégration. Il y a donc deux syntaxes

possibles pour `int` :

Pour trouver une primitive d'une expression dépendant de la variable `var` :

```
int (expression, var) ;
```

Pour intégrer une expression dépendant de la variable `var` entre `a` et `b` :

```
int (expression, var=a..b) ;
```

Par exemple pour calculer une primitive :

```
> int (2*x^2+3*cos(x)+ln(x/(x+1)), x) ;
```

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

Rem. : Maple ne rajoute pas de constante d'intégration

Pour intégrer une expression, on utilise la fonction `int`. Cette même fonction va également permettre de trouver une primitive de cette expression lorsque l'on n'indiquera pas de bornes d'intégration. Il y a donc deux syntaxes possibles pour `int` :

Pour trouver une primitive d'une expression dépendant de la variable `var` :

```
int (expression, var) ;
```

Pour intégrer une expression dépendant de la variable `var` entre `a` et `b` :

```
int (expression, var=a..b) ;
```

Par exemple pour calculer une primitive :

```
> int (2*x^2+3*cos(x)+ln(x/(x+1)), x) ;
```

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

Rem. : Maple ne rajoute pas de constante d'intégration

Pour intégrer une expression, on utilise la fonction `int`. Cette même fonction va également permettre de trouver une primitive de cette expression lorsque l'on n'indiquera pas de bornes d'intégration. Il y a donc deux syntaxes possibles pour `int` :

Pour trouver une primitive d'une expression dépendant de la variable `var` :

```
int (expression, var) ;
```

Pour intégrer une expression dépendant de la variable `var` entre `a` et `b` :

```
int (expression, var=a..b) ;
```

Par exemple pour calculer une primitive :

```
> int (2*x^2+3*cos(x)+ln(x/(x+1)), x) ;
```

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

Rem. : Maple ne rajoute pas de constante d'intégration

Pour intégrer une expression, on utilise la fonction `int`. Cette même fonction va également permettre de trouver une primitive de cette expression lorsque l'on n'indiquera pas de bornes d'intégration. Il y a donc deux syntaxes possibles pour `int` :

Pour trouver une primitive d'une expression dépendant de la variable `var` :

```
int (expression, var) ;
```

Pour intégrer une expression dépendant de la variable `var` entre `a` et `b` :

```
int (expression, var=a..b) ;
```

Par exemple pour calculer une primitive :

```
> int (2*x^2+3*cos(x)+ln(x/(x+1)), x) ;
```

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

Rem. : Maple ne rajoute pas de constante d'intégration

Pour intégrer une expression, on utilise la fonction `int`. Cette même fonction va également permettre de trouver une primitive de cette expression lorsque l'on n'indiquera pas de bornes d'intégration. Il y a donc deux syntaxes possibles pour `int` :

Pour trouver une primitive d'une expression dépendant de la variable `var` :

```
int (expression, var) ;
```

Pour intégrer une expression dépendant de la variable `var` entre `a` et `b` :

```
int (expression, var=a..b) ;
```

Par exemple pour calculer une primitive :

```
> int (2*x^2+3*cos(x)+ln(x/(x+1)), x) ;
```

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

Rem. : Maple ne rajoute pas de constante d'intégration

Pour intégrer une expression, on utilise la fonction `int`. Cette même fonction va également permettre de trouver une primitive de cette expression lorsque l'on n'indiquera pas de bornes d'intégration. Il y a donc deux syntaxes possibles pour `int` :

Pour trouver une primitive d'une expression dépendant de la variable `var` :

```
int (expression, var) ;
```

Pour intégrer une expression dépendant de la variable `var` entre `a` et `b` :

```
int (expression, var=a..b) ;
```

Par exemple pour calculer une primitive :

```
> int (2*x^2+3*cos(x)+ln(x/(x+1)), x) ;
```

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

Rem. : Maple ne rajoute pas de constante d'intégration

Pour intégrer une expression, on utilise la fonction `int`. Cette même fonction va également permettre de trouver une primitive de cette expression lorsque l'on n'indiquera pas de bornes d'intégration. Il y a donc deux syntaxes possibles pour `int` :

Pour trouver une primitive d'une expression dépendant de la variable `var` :

```
int (expression, var) ;
```

Pour intégrer une expression dépendant de la variable `var` entre `a` et `b` :

```
int (expression, var=a..b) ;
```

Par exemple pour calculer une primitive :

```
> int (2*x^2+3*cos(x)+ln(x/(x+1)), x) ;
```

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

Rem. : Maple ne rajoute pas de constante d'intégration

Pour intégrer une expression, on utilise la fonction `int`. Cette même fonction va également permettre de trouver une primitive de cette expression lorsque l'on n'indiquera pas de bornes d'intégration. Il y a donc deux syntaxes possibles pour `int` :

Pour trouver une primitive d'une expression dépendant de la variable `var` :

```
int (expression, var) ;
```

Pour intégrer une expression dépendant de la variable `var` entre `a` et `b` :

```
int (expression, var=a..b) ;
```

Par exemple pour calculer une primitive :

```
> int (2*x^2+3*cos(x)+ln(x/(x+1)), x) ;
```

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

Rem. : Maple ne rajoute pas de constante d'intégration.

Pour intégrer une expression, on utilise la fonction `int`. Cette même fonction va également permettre de trouver une primitive de cette expression lorsque l'on n'indiquera pas de bornes d'intégration. Il y a donc deux syntaxes possibles pour `int` :

Pour trouver une primitive d'une expression dépendant de la variable `var` :

```
int (expression, var) ;
```

Pour intégrer une expression dépendant de la variable `var` entre `a` et `b` :

```
int (expression, var=a..b) ;
```

Par exemple pour calculer une primitive :

```
> int (2*x^2+3*cos(x)+ln(x/(x+1)), x) ;
```

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

Rem. : Maple ne rajoute pas de constante d'intégration

Pour intégrer une expression, on utilise la fonction `int`. Cette même fonction va également permettre de trouver une primitive de cette expression lorsque l'on n'indiquera pas de bornes d'intégration. Il y a donc deux syntaxes possibles pour `int` :

Pour trouver une primitive d'une expression dépendant de la variable `var` :

```
int (expression, var) ;
```

Pour intégrer une expression dépendant de la variable `var` entre `a` et `b` :

```
int (expression, var=a..b) ;
```

Par exemple pour calculer une primitive :

```
> int (2*x^2+3*cos(x)+ln(x/(x+1)), x) ;
```

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

Rem. : Maple ne rajoute pas de constante d'intégration.

On peut aussi intégrer entre deux bornes  $a$  et  $b$  :

```
> int(cos(x)+sin(ln(x)), x=0..10);
```

$$\sin(10) + 5 \sin(\ln(2) + \ln(5)) - 5 \cos(\ln(2) + \ln(5))$$

On obtient alors sa valeur approchée à l'aide de `evalf` :

```
> evalf(%);
```

6.516888125

On peut aussi intégrer entre deux bornes  $a$  et  $b$  :

```
> int (cos (x)+sin (ln (x)) , x=0..10) ;
```

$$\sin(10) + 5 \sin(\ln(2) + \ln(5)) - 5 \cos(\ln(2) + \ln(5))$$

On obtient alors sa valeur approchée à l'aide de `evalf` :

```
> evalf(%);
```

6.516888125

On peut aussi intégrer entre deux bornes  $a$  et  $b$  :

```
> int (cos (x) + sin (ln (x) ) , x=0..10) ;
```

$$\sin(10) + 5 \sin(\ln(2) + \ln(5)) - 5 \cos(\ln(2) + \ln(5))$$

On obtient alors sa valeur approchée à l'aide de `evalf` :

```
> evalf(%);
```

6.516888125

On peut aussi intégrer entre deux bornes  $a$  et  $b$  :

```
> int (cos (x) + sin (ln (x) ) , x=0..10) ;
```

$$\sin(10) + 5 \sin(\ln(2) + \ln(5)) - 5 \cos(\ln(2) + \ln(5))$$

On obtient alors sa valeur approchée à l'aide de `evalf` :

```
> evalf(%);
```

6.516888125

On peut aussi intégrer entre deux bornes  $a$  et  $b$  :

```
> int (cos (x)+sin (ln (x)) , x=0..10) ;
```

$$\sin(10) + 5 \sin(\ln(2) + \ln(5)) - 5 \cos(\ln(2) + \ln(5))$$

On obtient alors sa valeur approchée à l'aide de `evalf` :

```
> evalf(%);
```

6.516888125

On peut aussi intégrer entre deux bornes  $a$  et  $b$  :

```
> int (cos (x) + sin (ln (x) ) , x=0..10) ;
```

$$\sin(10) + 5 \sin(\ln(2) + \ln(5)) - 5 \cos(\ln(2) + \ln(5))$$

On obtient alors sa valeur approchée à l'aide de `evalf` :

```
> evalf(%);
```

6.516888125

Maple peut aussi calculer des intégrales généralisées :

```
> int((sin(x)/x)^2, x=0..infinity);
```

$$\frac{1}{2}\pi$$

Maple permet d'afficher la forme inerte de l'intégrale si on utilise la fonction `Int` :

```
> Int(ln(x)/x, x);
```

$$\int \frac{\ln(x)}{x} dx$$

On peut alors très simplement obtenir la valeur de cette intégrale en utilisant la fonction `value` :

```
> value(%);
```

$$\frac{1}{2} \ln(x)^2$$

Maple peut aussi calculer des intégrales généralisées :

```
> int((sin(x)/x)^2, x=0..infinity);
```

$$\frac{1}{2}\pi$$

Maple permet d'afficher la forme inerte de l'intégrale si on utilise la fonction `Int` :

```
> Int(ln(x)/x, x);
```

$$\int \frac{\ln(x)}{x} dx$$

On peut alors très simplement obtenir la valeur de cette intégrale en utilisant la fonction `value` :

```
> value(%);
```

$$\frac{1}{2} \ln(x)^2$$

Maple peut aussi calculer des intégrales généralisées :

```
> int((sin(x)/x)^2, x=0..infinity);
```

$$\frac{1}{2}\pi$$

Maple permet d'afficher la forme inerte de l'intégrale si on utilise la fonction `Int` :

```
> Int(ln(x)/x, x);
```

$$\int \frac{\ln(x)}{x} dx$$

On peut alors très simplement obtenir la valeur de cette intégrale en utilisant la fonction `value` :

```
> value(%);
```

$$\frac{1}{2} \ln(x)^2$$

Maple peut aussi calculer des intégrales généralisées :

```
> int((sin(x)/x)^2, x=0..infinity);
```

$$\frac{1}{2}\pi$$

Maple permet d'afficher la forme inerte de l'intégrale si on utilise la fonction `Int` :

```
> Int(ln(x)/x, x);
```

$$\int \frac{\ln(x)}{x} dx$$

On peut alors très simplement obtenir la valeur de cette intégrale en utilisant la fonction `value` :

```
> value(%);
```

$$\frac{1}{2} \ln(x)^2$$

Maple peut aussi calculer des intégrales généralisées :

```
> int((sin(x)/x)^2, x=0..infinity);
```

$$\frac{1}{2}\pi$$

Maple permet d'afficher la forme inerte de l'intégrale si on utilise la fonction `Int` :

```
> Int(ln(x)/x, x);
```

$$\int \frac{\ln(x)}{x} dx$$

On peut alors très simplement obtenir la valeur de cette intégrale en utilisant la fonction `value` :

```
> value(%);
```

$$\frac{1}{2} \ln(x)^2$$

Maple peut aussi calculer des intégrales généralisées :

```
> int((sin(x)/x)^2, x=0..infinity);
```

$$\frac{1}{2}\pi$$

Maple permet d'afficher la forme inerte de l'intégrale si on utilise la fonction `Int` :

```
> Int(ln(x)/x, x);
```

$$\int \frac{\ln(x)}{x} dx$$

On peut alors très simplement obtenir la valeur de cette intégrale en utilisant la fonction `value` :

```
> value(%);
```

$$\frac{1}{2} \ln(x)^2$$

Maple peut aussi calculer des intégrales généralisées :

```
> int((sin(x)/x)^2, x=0..infinity);
```

$$\frac{1}{2}\pi$$

Maple permet d'afficher la forme inerte de l'intégrale si on utilise la fonction `Int` :

```
> Int(ln(x)/x, x);
```

$$\int \frac{\ln(x)}{x} dx$$

On peut alors très simplement obtenir la valeur de cette intégrale en utilisant la fonction `value` :

```
> value(%);
```

$$\frac{1}{2} \ln(x)^2$$

Maple peut aussi calculer des intégrales généralisées :

```
> int((sin(x)/x)^2, x=0..infinity);
```

$$\frac{1}{2}\pi$$

Maple permet d'afficher la forme inerte de l'intégrale si on utilise la fonction `Int` :

```
> Int(ln(x)/x, x);
```

$$\int \frac{\ln(x)}{x} dx$$

On peut alors très simplement obtenir la valeur de cette intégrale en utilisant la fonction `value` :

```
> value(%);
```

$$\frac{1}{2} \ln(x)^2$$

Maple peut aussi calculer des intégrales généralisées :

```
> int((sin(x)/x)^2, x=0..infinity);
```

$$\frac{1}{2}\pi$$

Maple permet d'afficher la forme inerte de l'intégrale si on utilise la fonction `Int` :

```
> Int(ln(x)/x, x);
```

$$\int \frac{\ln(x)}{x} dx$$

On peut alors très simplement obtenir la valeur de cette intégrale en utilisant la fonction `value` :

```
> value(%);
```

$$\frac{1}{2} \ln(x)^2$$

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

## 1 Calculs sur les nombres entiers et réels

- Calculs sur les nombres réels
- Calculs sur les entiers
- Les nombres premiers
- Analyse combinatoire
- Sommation
- Produit

## 2 Les nombres complexes

## 3 Fonctions numériques

- Dérivation
- Limites
- Intégration
- Développements limités

## 4 Expressions symboliques

- Polynômes
- Fractions rationnelles

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$  ;
- 2  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$  ;
- 3  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$  ;
- 2  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$  ;
- 3  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1**  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$  ;
- 2**  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$  ;
- 3**  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1**  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$  ;
- 2**  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$  ;
- 3**  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1**  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$  ;
- 2**  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$  ;
- 3**  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1**  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$ ;
- 2**  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$ ;
- 3**  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1**  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$  ;
- 2**  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$  ;
- 3**  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1**  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$  ;
- 2**  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$  ;
- 3**  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1**  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$  ;
- 2**  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$  ;
- 3**  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1**  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$  ;
- 2**  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$  ;
- 3**  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1**  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$  ;
- 2**  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$  ;
- 3**  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1**  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$  ;
- 2**  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$  ;
- 3**  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

# Rappels

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

Soient  $a \in \mathbb{R}$  et  $f, g$  deux fonctions numériques.

- 1**  $f = \mathcal{O}(g)$  quand  $x \rightarrow a$  si, et seulement si, il existe  $d > 0$  et  $C > 0$  tel que quel que soit  $x$  avec  $|x - a| < d$ ,  $|f(x)| \leq C|g(x)|$  ;
- 2**  $f = \mathcal{O}(g)$  quand  $x \rightarrow \infty$  si, et seulement si, il existe  $N$  et  $C$  tel que quel que soit  $x > N$ ,  $|f(x)| \leq C|g(x)|$  ;
- 3**  $f = o(g)$  quand  $x \rightarrow a$  si, et seulement si,  $\frac{f(x)}{g(x)} \rightarrow 0$  quand  $x \rightarrow a$ .

La fonction `taylor` permet de réaliser des développements limités. Le DL d'une expression de la variable `var` autour de `point` à l'ordre `ordre` est donné par :

```
> taylor(expression, var=point, ordre);
```

Par défaut, l'ordre est fixé à 6. On peut aussi noter que l'ordre du développement est le degré du premier terme qui est négligeable. Enfin, on signale que Maple emploie des  $\mathcal{O}$  au lieu des traditionnels  $o$ .

Par exemple :

```
> taylor(sin(x), x=0, 8);
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \mathcal{O}(x^8)$$

La fonction `taylor` permet de réaliser des développements limités. Le DL d'une expression de la variable `var` autour de `point` à l'ordre `ordre` est donné par :

```
> taylor(expression, var=point, ordre);
```

Par défaut, l'ordre est fixé à 6. On peut aussi noter que l'ordre du développement est le degré du premier terme qui est négligeable. Enfin, on signale que Maple emploie des  $\mathcal{O}$  au lieu des traditionnels  $o$ .

Par exemple :

```
> taylor(sin(x), x=0, 8);
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \mathcal{O}(x^8)$$

La fonction `taylor` permet de réaliser des développements limités. Le DL d'une expression de la variable `var` autour de `point` à l'ordre `ordre` est donné par :

```
> taylor(expression, var=point, ordre) ;
```

Par défaut, l'ordre est fixé à 6. On peut aussi noter que l'ordre du développement est le degré du premier terme qui est négligeable. Enfin, on signale que Maple emploie des  $\mathcal{O}$  au lieu des traditionnels  $o$ .

Par exemple :

```
> taylor(sin(x), x=0, 8) ;
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \mathcal{O}(x^8)$$

La fonction `taylor` permet de réaliser des développements limités. Le DL d'une expression de la variable `var` autour de `point` à l'ordre `ordre` est donné par :

```
> taylor(expression, var=point, ordre) ;
```

Par défaut, l'ordre est fixé à 6. On peut aussi noter que l'ordre du développement est le degré du premier terme qui est négligeable. Enfin, on signale que Maple emploie des  $\mathcal{O}$  au lieu des traditionnels  $o$ .

Par exemple :

```
> taylor(sin(x), x=0, 8) ;
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \mathcal{O}(x^8)$$

La fonction `taylor` permet de réaliser des développements limités. Le DL d'une expression de la variable `var` autour de `point` à l'ordre `ordre` est donné par :

```
> taylor(expression, var=point, ordre) ;
```

Par défaut, l'ordre est fixé à 6. On peut aussi noter que l'ordre du développement est le degré du premier terme qui est négligeable. Enfin, on signale que Maple emploie des  $\mathcal{O}$  au lieu des traditionnels  $o$ .

Par exemple :

```
> taylor(sin(x), x=0, 8) ;
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \mathcal{O}(x^8)$$

La fonction `taylor` permet de réaliser des développements limités. Le DL d'une expression de la variable `var` autour de `point` à l'ordre `ordre` est donné par :

```
> taylor(expression, var=point, ordre) ;
```

Par défaut, l'ordre est fixé à 6. On peut aussi noter que l'ordre du développement est le degré du premier terme qui est négligeable. Enfin, on signale que Maple emploie des  $\mathcal{O}$  au lieu des traditionnels  $o$ .

Par exemple :

```
> taylor(sin(x), x=0, 8) ;
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \mathcal{O}(x^8)$$

La fonction `taylor` permet de réaliser des développements limités. Le DL d'une expression de la variable `var` autour de `point` à l'ordre `ordre` est donné par :

```
> taylor(expression, var=point, ordre) ;
```

Par défaut, l'ordre est fixé à 6. On peut aussi noter que l'ordre du développement est le degré du premier terme qui est négligeable. Enfin, on signale que Maple emploie des  $\mathcal{O}$  au lieu des traditionnels  $o$ .

Par exemple :

```
> taylor(sin(x), x=0, 8) ;
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \mathcal{O}(x^8)$$

La fonction `taylor` permet de réaliser des développements limités. Le DL d'une expression de la variable `var` autour de `point` à l'ordre `ordre` est donné par :

```
> taylor(expression, var=point, ordre) ;
```

Par défaut, l'ordre est fixé à 6. On peut aussi noter que l'ordre du développement est le degré du premier terme qui est négligeable. Enfin, on signale que Maple emploie des  $\mathcal{O}$  au lieu des traditionnels  $o$ .

Par exemple :

```
> taylor(sin(x), x=0, 8) ;
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \mathcal{O}(x^8)$$

La fonction `taylor` permet de réaliser des développements limités. Le DL d'une expression de la variable `var` autour de `point` à l'ordre `ordre` est donné par :

```
> taylor(expression, var=point, ordre) ;
```

Par défaut, l'ordre est fixé à 6. On peut aussi noter que l'ordre du développement est le degré du premier terme qui est négligeable. Enfin, on signale que Maple emploie des  $\mathcal{O}$  au lieu des traditionnels  $o$ .

Par exemple :

```
> taylor(sin(x), x=0, 8) ;
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + O(x^8)$$

On peut récupérer la partie polynômiale du DL en convertissant ce dernier en un polynôme :

```
> convert(taylor(sin(x), x=0), polynom);
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5$$

On peut évidemment effectuer un DL en un point différent de 0 :

```
> taylor(sin(x), x=Pi, 4);
```

$$-(x - \pi) + \frac{1}{6}(x - \pi)^3 + O((x - \pi)^4)$$

On peut récupérer la partie polynômiale du DL en convertissant ce dernier en un polynôme :

```
> convert(taylor(sin(x), x=0), polynom) ;
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5$$

On peut évidemment effectuer un DL en un point différent de 0 :

```
> taylor(sin(x), x=Pi, 4) ;
```

$$-(x - \pi) + \frac{1}{6}(x - \pi)^3 + O((x - \pi)^4)$$

On peut récupérer la partie polynômiale du DL en convertissant ce dernier en un polynôme :

> convert(taylor(sin(x), x=0), polynom) ;

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5$$

On peut évidemment effectuer un DL en un point différent de 0 :

> taylor(sin(x), x=Pi, 4) ;

$$-(x - \pi) + \frac{1}{6}(x - \pi)^3 + O((x - \pi)^4)$$

On peut récupérer la partie polynômiale du DL en convertissant ce dernier en un polynôme :

```
> convert(taylor(sin(x), x=0), polynom) ;
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5$$

On peut évidemment effectuer un DL en un point différent de 0 :

```
> taylor(sin(x), x=Pi, 4) ;
```

$$-(x - \pi) + \frac{1}{6}(x - \pi)^3 + O((x - \pi)^4)$$

On peut récupérer la partie polynômiale du DL en convertissant ce dernier en un polynôme :

> `convert(taylor(sin(x), x=0), polynom) ;`

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5$$

On peut évidemment effectuer un DL en un point différent de 0 :

> `taylor(sin(x), x=Pi, 4) ;`

$$-(x - \pi) + \frac{1}{6}(x - \pi)^3 + O((x - \pi)^4)$$

On peut récupérer la partie polynômiale du DL en convertissant ce dernier en un polynôme :

> `convert(taylor(sin(x), x=0), polynom) ;`

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5$$

On peut évidemment effectuer un DL en un point différent de 0 :

> `taylor(sin(x), x=Pi, 4) ;`

$$-(x - \pi) + \frac{1}{6}(x - \pi)^3 + O((x - \pi)^4)$$

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

- 1 Calculs sur les nombres entiers et réels
  - Calculs sur les nombres réels
  - Calculs sur les entiers
  - Les nombres premiers
  - Analyse combinatoire
  - Sommation
  - Produit
- 2 Les nombres complexes
- 3 Fonctions numériques
  - Dérivation
  - Limites
  - Intégration
  - Développements limités
- 4 Expressions symboliques
  - Polynômes
  - Fractions rationnelles

On peut définir un polynôme :

```
> A := x^2 + 3*x + 1;
```

$$A := x^2 + 3x + 1$$

```
> B := x - 1;
```

$$B := x - 1$$

On peut tester si  $A$  divise  $B$  par l'intermédiaire de la fonction `divide` :

```
> divide(A, B, x);
```

*false*

On peut définir un polynôme :

```
> A := x^2 + 3 * x + 1 ;
```

$$A := x^2 + 3x + 1$$

```
> B := x - 1 ;
```

$$B := x - 1$$

On peut tester si  $A$  divise  $B$  par l'intermédiaire de la fonction `divide` :

```
> divide(A, B, x) ;
```

*false*

On peut définir un polynôme :

```
> A := x^2 + 3 * x + 1 ;
```

$$A := x^2 + 3x + 1$$

```
> B := x - 1 ;
```

$$B := x - 1$$

On peut tester si  $A$  divise  $B$  par l'intermédiaire de la fonction `divide` :

```
> divide(A, B, x) ;
```

*false*

On peut définir un polynôme :

```
> A := x^2 + 3 * x + 1 ;
```

$$A := x^2 + 3x + 1$$

```
> B := x - 1 ;
```

$$B := x - 1$$

On peut tester si  $A$  divise  $B$  par l'intermédiaire de la fonction `divide` :

```
> divide(A, B, x) ;
```

*false*

On peut définir un polynôme :

```
> A := x^2 + 3 * x + 1 ;
```

$$A := x^2 + 3x + 1$$

```
> B := x - 1 ;
```

$$B := x - 1$$

On peut tester si  $A$  divise  $B$  par l'intermédiaire de la fonction `divide` :

```
> divide(A, B, x) ;
```

*false*

On peut définir un polynôme :

```
> A := x^2 + 3 * x + 1 ;
```

$$A := x^2 + 3x + 1$$

```
> B := x - 1 ;
```

$$B := x - 1$$

On peut tester si  $A$  divise  $B$  par l'intermédiaire de la fonction `divide` :

```
> divide(A, B, x) ;
```

*false*

On peut définir un polynôme :

```
> A := x^2 + 3 * x + 1 ;
```

$$A := x^2 + 3x + 1$$

```
> B := x - 1 ;
```

$$B := x - 1$$

On peut tester si  $A$  divise  $B$  par l'intermédiaire de la fonction `divide` :

```
> divide(A, B, x) ;
```

*false*

On peut définir un polynôme :

```
> A := x^2 + 3 * x + 1 ;
```

$$A := x^2 + 3x + 1$$

```
> B := x - 1 ;
```

$$B := x - 1$$

On peut tester si  $A$  divise  $B$  par l'intermédiaire de la fonction `divide` :

```
> divide(A, B, x) ;
```

*false*

On peut écrire le polynôme  $A$  sous la forme  $A = BQ + R$  à l'aide d'une division euclidienne. On commence par calculer le quotient :

```
> quo (A, B, x) ;
```

$x + 4$

Puis on calcule le reste :

```
> rem (A, B, x) ;
```

5

On peut écrire le polynôme  $A$  sous la forme  $A = BQ + R$  à l'aide d'une division euclidienne. On commence par calculer le quotient :

```
> quo (A, B, x) ;
```

$x + 4$

Puis on calcule le reste :

```
> rem (A, B, x) ;
```

5

On peut écrire le polynôme  $A$  sous la forme  $A = BQ + R$  à l'aide d'une division euclidienne. On commence par calculer le quotient :

> quo (A, B, x) ;

$x + 4$

Puis on calcule le reste :

> rem (A, B, x) ;

5

On peut écrire le polynôme  $A$  sous la forme  $A = BQ + R$  à l'aide d'une division euclidienne. On commence par calculer le quotient :

> quo (A, B, x) ;

$$x + 4$$

Puis on calcule le reste :

> rem (A, B, x) ;

$$5$$

On peut écrire le polynôme  $A$  sous la forme  $A = BQ + R$  à l'aide d'une division euclidienne. On commence par calculer le quotient :

`> quo(A, B, x) ;`

$x + 4$

Puis on calcule le reste :

`> rem(A, B, x) ;`

5

On peut écrire le polynôme  $A$  sous la forme  $A = BQ + R$  à l'aide d'une division euclidienne. On commence par calculer le quotient :

> quo (A, B, x) ;

$$x + 4$$

Puis on calcule le reste :

> rem (A, B, x) ;

$$5$$

On peut écrire le polynôme  $A$  sous la forme  $A = BQ + R$  à l'aide d'une division euclidienne. On commence par calculer le quotient :

> quo (A, B, x) ;

$$x + 4$$

Puis on calcule le reste :

> rem (A, B, x) ;

$$5$$

Bien sûr on peut aussi développer et factoriser les polynômes :

```
> C := (x-3) * (x-2) ;
```

```
> expand(C) ; # Pour développer C
```

$$x^2 - 5x + 6$$

On peut alors retrouver l'expression d'origine de  $C$  à l'aide de `factor` :

```
> factor(%);
```

$$(x - 3)(x - 2)$$

Bien sûr on peut aussi développer et factoriser les polynômes :

```
> C := (x-3) * (x-2) :
```

```
> expand(C) ; # Pour développer C
```

$$x^2 - 5x + 6$$

On peut alors retrouver l'expression d'origine de  $C$  à l'aide de factor :

```
> factor(%);
```

$$(x - 3)(x - 2)$$

Bien sûr on peut aussi développer et factoriser les polynômes :

```
> C := (x-3) * (x-2) :
```

```
> expand(C) ; # Pour développer C
```

$$x^2 - 5x + 6$$

On peut alors retrouver l'expression d'origine de  $C$  à l'aide de `factor` :

```
> factor(%);
```

$$(x - 3)(x - 2)$$

Bien sûr on peut aussi développer et factoriser les polynômes :

```
> C := (x-3) * (x-2) :
```

```
> expand(C) ; # Pour développer C
```

$$x^2 - 5x + 6$$

On peut alors retrouver l'expression d'origine de  $C$  à l'aide de `factor` :

```
> factor(%);
```

$$(x - 3)(x - 2)$$

Bien sûr on peut aussi développer et factoriser les polynômes :

```
> C := (x-3) * (x-2) :
```

```
> expand(C) ; # Pour développer C
```

$$x^2 - 5x + 6$$

On peut alors retrouver l'expression d'origine de  $C$  à l'aide de `factor` :

```
> factor(%);
```

$$(x - 3)(x - 2)$$

Bien sûr on peut aussi développer et factoriser les polynômes :

```
> C := (x-3) * (x-2) :
```

```
> expand(C) ; # Pour développer C
```

$$x^2 - 5x + 6$$

On peut alors retrouver l'expression d'origine de  $C$  à l'aide de `factor` :

```
> factor(%);
```

$$(x - 3)(x - 2)$$

Bien sûr on peut aussi développer et factoriser les polynômes :

```
> C := (x-3) * (x-2) :
```

```
> expand(C) ; # Pour développer C
```

$$x^2 - 5x + 6$$

On peut alors retrouver l'expression d'origine de  $C$  à l'aide de `factor` :

```
> factor(%);
```

$$(x - 3)(x - 2)$$

Dans certains cas, on peut vouloir factoriser dans  $\mathbb{C}$  :

```
> E := x^2 + 2*x + 2 ;
```

$$E := x^2 + 2x + 2$$

```
> factor(E) ;
```

$$x^2 + 2x + 2$$

Par défaut, Maple factorise dans  $\mathbb{R}$ . On peut cependant forcer la factorisation dans  $\mathbb{C}$  en employant l'option `complex` :

```
> factor(E, complex) ;
```

$$(x + 1 + 1.I)(x + 1 - 1.I)$$

Dans certains cas, on peut vouloir factoriser dans  $\mathbb{C}$  :

```
> E := x^2 + 2*x + 2 ;
```

$$E := x^2 + 2x + 2$$

```
> factor(E) ;
```

$$x^2 + 2x + 2$$

Par défaut, Maple factorise dans  $\mathbb{R}$ . On peut cependant forcer la factorisation dans  $\mathbb{C}$  en employant l'option `complex` :

```
> factor(E, complex) ;
```

$$(x + 1 + 1.I)(x + 1 - 1.I)$$

Dans certains cas, on peut vouloir factoriser dans  $\mathbb{C}$  :

>  $E := x^2 + 2x + 2$  ;

$$E := x^2 + 2x + 2$$

> factor(E) ;

$$x^2 + 2x + 2$$

Par défaut, Maple factorise dans  $\mathbb{R}$ . On peut cependant forcer la factorisation dans  $\mathbb{C}$  en employant l'option `complex` :

> factor(E, complex) ;

$$(x + 1 + 1.i)(x + 1 - 1.i)$$

Dans certains cas, on peut vouloir factoriser dans  $\mathbb{C}$  :

```
> E := x^2 + 2 * x + 2 ;
```

$$E := x^2 + 2x + 2$$

```
> factor(E) ;
```

$$x^2 + 2x + 2$$

Par défaut, Maple factorise dans  $\mathbb{R}$ . On peut cependant forcer la factorisation dans  $\mathbb{C}$  en employant l'option `complex` :

```
> factor(E, complex) ;
```

$$(x + 1 + 1.I)(x + 1 - 1.I)$$

Dans certains cas, on peut vouloir factoriser dans  $\mathbb{C}$  :

```
> E := x^2 + 2*x + 2 ;
```

$$E := x^2 + 2x + 2$$

```
> factor(E) ;
```

$$x^2 + 2x + 2$$

Par défaut, Maple factorise dans  $\mathbb{R}$ . On peut cependant forcer la factorisation dans  $\mathbb{C}$  en employant l'option `complex` :

```
> factor(E, complex) ;
```

$$(x + 1 + 1.i)(x + 1 - 1.i)$$

Dans certains cas, on peut vouloir factoriser dans  $\mathbb{C}$  :

```
> E := x^2 + 2*x + 2 ;
```

$$E := x^2 + 2x + 2$$

```
> factor(E) ;
```

$$x^2 + 2x + 2$$

Par défaut, Maple factorise dans  $\mathbb{R}$ . On peut cependant forcer la factorisation dans  $\mathbb{C}$  en employant l'option `complex` :

```
> factor(E, complex) ;
```

$$(x + 1 + 1.i)(x + 1 - 1.i)$$

Dans certains cas, on peut vouloir factoriser dans  $\mathbb{C}$  :

```
> E := x^2 + 2*x + 2 ;
```

$$E := x^2 + 2x + 2$$

```
> factor(E) ;
```

$$x^2 + 2x + 2$$

Par défaut, Maple factorise dans  $\mathbb{R}$ . On peut cependant forcer la factorisation dans  $\mathbb{C}$  en employant l'option `complex` :

```
> factor(E, complex) ;
```

$$(x + 1 + 1.i)(x + 1 - 1.i)$$

Dans certains cas, on peut vouloir factoriser dans  $\mathbb{C}$  :

```
> E := x^2 + 2*x + 2 ;
```

$$E := x^2 + 2x + 2$$

```
> factor(E) ;
```

$$x^2 + 2x + 2$$

Par défaut, Maple factorise dans  $\mathbb{R}$ . On peut cependant forcer la factorisation dans  $\mathbb{C}$  en employant l'option `complex` :

```
> factor(E, complex) ;
```

$$(x + 1 + 1.i)(x + 1 - 1.i)$$

Dans certains cas, on peut vouloir factoriser dans  $\mathbb{C}$  :

```
> E := x^2 + 2*x + 2 ;
```

$$E := x^2 + 2x + 2$$

```
> factor(E) ;
```

$$x^2 + 2x + 2$$

Par défaut, Maple factorise dans  $\mathbb{R}$ . On peut cependant forcer la factorisation dans  $\mathbb{C}$  en employant l'option `complex` :

```
> factor(E, complex) ;
```

$$(x + 1 + 1.I)(x + 1 - 1.I)$$

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de  $H$ 
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de  $H$ 
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de  $H$ 
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de  $H$ 
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de  $H$ 
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de  $H$ 
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de H
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de H
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de H
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de H
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de H
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de H
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de  $H$ 
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de  $H$ 
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de  $H$ 
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de  $H$ 
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de  $H$ 
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de  $H$ 
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de  $H$ 
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de  $H$ 
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de  $H$ 
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de  $H$ 
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de  $H$ 
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de  $H$ 
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de  $H$ 
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de  $H$ 
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de  $H$ 
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de  $H$ 
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de  $H$ 
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de  $H$ 
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de  $H$ 
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de  $H$ 
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de  $H$ 
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de  $H$ 
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de  $H$ 
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de  $H$ 
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de H
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de H
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de H
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de  $H$ 
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de  $H$ 
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de  $H$ 
```

4

## Fonctions de manipulation des coefficients d'un polynôme :

```
>H :=2*x^3+4x :
```

```
>coeff(H,x,3) ; # Le coefficient de degré 3  
de  $H$ 
```

2

```
>coeff(H,x,2) ;
```

0

```
>coeff(H,x,1) ;
```

4

```
>lcoeff(H) ; # Coefficient de plus haut  
degré de  $H$ 
```

2

```
>tcoeff(H) ; # Coefficient de plus bas degré  
de  $H$ 
```

4

Enfin on obtient le degré d'un polynôme par

```
>degree (H) ;
```

3

Enfin on obtient le degré d'un polynôme par  
>degree (H) ;

3

Enfin on obtient le degré d'un polynôme par  
>degree (H) ;

3

# Plan

Chap. II :  
Analyse avec  
Maple

Laurent  
Poinsot

Calculs sur  
les nombres  
entiers et  
réels

Les nombres  
complexes

Fonctions  
numériques

Expressions  
symboliques

## 1 Calculs sur les nombres entiers et réels

- Calculs sur les nombres réels
- Calculs sur les entiers
- Les nombres premiers
- Analyse combinatoire
- Sommation
- Produit

## 2 Les nombres complexes

## 3 Fonctions numériques

- Dérivation
- Limites
- Intégration
- Développements limités

## 4 Expressions symboliques

- Polynômes
- Fractions rationnelles

Voici un exemple de fraction rationnelle :

```
> F := (3*x^4+4*x^2+2*x+1) / (x^3+2*x+1) ;
```

$$F := \frac{3x^4 + 4x^2 + 2x + 1}{x^3 + 2x + 1}$$

```
>denom(F) ;
```

$$x^3 + 2x + 1$$

```
>numer(F) ;
```

$$3x^4 + 4x^3 + 2x + 1$$

Voici un exemple de fraction rationnelle :

> F := (3\*x^4+4\*x^2+2\*x+1) / (x^3+2\*x+1) ;

$$F := \frac{3x^4 + 4x^2 + 2x + 1}{x^3 + 2x + 1}$$

>denom(F) ;

$$x^3 + 2x + 1$$

>numer(F) ;

$$3x^4 + 4x^3 + 2x + 1$$

Voici un exemple de fraction rationnelle :

> F := (3\*x^4+4\*x^2+2\*x+1) / (x^3+2\*x+1) ;

$$F := \frac{3x^4 + 4x^2 + 2x + 1}{x^3 + 2x + 1}$$

>denom(F) ;

$$x^3 + 2x + 1$$

>numer(F) ;

$$3x^4 + 4x^3 + 2x + 1$$

Voici un exemple de fraction rationnelle :

> F := (3\*x^4+4\*x^2+2\*x+1) / (x^3+2\*x+1) ;

$$F := \frac{3x^4 + 4x^2 + 2x + 1}{x^3 + 2x + 1}$$

>denom(F) ;

$$x^3 + 2x + 1$$

>numer(F) ;

$$3x^4 + 4x^3 + 2x + 1$$

Voici un exemple de fraction rationnelle :

> F := (3\*x^4+4\*x^2+2\*x+1) / (x^3+2\*x+1) ;

$$F := \frac{3x^4 + 4x^2 + 2x + 1}{x^3 + 2x + 1}$$

>denom(F) ;

$$x^3 + 2x + 1$$

>numer(F) ;

$$3x^4 + 4x^3 + 2x + 1$$

Voici un exemple de fraction rationnelle :

> F := (3\*x^4+4\*x^2+2\*x+1) / (x^3+2\*x+1) ;

$$F := \frac{3x^4 + 4x^2 + 2x + 1}{x^3 + 2x + 1}$$

>denom(F) ;

$$x^3 + 2x + 1$$

>numer(F) ;

$$3x^4 + 4x^3 + 2x + 1$$

Voici un exemple de fraction rationnelle :

> F := (3\*x^4+4\*x^2+2\*x+1) / (x^3+2\*x+1) ;

$$F := \frac{3x^4 + 4x^2 + 2x + 1}{x^3 + 2x + 1}$$

>denom(F) ;

$$x^3 + 2x + 1$$

>numer(F) ;

$$3x^4 + 4x^3 + 2x + 1$$

On peut décomposer une fraction rationnelle en éléments simples :

```
> G := (x^2-x-6) / (x^2+1) ;
```

$$G := \frac{x^2 - x - 6}{x^2 + 1}$$

```
> convert(G, parfrac, x) ;
```

$$1 + \frac{x - 7}{x^2 + 1}$$

On peut décomposer une fraction rationnelle en éléments simples :

```
> G := (x^2-x-6) / (x^2+1) ;
```

$$G := \frac{x^2 - x - 6}{x^2 + 1}$$

```
> convert(G, parfrac, x) ;
```

$$1 + \frac{x - 7}{x^2 + 1}$$

On peut décomposer une fraction rationnelle en éléments simples :

> G := (x<sup>2</sup>-x-6) / (x<sup>2</sup>+1) ;

$$G := \frac{x^2 - x - 6}{x^2 + 1}$$

> convert(G, parfrac, x) ;

$$1 + \frac{x - 7}{x^2 + 1}$$

On peut décomposer une fraction rationnelle en éléments simples :

> G := (x<sup>2</sup>-x-6) / (x<sup>2</sup>+1) ;

$$G := \frac{x^2 - x - 6}{x^2 + 1}$$

> convert(G, parfrac, x) ;

$$1 + \frac{x - 7}{x^2 + 1}$$

On peut décomposer une fraction rationnelle en éléments simples :

```
> G := (x^2-x-6) / (x^2+1) ;
```

$$G := \frac{x^2 - x - 6}{x^2 + 1}$$

```
> convert(G, parfrac, x) ;
```

$$1 + \frac{x - 7}{x^2 + 1}$$