



## Leçon 5 - Calculs et opérations avec $\sum$ (ou $\prod$ )

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Leçon 5 - Calculs et opérations avec $\sum$ (ou $\prod$ )

⇒ **Savoir manipuler des sommes. Résultats exacts**

⇒ **Sommes doubles**

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples. . .)

2.6. Exercices d'application

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples. . .)

2.6. Exercices d'application

# Sommation par paquets. Relation de Chasles

On a alors la relation de Chasles pour les sommes :

## Proposition - Sommation par paquets

Soit une famille  $(E_r)_{r \in S}$  une famille d'ensembles indexés par  $S$ .

On suppose qu'il s'agit d'une famille d'ensembles 2 à 2 disjoints :

$$\forall r \neq r' \in S, E_r \cap E_{r'} = \emptyset.$$

Alors :

$$\sum_{r \in S} \left( \sum_{k \in E_r} a_k \right) = \sum_{k \in \bigsqcup_{r \in S} E_r} a_k$$

On voit apparaître ici une double somme. On en reparlera plus loin.

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Sommation par paquets. Savoir-faire

⇒ Savoir manipuler  
des sommes.

Résultats exacts

⇒ Sommes doubles

**Remarque** Interversion des symboles  $\Sigma$ .

1. Quelques  
problèmes

2. Symboles  $\Sigma$  et  $\Pi$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

**Remarque** Interversion des symboles  $\sum$ .

## Savoir-faire. Exploiter une sommation par paquets

On a parfois intérêt à découper l'ensemble  $E$  en (réunion de)  $m$  sous-ensembles disjoints  $E = E_1 \uplus E_2 \cdots \uplus E_m$ .

On calcule alors la somme par paquets : 
$$\sum_{k \in E} a_k = \sum_{i=1}^m \left( \sum_{k \in E_i} a_k \right).$$

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Savoir-faire. Méthode du télescopage (ou dominos)

Soit  $(u_n)$  une suite. Soient  $p, n \in \mathbb{N}$  tels que  $p \leq n$

$$\text{Alors } \sum_{k=p}^n (u_{k+1} - u_k) = u_{n+1} - u_p$$

$$\left( = (u_{p+1} - u_p) + (u_{p+2} - u_{p+1}) + \cdots + (u_{n+1} - u_n) \right)$$

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

## Savoir-faire. Méthode du télescopage (ou dominos)

Soit  $(u_n)$  une suite. Soient  $p, n \in \mathbb{N}$  tels que  $p \leq n$

$$\text{Alors } \sum_{k=p}^n (u_{k+1} - u_k) = u_{n+1} - u_p$$
$$\left( = (u_{p+1} - u_p) + (u_{p+2} - u_{p+1}) + \cdots + (u_{n+1} - u_n) \right)$$

**Remarque** « Voir » le télescopage

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Savoir-faire. Méthode du télescopage (ou dominos)

Soit  $(u_n)$  une suite. Soient  $p, n \in \mathbb{N}$  tels que  $p \leq n$

$$\text{Alors } \sum_{k=p}^n (u_{k+1} - u_k) = u_{n+1} - u_p$$

$$\left( = (u_{p+1} - u_p) + (u_{p+2} - u_{p+1}) + \cdots + (u_{n+1} - u_n) \right)$$

**Remarque** « Voir » le télescopage

### Exercice

$$\text{Calculer } \sum_{k=1}^n \ln \frac{k+1}{k}.$$

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

⇒ **Savoir manipuler des sommes. Résultats exacts**

⇒ **Sommes doubles**

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

Pour calculer  $S = \sum_{i \in I} a_i$ , la méthode est simple.

On exploite un raisonnement (calcul) par récurrence :

- ▶ avec une boucle `for` si  $I$  est bien connu (par exemple  $\mathbb{N}_n$ )
- ▶ avec une boucle `while` si  $I$  se découvre au fur et à mesure du calcul.

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Somme avec Python

$$\text{Python } \sum_{k=n}^m a(k)$$

```
1 def Somme1(n,m):
2     S=0
3     for k in range(n,m+1):
4         S=S+a(k)
5     return(S)
```

⇒ Savoir manipuler  
des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques  
problèmes

2. Symboles  $\Sigma$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

# Somme avec Python

$$\text{Python } \sum_{k=n}^m a(k)$$

```
1 def Somme1(n,m):
2     S=0
3     for k in range(n,m+1):
4         S=S+a(k)
5     return(S)
```

$$\text{Python } \sum_{i \mid f(i) \leq n} a(i)$$

```
1 def Somme2(n):
2     S,i=0,0
3     while f(i)<=n:
4         S=S+a(i)
5         i=i+1
```

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Faire un programme

**Remarque** L'ordinateur (calculatrice) comme un obstacle ?

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\Sigma$  et  $\Pi$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Faire un programme

**Remarque** L'ordinateur (calculatrice) comme un obstacle ?

## Truc & Astuce pour le calcul. Écrire un programme

Écrire un programme permet souvent de mieux comprendre la nature du calcul. C'est le cas en particulier :

- ▶ pour le calcul de somme.
- ▶ pour le calcul de probabilités.

Dans ce cas, ce n'est pas le calcul mais la modélisation elle-même du problème qui est mieux comprise.

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Faire un programme

**Remarque** L'ordinateur (calculatrice) comme un obstacle ?

## Truc & Astuce pour le calcul. Écrire un programme

Ecrire un programme permet souvent de mieux comprendre la nature du calcul. C'est le cas en particulier :

- ▶ pour le calcul de somme.
  - ▶ pour le calcul de probabilités.
- Dans ce cas, ce n'est pas le calcul mais la modélisation elle-même du problème qui est mieux comprise.

### Exercice

Ecrire une boucle (double ?) pour faire le calcul :

$$\sum_{i=1}^{100} \sum_{j=i}^{200-i} i \times j$$

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Faire un programme

**Remarque** L'ordinateur (calculatrice) comme un obstacle ?

### Truc & Astuce pour le calcul. Écrire un programme

Ecrire un programme permet souvent de mieux comprendre la nature du calcul. C'est le cas en particulier :

- ▶ pour le calcul de somme.
  - ▶ pour le calcul de probabilités.
- Dans ce cas, ce n'est pas le calcul mais la modélisation elle-même du problème qui est mieux comprise.

### Exercice

Ecrire une boucle (double ?) pour faire le calcul :

$$\sum_{i=1}^{100} \sum_{j=i}^{200-i} i \times j$$

**Remarque** Varier les paramètres et informatique

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

⇒ **Savoir manipuler des sommes. Résultats exacts**

⇒ **Sommes doubles**

1. Quelques problèmes

2. Symboles  $\Sigma$  et  $\Pi$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

**2.4. Des sommes connues**

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

⇒ Savoir manipuler des sommes. Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\Sigma$  et  $\Pi$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

**2.4. Des sommes connues**

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Trois sommes à connaître

## Proposition - Sommes de puissances d'entiers consécutifs

Soit  $n \in \mathbb{N}^*$ . Alors :

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}; \quad \sum_{k=1}^n k^2 = \frac{n(n+1)(2n+1)}{6}; \quad \sum_{k=1}^n k^3 = \left(\frac{n(n+1)}{2}\right)^2$$

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Trois sommes à connaître

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

## Proposition - Sommes de puissances d'entiers consécutifs

Soit  $n \in \mathbb{N}^*$ . Alors :

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}; \quad \sum_{k=1}^n k^2 = \frac{n(n+1)(2n+1)}{6}; \quad \sum_{k=1}^n k^3 = \left(\frac{n(n+1)}{2}\right)^2$$

On peut faire une récurrence, ou bien chercher à utiliser le télescopage.

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

# Trois sommes à connaître

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

## Proposition - Sommes de puissances d'entiers consécutifs

Soit  $n \in \mathbb{N}^*$ . Alors :

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}; \quad \sum_{k=1}^n k^2 = \frac{n(n+1)(2n+1)}{6}; \quad \sum_{k=1}^n k^3 = \left(\frac{n(n+1)}{2}\right)^2$$

On peut faire une récurrence, ou bien chercher à utiliser le télescopage.

### Démonstration

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Trois sommes à connaître

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

## Proposition - Sommes de puissances d'entiers consécutifs

Soit  $n \in \mathbb{N}^*$ . Alors :

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}; \quad \sum_{k=1}^n k^2 = \frac{n(n+1)(2n+1)}{6}; \quad \sum_{k=1}^n k^3 = \left(\frac{n(n+1)}{2}\right)^2$$

On peut faire une récurrence, ou bien chercher à utiliser le télescopage.

### Démonstration

Exercice Faire la récurrence

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

**Savoir-faire.** Se passer du formalisme d'une récurrence ou invariant de boucle

On peut souvent se passer de la formalisation de la récurrence (mais avec les mêmes calculs).

Ici, on considère la suite  $(u_n) = \sum_{k=1}^n k^3 - \left(\frac{n(n+1)}{2}\right)^2$ .

On note que (calculs) :  $u_{n+1} = u_n$  (pour tout  $n \in \mathbb{N}$ ) et  $u_1 = 0$ .  
Donc pour tout  $n \in \mathbb{N}$ ,  $u_n = 0$ . CQFD.

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Proposition - Calcul d'une somme arithmétique

Pour une suite arithmétique :

$$\forall n \in \mathbb{N}, u_{n+1} = u_n + r$$

on a :

$$\sum_{k=n}^m u_k = (m - n + 1) \times \frac{u_m + u_n}{2}$$

C'est-à-dire :

somme de trm successifs = (nb de trm)  $\times$  (moyenne des trm extrêmes)

$\Rightarrow$  Savoir manipuler des sommes.  
Résultats exacts

$\Rightarrow$  Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Proposition - Calcul d'une somme arithmétique

Pour une suite arithmétique :

$$\forall n \in \mathbb{N}, u_{n+1} = u_n + r$$

on a :

$$\sum_{k=n}^m u_k = (m - n + 1) \times \frac{u_m + u_n}{2}$$

C'est-à-dire :

somme de trm successifs = (nb de trm)  $\times$  (moyenne des trm extrêmes)

## Démonstration

$\Rightarrow$  Savoir manipuler des sommes.  
Résultats exacts

$\Rightarrow$  Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Somme arithmétique

## Proposition - Calcul d'une somme arithmétique

Pour une suite arithmétique :

$$\forall n \in \mathbb{N}, u_{n+1} = u_n + r$$

on a :

$$\sum_{k=n}^m u_k = (m - n + 1) \times \frac{u_m + u_n}{2}$$

C'est-à-dire :

somme de trm successifs = (nb de trm)  $\times$  (moyenne des trm extrêmes)

## Démonstration

### Exercice

Et avec la méthode de Gauss (double somme inversée)

=> Savoir manipuler des sommes.

Résultats exacts

=> Sommes doubles

1. Quelques problèmes

2. Symboles  $\Sigma$  et  $\Pi$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Proposition - Calcul d'une somme géométrique

Soit  $x \in \mathbb{R}$  (ou  $x \in \mathbb{C}$ ) et  $n \in \mathbb{N}$ . On a alors :

$$\sum_{k=0}^n x^k = \begin{cases} n + 1 & \text{si } x = 1 \\ \frac{1 - x^{n+1}}{1 - x} & \text{si } x \neq 1 \end{cases}$$

Plus généralement pour une suite géométrique de raison  $q \neq 1$  :

$$\text{somme de termes successifs} = \text{1er terme} \times \frac{1 - q^{\text{nb de termes}}}{1 - q}$$

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Proposition - Calcul d'une somme géométrique

Soit  $x \in \mathbb{R}$  (ou  $x \in \mathbb{C}$ ) et  $n \in \mathbb{N}$ . On a alors :

$$\sum_{k=0}^n x^k = \begin{cases} n + 1 & \text{si } x = 1 \\ \frac{1 - x^{n+1}}{1 - x} & \text{si } x \neq 1 \end{cases}$$

Plus généralement pour une suite géométrique de raison  $q \neq 1$  :

$$\text{somme de termes successifs} = \text{1er terme} \times \frac{1 - q^{\text{nb de termes}}}{1 - q}$$

## Démonstration

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Proposition - Calcul d'une somme géométrique

Soit  $x \in \mathbb{R}$  (ou  $x \in \mathbb{C}$ ) et  $n \in \mathbb{N}$ . On a alors :

$$\sum_{k=0}^n x^k = \begin{cases} n + 1 & \text{si } x = 1 \\ \frac{1 - x^{n+1}}{1 - x} & \text{si } x \neq 1 \end{cases}$$

Plus généralement pour une suite géométrique de raison  $q \neq 1$  :

$$\text{somme de termes successifs} = \text{1er terme} \times \frac{1 - q^{\text{nb de termes}}}{1 - q}$$

## Démonstration

### Exercice

Calculer  $1 + 2 + 4 + 8 + 16 + 32 + 64 + 128$

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Une généralisation

On se rappelle avec les petits Bernoullis :

## Une factorisation à connaître

Soient  $a$  et  $b$  deux réels (ou deux complexes), alors :

$$a^n - b^n = (a - b) \underbrace{\sum_{k=0}^{n-1} a^{n-1-k} b^k}_{b_a^{n-1}(b)}$$

⇒ Savoir manipuler  
des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques  
problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

# Une généralisation

On se rappelle avec les petits Bernoullis :

## Une factorisation à connaître

Soient  $a$  et  $b$  deux réels (ou deux complexes), alors :

$$a^n - b^n = (a - b) \underbrace{\sum_{k=0}^{n-1} a^{n-1-k} b^k}_{b_a^{n-1}(b)}$$

### Exercice

Ecrire  $3^n - 2^n$ , sous forme d'une addition de  $n$  termes

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Une généralisation

On se rappelle avec les petits Bernoullis :

## Une factorisation à connaître

Soient  $a$  et  $b$  deux réels (ou deux complexes), alors :

$$a^n - b^n = (a - b) \underbrace{\sum_{k=0}^{n-1} a^{n-1-k} b^k}_{b_a^{n-1}(b)}$$

### Exercice

Ecrire  $3^n - 2^n$ , sous forme d'une addition de  $n$  termes

**Application** Factoriser  $a^5 - b^5$ .

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Une généralisation

On se rappelle avec les petits Bernoullis :

## Une factorisation à connaître

Soient  $a$  et  $b$  deux réels (ou deux complexes), alors :

$$a^n - b^n = (a - b) \underbrace{\sum_{k=0}^{n-1} a^{n-1-k} b^k}_{b_a^{n-1}(b)}$$

### Exercice

Ecrire  $3^n - 2^n$ , sous forme d'une addition de  $n$  termes

**Application** Factoriser  $a^5 - b^5$ .

### Démonstration

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Une généralisation

On se rappelle avec les petits Bernoullis :

## Une factorisation à connaître

Soient  $a$  et  $b$  deux réels (ou deux complexes), alors :

$$a^n - b^n = (a - b) \underbrace{\sum_{k=0}^{n-1} a^{n-1-k} b^k}_{b_a^{n-1}(b)}$$

### Exercice

Ecrire  $3^n - 2^n$ , sous forme d'une addition de  $n$  termes

**Application** Factoriser  $a^5 - b^5$ .

### **Démonstration**

Exercice Peut-on factoriser  $a^n + b^n$  ? Si oui, faites-le.

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

⇒ **Savoir manipuler des sommes. Résultats exacts**

⇒ **Sommes doubles**

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Heuristique. Somme matricielle

## Heuristique. Somme à multiples indices

On peut faire une somme d'éléments pris dans un ensemble fini.  
Mais la description de ces éléments n'est pas toujours naturellement donnée sous la forme  $x_i, i \in \llbracket 0, n \rrbracket$ .

Parfois les éléments apparaissent comme les éléments d'un tableau (*matrice*) et sont donc doublement (ou plus) indexés :  
 $x_{i,j}, i \in \mathbb{N}_n, j \in \mathbb{N}_m$ .

Les choses se présentent différemment selon que  $i$  et  $j$  sont « indépendants » entre eux ou non.

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Cas $i$ et $j$ indépendants. Produit cartésien d'ensembles

**Analyse** Du sens des formules

Comment décrire avec  $\sum$  la somme

$$S = a_{1,1} + a_{1,2} + \dots + a_{1,m} + a_{2,1} + \dots + a_{2,m} + \dots + a_{n,1} + \dots + a_{n,m} ?$$

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Cas $i$ et $j$ indépendants. Produit cartésien d'ensembles

**Analyse** Du sens des formules

Comment décrire avec  $\sum$  la somme

$$S = a_{1,1} + a_{1,2} + \dots + a_{1,m} + a_{2,1} + \dots + a_{2,m} + \dots + a_{n,1} + \dots + a_{n,m} ?$$

### Définition - Somme double

On considère une famille de nombres réels ou complexes  $(a_{i,j})$  indexée par deux indices  $i$  et  $j$ ,  $i$  compris entre 1 et  $n$ ,  $j$  compris entre 1 et  $m$  où  $n$  et  $m$  sont deux entiers non nuls donnés :

$$\begin{array}{ccccccc} a_{1,1} & \dots & a_{1,j} & \dots & \dots & a_{1,m} \\ \vdots & & \vdots & & & \vdots \\ a_{i,1} & \dots & a_{i,j} & \dots & \dots & a_{i,m} \\ \vdots & & \vdots & & & \vdots \\ a_{n,1} & \dots & a_{n,j} & \dots & \dots & a_{n,m} \end{array}$$

Leur somme est notée  $\sum_{1 \leq i \leq n, 1 \leq j \leq m} a_{i,j} = \sum_{(i,j) \in \llbracket 1, n \rrbracket \times \llbracket 1, j \rrbracket} a_{i,j}$ .

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Cas $i$ et $j$ indépendants. Produit cartésien d'ensembles

### Savoir-faire. Somme multiple (indépendance)

Première façon : sommer des termes ligne par ligne, puis d'additionner les résultats :

$$\sum_{1 \leq i \leq n, 1 \leq j \leq m} a_{i,j} = \sum_{i=1}^n \underbrace{\left( \sum_{j=1}^m a_{i,j} \right)}_{\text{somme de la ligne } i} = \sum_{i=1}^n \sum_{j=1}^m a_{i,j},$$

Seconde façon : sommer d'abord les termes colonne par colonne puis d'additionner les résultats :

$$\sum_{1 \leq i \leq n, 1 \leq j \leq m} a_{i,j} = \sum_{j=1}^m \underbrace{\left( \sum_{i=1}^n a_{i,j} \right)}_{\text{somme de la colonne } j} = \sum_{j=1}^m \sum_{i=1}^n a_{i,j}.$$

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Cas $i$ et $j$ indépendants. Produit cartésien d'ensembles

### Savoir-faire. Somme multiple (indépendance)

Première façon : sommer des termes ligne par ligne, puis d'additionner les résultats :

$$\sum_{1 \leq i \leq n, 1 \leq j \leq m} a_{i,j} = \sum_{i=1}^n \underbrace{\left( \sum_{j=1}^m a_{i,j} \right)}_{\text{somme de la ligne } i} = \sum_{i=1}^n \sum_{j=1}^m a_{i,j},$$

Seconde façon : sommer d'abord les termes colonne par colonne puis d'additionner les résultats :

$$\sum_{1 \leq i \leq n, 1 \leq j \leq m} a_{i,j} = \sum_{j=1}^m \underbrace{\left( \sum_{i=1}^n a_{i,j} \right)}_{\text{somme de la colonne } j} = \sum_{j=1}^m \sum_{i=1}^n a_{i,j}.$$

### Remarque Diagonale

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Factorisation

## Exercice

Calculer

$$\sum_{1 \leq i \leq n, 1 \leq j \leq p} ij$$

⇒ Savoir manipuler  
des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques  
problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

## Exercice

Calculer  $\sum_{1 \leq i \leq n, 1 \leq j \leq p} ij$  Comme le montre l'exercice précédent :

## Proposition - Produit de deux sommes (développement ou factorisation)

Soient des réels (ou des complexes)  $a_i$  et  $b_j$ ,  
 $1 \leq i \leq n, 1 \leq j \leq p$ . Alors :

$$\left( \sum_{i=1}^n a_i \right) \left( \sum_{j=1}^p b_j \right) = \sum_{(i,j) \in \llbracket 1, n \rrbracket \times \llbracket 1, p \rrbracket} a_i b_j$$

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Exercice

Calculer  $\sum_{1 \leq i \leq n, 1 \leq j \leq p} ij$  Comme le montre l'exercice précédent :

## Proposition - Produit de deux sommes (développement ou factorisation)

Soient des réels (ou des complexes)  $a_i$  et  $b_j$ ,  
 $1 \leq i \leq n, 1 \leq j \leq p$ . Alors :

$$\left( \sum_{i=1}^n a_i \right) \left( \sum_{j=1}^p b_j \right) = \sum_{(i,j) \in \llbracket 1, n \rrbracket \times \llbracket 1, p \rrbracket} a_i b_j$$

## Démonstration

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Cas $i$ et $j$ dépendants

## Heuristique. Cas : $i$ et $j$ dépendants

Ici on somme seulement certains termes du tableau rectangulaire  $a_{i,j}$ .

Et donc les indices sont dépendants l'un de l'autre !

Par exemple, dans cette situation, les valeurs prises par  $j$  (à l'intérieur de la somme) dépendent de celles prises par  $i$  (à l'extérieur de la somme). Il y a, en revanche, souvent liberté dans le choix de l'ordre de sommation (d'abord  $i$  ou d'abord  $j$ ).

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Cas $i$ et $j$ dépendants

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

## Heuristique. Cas : $i$ et $j$ dépendants

Ici on somme seulement certains termes du tableau rectangulaire  $a_{i,j}$ .

Et donc les indices sont dépendants l'un de l'autre !

Par exemple, dans cette situation, les valeurs prises par  $j$  (à l'intérieur de la somme) dépendent de celles prises par  $i$  (à l'extérieur de la somme). Il y a, en revanche, souvent liberté dans le choix de l'ordre de sommation (d'abord  $i$  ou d'abord  $j$ ).

**Analyse**  $G = \{a_{i,j}, i \in \mathbb{N}_n, j \in A_i\}$

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

Proposition - Somme double classique  $\sum_{1 \leq j \leq i \leq n} a_{i,j}$

Soit  $(a_{i,j})_{(i,j) \in \mathbb{N}^2}$  une famille de nombres réels ou complexes :

$$\sum_{1 \leq j \leq i \leq n} a_{i,j} = \sum_{i=1}^n \sum_{j=1}^i a_{i,j} = \sum_{j=1}^n \sum_{i=j}^n a_{i,j}.$$

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Exercices

## Exercice

Calculer  $\sum_{1 \leq i \leq j \leq n} ij$

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

## Exercice

Calculer  $\sum_{1 \leq i \leq j \leq n} ij$

## Savoir-faire. Somme multiple (dépendance)

On ordonne les indices de sommation :

1. On choisit celle qui sera la plus à l'extérieur (à gauche) des symboles  $\sum$ . Elle ne dépend que des paramètres fixés et d'aucun indice.
2. on choisit ensuite la suivante. Elle dépend des paramètres fixés et de l'indice précédent.

...

Exemple : 
$$\sum_{1 \leq i < j \leq n} a_{i,j} = \sum_{i=1}^{n-1} \left( \sum_{j=i+1}^n a_{i,j} \right) = \sum_{j=2}^n \sum_{i=1}^{j-1} a_{i,j},$$

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

## Exercice

Calculer  $\sum_{1 \leq i \leq j \leq n} ij$

## Savoir-faire. Somme multiple (dépendance)

On ordonne les indices de sommation :

1. On choisit celle qui sera la plus à l'extérieur (à gauche) des symboles  $\sum$ . Elle ne dépend que des paramètres fixés et d'aucun indice.
2. on choisit ensuite la suivante. Elle dépend des paramètres fixés et de l'indice précédent.

...

Exemple : 
$$\sum_{1 \leq i < j \leq n} a_{i,j} = \sum_{i=1}^{n-1} \left( \sum_{j=i+1}^n a_{i,j} \right) = \sum_{j=2}^n \sum_{i=1}^{j-1} a_{i,j},$$

## Analyse Sens de ces modes de sommations

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

⇒ **Savoir manipuler des sommes. Résultats exacts**

⇒ **Sommes doubles**

1. Quelques problèmes

2. Symboles  $\Sigma$  et  $\Pi$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\Sigma$  et  $\Pi$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

⇒ Savoir manipuler  
des sommes.

Résultats exacts

⇒ Sommes doubles

## Exercice

Retrouver la valeur de  $S = \sum_{k=1}^n k$ , en notant que  $S = \sum_{k=1}^n \sum_{i=1}^k 1$ .

1. Quelques  
problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

⇒ Savoir manipuler des sommes.

Résultats exacts

⇒ Sommes doubles

## Exercice

Retrouver la valeur de  $S = \sum_{k=1}^n k$ , en notant que  $S = \sum_{k=1}^n \sum_{i=1}^k 1$ .

$$\text{Montrer que } \left( \sum_{k=1}^n d_k \right)^2 = \sum_{k=1}^n d_k^2 + 2 \sum_{1 \leq i < j \leq n} d_i d_j.$$

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

On peut aussi démontrer l'inégalité célèbre de Cauchy-Schwarz :

## Exercice

1. En développant  $\sum_{1 \leq i < j \leq 3} (a_i b_j - a_j b_i)^2$ , montrer que

$$\left( \sum_{k=1}^3 a_k b_k \right)^2 \leq \left( \sum_{k=1}^3 a_k^2 \right) \left( \sum_{k=1}^3 b_k^2 \right).$$

2. De même, montrer l'inégalité de Cauchy-Schwarz :

$$\left( \sum_{k=1}^n a_k b_k \right)^2 \leq \left( \sum_{k=1}^n a_k^2 \right) \left( \sum_{k=1}^n b_k^2 \right)$$

3. A quelle condition a-t-on :

$$\sum_{k=1}^n (a_i b_i)^2 = \left( \sum_{k=1}^n a_i \right)^2 \left( \sum_{k=1}^n b_k \right)^2 ?$$

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Conclusion

## Objectifs

- ⇒ Savoir manipuler des sommes. Calculs exacts
- ⇒ Sommes doubles

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\Sigma$  et  $\Pi$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

# Conclusion

## Objectifs

⇒ Savoir manipuler des sommes. Calculs exacts

- ▶ Utiliser le changement d'indices

⇒ Savoir manipuler  
des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques  
problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

# Conclusion

## Objectifs

⇒ Savoir manipuler des sommes. Calculs exacts

- ▶ Utiliser le changement d'indices
- ▶ Exploiter des récurrences

⇒ Savoir manipuler  
des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques  
problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

# Conclusion

## Objectifs

⇒ Savoir manipuler des sommes. Calculs exacts

- ▶ Utiliser le changement d'indices
- ▶ Exploiter des récurrences
- ▶ Sommation par paquets

⇒ Savoir manipuler  
des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques  
problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

# Conclusion

## Objectifs

⇒ Savoir manipuler des sommes. Calculs exacts

- ▶ Utiliser le changement d'indices
- ▶ Exploiter des récurrences
- ▶ Sommation par paquets
- ▶ Exploiter le télescopage (parfait pour un résultat exact)

⇒ Savoir manipuler  
des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques  
problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

# Conclusion

## Objectifs

⇒ Savoir manipuler des sommes. Calculs exacts

- ▶ Utiliser le changement d'indices
- ▶ Exploiter des récurrences
- ▶ Sommation par paquets
- ▶ Exploiter le télescopage (parfait pour un résultat exact)
- ▶ Quelques sommes à connaître :  $\sum_{k=1}^n k$ ,  $\sum_{k=1}^n k^2$ ,  $\sum_{k=1}^n k^3$  et  $\sum_{k=1}^n q^k$ .

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Conclusion

## Objectifs

- ⇒ Savoir manipuler des sommes. Calculs exacts
- ⇒ Sommes doubles

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\Sigma$  et  $\Pi$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application

# Conclusion

## Objectifs

⇒ Savoir manipuler des sommes. Calculs exacts

⇒ Sommes doubles

▶ Sens de  $\sum_{i,j \in A} a_{i,j}$

⇒ Savoir manipuler  
des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques  
problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

# Conclusion

## Objectifs

⇒ Savoir manipuler des sommes. Calculs exacts

⇒ Sommes doubles

▶ Sens de  $\sum_{i,j \in A} a_{i,j}$

▶ Si  $i$  et  $j$  indépendants : le calcul est simple (factorisation ou succession)

⇒ Savoir manipuler  
des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques  
problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

# Conclusion

## Objectifs

⇒ Savoir manipuler des sommes. Calculs exacts

⇒ Sommes doubles

▶ Sens de  $\sum_{i,j \in A} a_{i,j}$

▶ Si  $i$  et  $j$  indépendants : le calcul est simple (factorisation ou succession)

▶ Si  $i$  et  $j$  sont liés : on s'adapte (c'est plus compliqué)

⇒ Savoir manipuler  
des sommes.

Résultats exacts

⇒ Sommes doubles

1. Quelques  
problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

# Conclusion

## Objectifs

⇒ Savoir manipuler des sommes. Calculs exacts

⇒ Sommes doubles

- ▶ Sens de  $\sum_{i,j \in A} a_{i,j}$
- ▶ Si  $i$  et  $j$  indépendants : le calcul est simple (factorisation ou succession)
- ▶ Si  $i$  et  $j$  sont liés : on s'adapte (c'est plus compliqué)
- ▶ Un savoir faire :  $1 \leq i \leq j \leq n$

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

# Conclusion

## Objectifs

⇒ Savoir manipuler des sommes. Calculs exacts

⇒ Sommes doubles

- ▶ Sens de  $\sum_{i,j \in A} a_{i,j}$
- ▶ Si  $i$  et  $j$  indépendants : le calcul est simple (factorisation ou succession)
- ▶ Si  $i$  et  $j$  sont liés : on s'adapte (c'est plus compliqué)
- ▶ Un savoir faire :  $1 \leq i \leq j \leq n$
- ▶ Cauchy-Schwarz

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\sum$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles  
(multiples...)

2.6. Exercices d'application

## Objectifs

- ⇒ Savoir manipuler des sommes. Calculs exacts
- ⇒ Sommes doubles

## Pour la prochaine fois

- ▶ Lecture du cours :  
Chap 7 : Calculs avec des sommes et des produits.  
3. Coefficients binomiaux et formule du binôme
- ▶ Exercices 162, 164 & 169
- ▶ TD de jeudi :  
8h-10h : 163, 166, 168, 172, 174, 178  
10h-12h : 165, 167, 169, 173, 175, 181

⇒ Savoir manipuler des sommes.  
Résultats exacts

⇒ Sommes doubles

1. Quelques problèmes

2. Symboles  $\Sigma$  et  $\prod$

2.1. Définition

2.2. Quatre règles opératoires

2.3. Avec Python

2.4. Des sommes connues

2.5. Sommes doubles (multiples...)

2.6. Exercices d'application