

Méthodes Numériques II

Sup'Galilée, Ingénieurs Energétique, 1ère année (S6)

François Cuvelier

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

2024/01/25

- Volume horaire : $8 \times 3h$ cours/TDs, $4 \times 4h$ TPs.
- Prérequis du cours :
 - ▶ Cours commun *Mathématiques pour l'ingénieur, ...*
 - ▶ *Algorithmique*
 - ▶ Langage de programmation (TP): *Matlab/Octave*
- Note finale : $(2P + CC + TP)/4$.
 - ▶ P : partiel du 23 avril 2024 (2h30) ?,
 - ▶ TP : travaux pratiques en mai et juin,
 - ▶ CC : contrôle continu (2 interrogations écrites de 30mn, séances 4 (27/02) et 7 (19/03)?).

- Algorithmique numérique.
- Poursuite de l'apprentissage de Matlab/Octave.
- Résolution numériques d'équations différentielles ordinaires (E.D.O.[fr] ou O.D.E.[en]).
- Résolution numériques d'équations aux dérivées partielles (E.D.P.[fr] ou P.D.E.[en]) par des méthodes de différences finies.

- Chapitre I : Algorithmique numérique
- Chapitre II : Dérivation numérique
- Chapitre III : Résolution numérique des E.D.O.
- Chapitre IV : Résolution numérique des E.D.P.

Definition 1.1: Petit Robert 97

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

Exemple 1 : permutation

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

La première voiture, une Renault Zoé, est sur l'emplacement 2, la seconde, une Citroën C1, 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

- Il 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, ...)

Première approche méthodologique

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 Algorithme: méthodologie de construction
 - Principe
 - Exercices
- 4 Pseudo-langage algorithmique (suite)
 - Les fonctions
 - Exemple : résoudre $ax = b$
- 5 Histoire de ponts

- 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	\neq	$a \neq b$
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 $*$, $-$ 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.

Instructions simples

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

- 1 Introduction
- 2 Pseudo-langage algorithmique
 - Les bases
 - Les instructions structurées
- 3 Algorithme: méthodologie de construction
- 4 Pseudo-langage algorithmique (suite)
 - Principe
 - Exercices
 - Les fonctions
 - Exemple : résoudre $ax = b$
- 5 Histoire de ponts

Instructions structurées

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

Exemple : boucle «pour»

Algorithm Exemple boucle «pour»

Données : n , un entier positif

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

Listing: (Matlab) Exemple boucle for

```
n=input('n=');      1
assert(n>=0)        2
S=0;                3
for i=1:n           4
    S=S+cos(i^2);   5
end                 6
```

Mais que fait-il?

$S = ?$

Exemple : boucle «tant que»

Algorithm 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**
-

Listing: (Matlab) Exemple boucle while

```
i=0;x=1; 1
while (i<1000) 2
    x=x+i*i; 3
    i=i+1; 4
end 5
```

Mais que fait-il?

$i = ?$, $x = ?$

Exemple : instructions conditionnelles «si»

Algorithm Exemple instruction «si»

Données : *age*, un entier.

- 1: **Si** *age* \geq 64 **alors**
 - 2: affiche('retraite')
 - 3: **Sinon Si** *age* \geq 18 **alors**
 - 4: affiche('majeur')
 - 5: **Sinon**
 - 6: affiche('mineur')
 - 7: **Fin Si**
-

Listing: (Matlab) Exemple instruction if

```
age=input('age=');      1
if age>=64              2
    disp('retraite')    3
elseif age>=18         4
    disp('majeur')     5
else                   6
    disp('mineur')     7
end                   8
```

- 1 Introduction
- 2 Pseudo-langage algorithmique
 - Les bases
 - Les instructions structurées
- 3 Algorithme: méthodologie de construction
 - Principe
 - Exercices
- 4 Pseudo-langage algorithmique (suite)
 - Les fonctions
 - Exemple : résoudre $ax = b$
- 5 Histoire de ponts

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.
- Effectuer des raffinements successifs de chaque sous problème. Le niveau de raffinement le plus élémentaire étant celui des instructions.

Réalisation d'un algorithme

- Les types des données et des résultats doivent être précisés.
- L'algorithme doit fournir au moins un résultat (qui peut être graphique).
- L'algorithme doit être exécuté en un nombre fini d'opérations.
- L'algorithme doit être spécifié clairement, sans la moindre ambiguïté.

- 1 Introduction
- 2 Pseudo-langage algorithmique
 - Les bases
 - Les instructions structurées
- 3 Algorithme: méthodologie de construction
 - Principe
 - **Exercices**
- 4 Pseudo-langage algorithmique (suite)
 - Les fonctions
 - Exemple : résoudre $ax = b$
- 5 Histoire de ponts

Les deux exercices qui suivent sont intentionnellement mal rédigés!!!



Exercice

Ecrire un algorithme permettant de calculer

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





Exercice

Ecrire un algorithme permettant de calculer

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





Exercice

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

- 1 Introduction
- 2 Pseudo-langage algorithmique
 - Les bases
 - Les instructions structurées
- 3 Algorithme: méthodologie de construction
 - Principe
 - Exercices
- 4 Pseudo-langage algorithmique (suite)
 - Les fonctions
 - Exemple : résoudre $ax = b$
- 5 Histoire de ponts

Les fonctions permettent

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

- 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

Avant d'écrire une fonction, voici les questions à se poser:

- 1 Que doit-elle calculer/réaliser précisément (but)?
- 2 Quelles sont ses 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
```

- 1 Introduction
- 2 Pseudo-langage algorithmique
 - Les bases
 - Les instructions structurées
- 3 Algorithme: méthodologie de construction
 - Principe
 - Exercices
- 4 Pseudo-langage algorithmique (suite)
 - Les fonctions
 - Exemple : résoudre $ax = b$
- 5 Histoire de ponts

Résoudre $ax = b$

Ecrire une fonction permettant de résoudre

$$ax = b$$

- **But :**
- **Données :**
- **Résultats :**

Résoudre $ax = b$

Ecrire une fonction permettant de résoudre

$$ax = b$$

- **But :**

Énoncé insuffisamment précis! On choisit ici le problème:

$a \in \mathbb{R}^*$ et $b \in \mathbb{R}$ donnés, trouver $x \in \mathbb{R}$ solution de $ax = b$

On aurait pu prendre a une matrice réelle d'ordre n et $b \in \mathbb{R}^n$...

- **Données :**

$a \in \mathbb{R}^*$ et $b \in \mathbb{R}$.

- **Résultats :**

$x \in \mathbb{R}$.

Résoudre $ax = b$

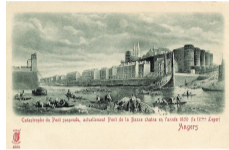
Algorithm Exemple de fonction : Résolution de l'équation du premier degré $ax = b$.

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**
-

Mais avant de poursuivre ...



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



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



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

Figure: Une histoire de ponts