

Analyse Numérique I

Sup'Galilée, Ingénieurs MACS, 1ère année & L3-MIM

Cours: Cuvelier F. - *TD:* Darbas M. & Japhet C.







Laboratoire d'Analyse Géométrie et Applications
Institut Galilée
Université Sorbonne Paris Nord.

v. 2021/09/28

- Volume horaire : $10 \times 3h$ cours - $10 \times 3h$ TD,
- Prérequis des cours : *Equations différentielles*, *Projets numériques*, *Elements Finis*, *Volumes Finis*, ...
- Note finale : $(P_1 + P_2)/2$ avec points de bonus
 - ▶ P_1 partiel 1, 19 novembre 2021 (1h30),
 - ▶ P_2 partiel 2, 14 janvier 2022 (1h30),
- 2 groupes de TD, en alternant chaque semaine, 1 groupe en distanciel avec Mme Japhet et 1 groupe en présentiel avec Mme Darbas.,
- Un serveur Discord dédié aux cours et aux TDs,
- Un polycopié,
- Des pages web dédiées cours, TDs,
- Pas présent sur l'ENT!

Outils de base de l'analyse numérique et du calcul scientifique.

- Mathématiques
- Numériques
- Algorithmique

-  P.G. Ciarlet, *Analyse numérique et equations différentielles*, DUNOD, 2006.
-  _____, *Introduction à l'analyse numérique matricielle et à l'optimisation*, DUNOD, 2006.
-  M. Crouzeix and A.L. Mignot, *Analyse numérique des équations différentielles*, Mathématiques appliquées pour la maîtrise, Masson, 1992.
-  J.P. Demailly, *Analyse numérique et equations différentielles*, PUG, 1994.
-  J.-P. Demailly, *Analyse numérique et équations différentielles*, Grenoble Sciences, EDP Sciences, 2006.
-  J.P. Demailly, *Analyse numérique et équations différentielles*, Grenoble Sciences, EDP Sciences, 2012.



W. Gander, M.J. Gander, and F. Kwok, *Scientific computing : an introduction using maple and matlab*, Springer, Cham, 2014.



T. Huckle, *Collection of software bugs*
<http://www.zenger.informatik.tu-muenchen.de/persons/huckle/bugse.html>.



P. Lascaux and R. Théodor, *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, Analyse numérique matricielle appliquée à l'art de l'ingénieur, no. vol. 1 et 2, Dunod, 2004.



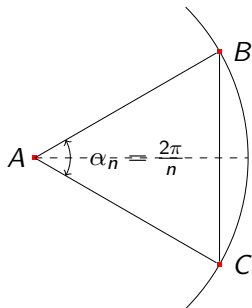
A. Quarteroni, R. Sacco, and F. Saleri, *Méthodes Numériques: Algorithmes, analyse et applications (French Edition)*, 1 ed., Springer, September 2007.

- Partie 1: Erreurs : arrondis, bug and Co.
- Partie 2: Langage algorithmique
- Partie 3: Rappels algèbre linéaire
- Partie 4: Résolution de systèmes non-linéaires
- Partie 5: Résolution de systèmes linéaires
- Partie 6: Polynômes d'interpolation
- Partie 7: Intégration numérique
- Partie 8: Dérivation numérique ?

Chapitre I

Erreurs : arrondis, bug and Co.

Un exemple : calcul approché de π



- Aire du cercle : $\mathcal{A} = \pi r^2$.
- Archimède vers 250 avant J-C :
 $\pi \in]3 + \frac{1}{7}, 3 + \frac{10}{71}[$ avec 96 polygônes.
- Algorithme polygônes inscrits ($r = 1$) :
 $\mathcal{A} = \lim_{n \rightarrow +\infty} \mathcal{A}_n = \lim_{n \rightarrow +\infty} \frac{n}{2} \sin(\alpha_n)$ et
 $\sin \frac{\alpha_n}{2} = \sqrt{\frac{1 - \sqrt{1 - \sin^2 \alpha_n}}{2}}$.

1: $s \leftarrow 1, n \leftarrow 4$

2: **Tantque** $s > 1e - 10$ **faire**

3: $s \leftarrow \text{sqrt}((1 - \text{sqrt}(1 - s * s)))/2$

4: $n \leftarrow 2 * n$

5: $A \leftarrow (n/2) * s$

6: **Fin Tantque**

▷ Initialisations

▷ Arrêt si $s = \sin(\alpha)$ est petit

▷ nouvelle valeur de $\sin(\alpha/2)$

▷ nouvelle valeur de n

▷ nouvelle valeur de l'aire du polygône

Un exemple : calcul approché de π

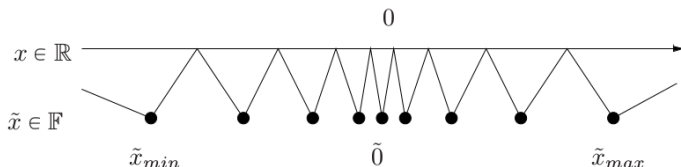
n	\mathcal{A}_n	$ \mathcal{A}_n - \pi $	$\sin(\alpha_n)$
4	2.000000000000000	1.141593e+00	1.000000e+00
64	3.13654849054594	5.044163e-03	9.801714e-02
4096	3.14159142150464	1.232085e-06	1.533980e-03
32768	3.14159263346325	2.012654e-08	1.917476e-04
65536	3.14159265480759	1.217796e-09	9.587380e-05
131072	3.14159264532122	8.268578e-09	4.793690e-05
524288	3.14159291093967	2.573499e-07	1.198423e-05
4194304	3.1415965370482	3.900115e-06	1.498030e-06
67108864	3.14245127249413	8.586189e-04	9.365235e-08
2147483648	0.000000000000000	3.141593e+00	0.000000e+00

Nombres flottants en machine

$$\begin{aligned}\tilde{x} &= \pm m \cdot b^e \\ m &= D.D \cdots D, \quad \text{mantisse} \\ e &= D \cdots D, \quad \text{exposant}\end{aligned}$$

où $D \in \{0, 1, \dots, b-1\}$ représente un chiffre et b la base.

Mantisse normalisée : 1er D (avant point) est non nul.



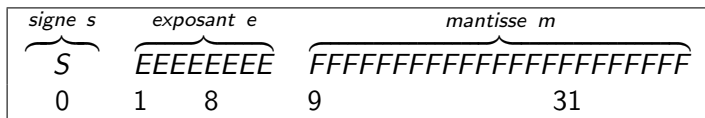
- **Précision machine** : plus petit nombre machine $\text{eps} > 0$ tel que

$$1 + \text{eps} > 1.$$

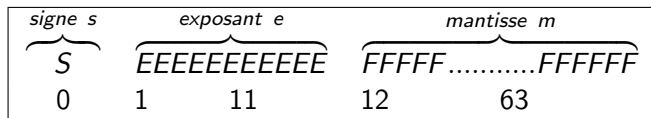
Matlab/Octave, langage C (double), ... : $\text{eps} = 2.220446049250313\text{e} - 16$

Système IEEE 754 (1985)

- **simple précision** : format 32 bits (4 octets), float en langage C.



- **double précision** : format 64 bits (8 octets), double en langage C.



dimension tableau	type	dimension (octets)	dimension total
1000×1000	float	4	$4 * 1000 * 1000 = 4Mo$
$10^4 \times 10^4$	float	4	$4 * 10^8 = 400Mo$
1000×1000	double	8	$8 * 1000 * 1000 = 8Mo$
$10^4 \times 10^4$	double	8	$8 * 10^8 = 800Mo$

En langage C (par ex.):

- `int a=2147483647,b; alors b=a+1; donne b==-2147483648,`
- `double a=1/2,b;b=a+1; donne a==0 et b==1,`

En Matlab (par ex.) :

- `x=eps; alors 1+eps==1 est faux (false=0)`
- `x=eps/2;y=1; alors (y+x)+x==1 est vrai et y+(x+x)==1 est faux.`
`(x + y) + z = x + (y + z)` n'est pas forcément vérifiée sur machine!

Il faut être attentifs aux signes.

- $f(x) = \frac{1}{1-\sqrt{1-x^2}}$, si $|x| \leq 0.5\sqrt{\text{eps}}$ alors $\sqrt{1-x^2} \equiv 1!$

$$\implies \bar{x} = 0.5\sqrt{\text{eps}}, \quad \boxed{\frac{1}{1-\sqrt{1-\bar{x}^2}} \equiv +\infty}$$

-

$$f(x) = \frac{1}{1-\sqrt{1-x^2}} = \frac{1}{1-\sqrt{1-x^2}} \times \frac{1+\sqrt{1-x^2}}{1+\sqrt{1-x^2}} = \frac{1+\sqrt{1-x^2}}{x^2}$$

$$\implies \bar{x} = 0.5\sqrt{\text{eps}}, \quad \boxed{\frac{1+\sqrt{1-\bar{x}^2}}{\bar{x}^2} \equiv 3.6029e+16}$$

Erreurs d'annulation : calcul de π , le retour

$$\sin \frac{\alpha_n}{2} = \sqrt{\frac{1 - \cos \alpha_n}{2}} = \sqrt{\frac{1 - \sqrt{1 - \sin^2 \alpha_n}}{2}},$$

C'est le calcul de $1 - \sqrt{1 - \sin^2 \alpha_n}$ qui pose problème!

$$\sin \frac{\alpha_n}{2} = \frac{\sqrt{1 + \sqrt{1 - \sin^2 \alpha_n}}}{\sqrt{1 + \sqrt{1 - \sin^2 \alpha_n}}} \sin \frac{\alpha_n}{2} = \frac{\sin \alpha_n}{\sqrt{2(1 + \sqrt{1 - \sin^2 \alpha_n})}}$$

- 1: $s \leftarrow 1, n \leftarrow 4,$ ▷ Initialisations
- 2: **Tantque** $s > 1e - 10$ **faire** ▷ Arrêt si $s = \sin(\alpha)$ est petit
- 3: $s \leftarrow s / \text{sqrt}(2 * (1 + \text{sqrt}(1 - s * s)))$ ▷ nouvelle valeur de $\sin(\alpha/2)$
- 4: $n \leftarrow 2 * n$ ▷ nouvelle valeur de n
- 5: $A \leftarrow (n/2) * s$ ▷ nouvelle valeur de l'aire du polygône
- 6: **Fin Tantque**

Un exemple : calcul approché de π

n	A_n	$ A_n - \pi $	$\sin(\alpha_n)$
4	2.000000000000000	1.141593e+00	1.000000e+00
64	3.13654849054594	5.044163e-03	9.801714e-02
4096	3.14159142151120	1.232079e-06	1.533980e-03
32768	3.14159263433856	1.925123e-08	1.917476e-04
65536	3.14159264877699	4.812807e-09	9.587380e-05
131072	3.14159265238659	1.203202e-09	4.793690e-05
524288	3.14159265351459	7.519985e-11	1.198422e-05
4194304	3.14159265358862	1.174172e-12	1.498028e-06
67108864	3.14159265358979	3.552714e-15	9.362676e-08
2147483648	3.14159265358979	1.332268e-15	2.925836e-09

Mais avant de poursuivre ...



(a) Pont de la Basse-Chaîne, *Angers* (1850)



(b) Takoma Narrows Bridge, *Washington* (1940)



(c) Millenium Bridge, *London* (2000)

Figure: Une histoire de ponts

Chapitre II

Langage algorithmique

- 1 Introduction
- 2 Pseudo-langage algorithmique
- 3 Méthodologie de construction
- 4 Pseudo-langage algorithmique (suite)

Definition 1.1: Petit Robert 97

Algorithmique : Enchaînement d'actions nécessaires à l'accomplissement d'une tâche.

Exemple 1 : permutation

Nous voulons permutter deux voitures sur un parking de trois places numérotées de 1 à 3 et ceci sans gêner la circulation.

La première voiture, une Sandero, est sur l'emplacement 2, la seconde, une Zoé, est sur l'emplacement 3.

Donner un algorithme permettant de résoudre cette tâche.

Exemple 2 :

Donner un algorithme permettant de résoudre

$$ax = b$$

Caractéristiques d'un *bon* algorithme

- Le problème ne souffre d'aucune ambiguïté \Rightarrow très clair.
- Combinaison d'opérations (actions) élémentaires.
- Pour toutes les données d'entrée, l'algorithme doit fournir un résultat en un nombre fini d'opérations.

Etape 1 : Définir clairement le problème.

Première approche méthodologique

Etape 1 : Définir clairement le problème.

Etape 2 : Rechercher une méthode de résolution (formules, ...)

Etape 1 : Définir clairement le problème.

Etape 2 : Rechercher une méthode de résolution (formules, ...)

Etape 3 : Ecrire l'algorithme (par raffinement successif pour des algorithmes *compliqués*).

- 1 Introduction
- 2 Pseudo-langage algorithmique
 - Les bases
 - Les instructions structurées
- 3 Méthodologie de construction
- 4 Pseudo-langage algorithmique (suite)

- constantes, variables,
- opérateurs (arithmétiques, relationnels, logiques),
- expressions,
- instructions (simples et composées),
- fonctions.

- Donnée \Rightarrow introduite par l'utilisateur
- Constante \Rightarrow symbole, identificateur non modifiable

Definition 2.1

Une variable est un objet dont la valeur est modifiable, qui possède un nom et un type (entier, caractère, réel, complexe, tableau, matrice, vecteur...).

Opérateurs arithmétiques

Nom	Symbole	Exemple
addition	+	$a + b$
soustraction	—	$a - b$
opposé	—	$-a$
produit	*	$a * b$
division	/	a/b

Opérateurs relationnels

Nom	Symbole	Exemple
identique	$==$	$a == b$
différent	$\sim =$	$a \sim = b$
inférieur	$<$	$a < b$
supérieur	$>$	$a > b$
inférieur ou égal	\leq	$a \leq b$
supérieur ou égal	\geq	$a \geq b$

Opérateurs logiques

Nom	Symbole	Exemple
négation	\sim	$\sim a$
ou	$ $	$a b$
et	$\&$	$a\&b$

Opérateur d'affectation

Nom	Symbole	Exemple
affectation	\leftarrow	$a \leftarrow b$

Definition 2.2

Une expression est un groupe d'opérandes (i.e. nombres, constantes, variables, ...) liées par certains opérateurs pour former un terme algébrique qui représente une valeur (i.e. un élément de donnée simple)

♥ Definition 2.3

Une expression est un groupe d'opérandes (i.e. nombres, constantes, variables, ...) liées par certains opérateurs pour former un terme algébrique qui représente une valeur (i.e. un élément de donnée simple)

Exemple d'expression numérique

$$(b * b - 4 * a * c) / (2 * a)$$

♥ Definition 2.4

Une expression est un groupe d'opérandes (i.e. nombres, constantes, variables, ...) liées par certains opérateurs pour former un terme algébrique qui représente une valeur (i.e. un élément de donnée simple)

Exemple d'expression numérique

$$(b * b - 4 * a * c) / (2 * a)$$

Opérandes \Rightarrow identifiants a , b , c ,
constantes 4 et 2.

♥ Definition 2.5

Une expression est un groupe d'opérandes (i.e. nombres, constantes, variables, ...) liées par certains opérateurs pour former un terme algébrique qui représente une valeur (i.e. un élément de donnée simple)

Exemple d'expression numérique

$$(b * b - 4 * a * c) / (2 * a)$$

Opérandes \Rightarrow identifiants a, b, c ,
constantes 4 et 2.

Opérateurs \Rightarrow symboles $*, -, \text{ et } /$

♥ Definition 2.6

Une expression est un groupe d'opérandes (i.e. nombres, constantes, variables, ...) liées par certains opérateurs pour former un terme algébrique qui représente une valeur (i.e. un élément de donnée simple)

Exemple d'expression booléenne

$$(x < 3.14)$$

♥ Definition 2.7

Une expression est un groupe d'opérandes (i.e. nombres, constantes, variables, ...) liées par certains opérateurs pour former un terme algébrique qui représente une valeur (i.e. un élément de donnée simple)

Exemple d'expression booléenne

$$(x < 3.14)$$

Opérandes \Rightarrow identifiants x et constantes 3.14

♥ Definition 2.8

Une expression est un groupe d'opérandes (i.e. nombres, constantes, variables, ...) liées par certains opérateurs pour former un terme algébrique qui représente une valeur (i.e. un élément de donnée simple)

Exemple d'expression booléenne

$$(x < 3.14)$$

Opérandes \Rightarrow identifiants x et constantes 3.14

Opérateurs \Rightarrow symboles $<$

Definition 2.9

Une **instruction** est un ordre ou un groupe d'ordres qui déclenche l'exécution de certaines actions par l'ordinateur. Il y a deux types d'instructions : simple et structuré.

- affectation d'une valeur a une variable.
- appel d'une fonction (procedure, subroutine, ... suivant les langages).

- ❶ les instructions composées, groupe de plusieurs instructions simples,
- ❷ les instructions répétitives, permettant l'exécution répétée d'instructions simples, (i.e. boucles «pour», «tant que»)
- ❸ les instructions conditionnelles, lesquels ne sont exécutées que si une certaine condition est respectée (i.e. «si»)

Exemple : boucle «pour»

Données : n

```
1:  $S \leftarrow 0$   
2: Pour  $i \leftarrow 1$  à  $n$  faire  
3:    $S \leftarrow S + \cos(i^2)$   
4: Fin Pour
```

Mais que fait-il?

Exemple : boucle «pour»

Données : n

- 1: $S \leftarrow 0$
- 2: **Pour** $i \leftarrow 1$ à n **faire**
- 3: $S \leftarrow S + \cos(i^2)$
- 4: **Fin Pour**

Mais que fait-il?

Calcul de $S = \sum_{i=1}^n \cos(i^2)$

Exemple : boucle «tant que»

```
1:  $i \leftarrow 0, x \leftarrow 1$   
2: Tantque  $i < 1000$  faire  
3:    $x \leftarrow x + i * i$   
4:    $i \leftarrow i + 1$   
5: Fin Tantque
```

Mais que fait-il?

Exemple : boucle «tant que»

```
1:  $i \leftarrow 0, x \leftarrow 1$   
2: Tantque  $i < 1000$  faire  
3:    $x \leftarrow x + i * i$   
4:    $i \leftarrow i + 1$   
5: Fin Tantque
```

Mais que fait-il?

Calcul de $x = 1 + \sum_{i=0}^{999} i^2$

Exemple : instructions conditionnelles «si»

Données :

age : un réel.

- 1: **Si** *age* \geq 18 **alors**
- 2: affiche('majeur')
- 3: **Sinon Si** *age* \geq 0 **alors**
- 4: affiche('mineur')
- 5: **Sinon**
- 6: affiche('en devenir')
- 7: **Fin Si**

- 1 Introduction
- 2 Pseudo-langage algorithmique
- 3 Méthodologie de construction
 - Principe
 - Exercices
- 4 Pseudo-langage algorithmique (suite)

Description du problème

- Spécification d'un ensemble de données
Origine : énoncé, hypothèses, sources externes, ...
- Spécification d'un ensemble de buts à atteindre
Origine : résultats, opérations à effectuer, ...
- Spécification des contraintes

Recherche d'une méthode de résolution

- Clarifier l'énoncé.
- Simplifier le problème.
- Ne pas chercher à le traiter directement dans sa globalité.
- S'assurer que le problème est soluble (sinon problème d'indécidabilité!)
- Recherche d'une stratégie de construction de l'algorithme
- Décomposer le problème en sous problèmes partiels plus simples : raffinement.
- Effectuer des raffinements successifs.
- Le niveau de raffinement le plus élémentaire est celui des instructions.

- Le type des données et des résultats doivent être précisés.
- L'algorithme doit fournir au moins un résultat.
- L'algorithme doit être exécuté en un nombre fini d'opérations.
- L'algorithme doit être spécifié clairement, sans la moindre ambiguïté.



Exercice 3.1

Ecrire un algorithme permettant de calculer

$$S(x) = \sum_{k=1}^n k \sin(2 * k * x)$$





Exercice 3.2

Ecrire un algorithme permettant de calculer

$$P(z) = \prod_{n=1}^k \sin(2 * k * z/n)^k$$





Exercice 3.3

Reprendre les trois exercices précédents en utilisant les boucles «tant que».

- ① Introduction
- ② Pseudo-langage algorithmique
- ③ Méthodologie de construction
- ④ Pseudo-langage algorithmique (suite)
 - Les fonctions
 - Exemple : résolution d'une équation
 - Exemple : produit scalaire

Les fonctions permettent

- d'automatiser certaines tâches répétitives au sein d'un même algorithme,
- d'ajouter à la clarté de l'algorithme,
- l'utilisation de portion de code dans un autre algorithme,
- ...

- les fonctions d'affichage et de lecture : **Affiche**, **Lit**

Les fonctions prédéfinies

- les fonctions d'affichage et de lecture : **Affiche**, **Lit**
- les fonctions mathématiques :

\sin , \cos , \exp , \dots

Les fonctions prédéfinies

- les fonctions d'affichage et de lecture : **Affiche**, **Lit**
- les fonctions mathématiques :

\sin , \cos , \exp , ...

- les fonctions de gestion de fichiers
- ...

Ecrire ses propres fonctions

- ❶ Que doit-on calculer/réaliser précisément (but)?
- ❷ Quelles sont les données (avec leurs limitations)?

```
Fonction [ $args_1, \dots, args_n$ ]  $\leftarrow$  NomFonction(  $arge_1, \dots, arge_m$  )  
    instructions  
Fin Fonction
```

```
Fonction  $args \leftarrow$  NomFonction(  $arge_1, \dots, arge_m$  )  
    instructions  
Fin Fonction
```

Ecrire une fonction permettant de résoudre

$$ax + b = 0$$

- But :
- Données :
- Résultats :

Ecrire une fonction permettant de résoudre

$$ax + b = 0$$

- But :
trouver $x \in \mathbb{R}$ solution de $ax + b = 0$.
- Données :
- Résultats :

Ecrire une fonction permettant de résoudre

$$ax + b = 0$$

- But :
trouver $x \in \mathbb{R}$ solution de $ax + b = 0$.
- Données :
 $a \in \mathbb{R}^*$ et $b \in \mathbb{R}$.
- Résultats :

Ecrire une fonction permettant de résoudre

$$ax + b = 0$$

- But :
trouver $x \in \mathbb{R}$ solution de $ax + b = 0$.
- Données :
 $a \in \mathbb{R}^*$ et $b \in \mathbb{R}$.
- Résultats :
 $x \in \mathbb{R}$.

Algorithme 1 Exemple de fonction : Résolution de l'équation du premier degré $ax + b = 0$.

Données : a : nombre réel différent de 0
 b : nombre réel.

Résultat : x : un réel.

- 1: **Fonction** $x \leftarrow \text{REPD}(a, b)$
 - 2: $x \leftarrow -b/a$
 - 3: **Fin Fonction**
-

Ecrire une fonction permettant de calculer le produit des deux vecteurs \mathbf{u} et \mathbf{v} de \mathbb{R}^n .

- Formule: $\langle \mathbf{u}, \mathbf{v} \rangle \stackrel{\text{def}}{=} \sum_{i=1}^n u_i v_i$
- But :
- Données :
- Résultats :

Ecrire une fonction permettant de calculer le produit des deux vecteurs \mathbf{u} et \mathbf{v} de \mathbb{R}^n .

- Formule: $\langle \mathbf{u}, \mathbf{v} \rangle \stackrel{\text{def}}{=} \sum_{i=1}^n u_i v_i$
- But :
calculer $s = \langle \mathbf{u}, \mathbf{v} \rangle \in \mathbb{R}$
- Données :
- Résultats :

Ecrire une fonction permettant de calculer le produit des deux vecteurs \mathbf{u} et \mathbf{v} de \mathbb{R}^n .

- Formule: $\langle \mathbf{u}, \mathbf{v} \rangle \stackrel{\text{def}}{=} \sum_{i=1}^n u_i v_i$
- But :
calculer $s = \langle \mathbf{u}, \mathbf{v} \rangle \in \mathbb{R}$
- Données :
 $\mathbf{u} \in \mathbb{R}^n$ et $\mathbf{v} \in \mathbb{R}^n$.
- Résultats :

Ecrire une fonction permettant de calculer le produit des deux vecteurs \mathbf{u} et \mathbf{v} de \mathbb{R}^n .

- Formule: $\langle \mathbf{u}, \mathbf{v} \rangle \stackrel{\text{def}}{=} \sum_{i=1}^n u_i v_i$
- But :
calculer $s = \langle \mathbf{u}, \mathbf{v} \rangle \in \mathbb{R}$
- Données :
 $\mathbf{u} \in \mathbb{R}^n$ et $\mathbf{v} \in \mathbb{R}^n$.
- Résultats :
 $s \in \mathbb{R}$.

Algorithme 2 Calcul du produit scalaire de deux vecteurs de \mathbb{R}^n

Données : \mathbf{u} : vecteur de \mathbb{R}^n
 \mathbf{v} : vecteur de \mathbb{R}^n .

Résultat : s : un réel tel que $s = \langle \mathbf{u}, \mathbf{v} \rangle$.

```
1: Fonction  $s \leftarrow$  PS (  $\mathbf{u}, \mathbf{v}$  )  
2:    $s \leftarrow 0$   
3:   Pour  $i \leftarrow 1$  à  $n$  faire  
4:      $s \leftarrow s + \mathbf{u}(i) * \mathbf{v}(i)$   
5:   Fin Pour  
6: Fin Fonction
```

Chapitre III

Rappels algèbre linéaire

Chapitre IV

Résolution de systèmes non-linéaires

Chapitre V

Résolution de systèmes linéaires

Chapitre VI

Polynômes d'interpolation

Chapitre VII

Intégration numérique

Chapitre VIII

Dérivation numérique ?