Méthodes Numériques (G3SEMN)*

Sup'Galilée, Ingénieurs Energétique, 1ère année

François Cuvelier

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

2026/02/10

*Compilé le 2026/02/10 à 08:12:02

Divers

- 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 14 avril 2026 (2h30) ?,
  - TP : travaux pratiques en mai et juin,
  - CC : contrôle continu (2 interrogations écrites de 30mn, séances 4 (10/03) et 8 (30/03)?).
- mail: cuvelier@math.univ-paris13.fr
- web : https://www.math.univ-paris13.fr/~cuvelier
  ☛ rubrique: `Enseignements/Ing. Energétique/2025-2026`
- polycopié disponible sur page web.

Objectifs

- 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.

Plan du cours

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

Définition 1.1 : *Petit Robert 97*

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

Exemple 1.1 : *permutation*

Nous voulons permutter deux voitures sur un parking de trois places numérotées de 1 à 3 et ceci sans géner 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 1.2

Donner un algorithme permettant de résoudre

$$ax = b$$

Caractéristiques d'un *bon* algorithme:

- Il ne souffre d'aucune ambiguité $\Rightarrow$ 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.

Première approche méthodologique:

1. Définir clairement le problème.
2. Rechercher une méthode de résolution (formules, ...).
3. Ecrire l'algorithme (par raffinement successif pour des algorithmes *compliqués*).

Plan

- **Vocabulaire de base:**
  1. constantes,variables,
  2. opérateurs (arithmétiques, relationnels, logiques),
  3. expressions,
  4. instructions (simples et composées),
  5. fonctions.

1. **Constantes, variables:**
   - Constante ⇒ symbole, identificateur non modifiable
   - Une variable est un objet dont la valeur est modifiable, qui possède un nom et un type (entier, caractère, réel, complexe, tableau, matrice, vecteur...).
   - Donnée ⇒ introduite par l'utilisateur (souvent sous forme d'une variable)

2 Opérateurs:

- Opérateurs arithmétiques:

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

- Opérateurs logiques:

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

- Opérateurs relationnels:

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

- Opérateur d'affectation:

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

3. **Expression:** 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:	identifiants a, b, c, constantes 4 et 2.
Opérateurs:	symboles $*$, $-$ et $/$

   - **Exemple d'expression booléenne:**

$$(x < 3.14)$$

Opérandes:	identifiants x et contante 3.14
Opérateurs:	symboles $<$

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

Exemple : boucle «pour»

Algorithme Exemple boucle «pour»

Données : n, un entier positif

1: $S \leftarrow 0$
2: **Pour** $i \leftarrow 1$ **à** n **faire**
3: $\quad S \leftarrow S + \cos(i\hat{}2)$
4: **Fin**

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»

Algorithme Exemple boucle «tant que»

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

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»

Algorithme Exemple instruction «si»

Données : *age*, un entier.

1: **Si** *age* $>=$ 64 **alors**
2: affiche('retraite')
3: **Sinon Si** *age* $>=$ 18 **alors**
4: affiche('majeur')
5: **Sinon**
6: affiche('mineur')
7: **Fin**

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
```

Plan

- **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:**
  - Simplifier et clarifier le problème.
  - 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.
- **Ecriture 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.

Plan

Exercices

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

Exercice 1 

Ecrire un algorithme permettant de calculer

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

Exercice 2 

Ecrire un algorithme permettant de calculer

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

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

Plan

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

    $$\sin,\ \cos,\ \exp,\ \cdots$$

  - les fonctions de gestion de fichiers,
  - les fonctions graphiques ...

Ecrire ses propres fonctions

- Avant d'écrire une fonction, voici les questions à se poser:
  1. Que doit-elle calculer/réaliser précisement (but)?
  2. Quelles sont ses données (avec leurs limitations)?
- Syntaxe: m arguments en entrée (données), 1 argument en sortie (résultat)

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

- Syntaxe: m arguments en entrée (données), n arguments en sortie (résultats)

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

Exemple

Ecrire une fonction permettant de résoudre

$$ax = b$$

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

Exemple

Ecrire une fonction permettant de résoudre

$$ax = b$$

- **But** :
  Enoncé insuffisament précis! On choisi 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 $ax = b$

Algorithme 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** $x \leftarrow \mathrm{REPD}(\ a, b\ )$
2: $x \leftarrow b/a$
3: **Fin**

- REPD est le nom de la fonction,
- Les paramètres d'entrée: a et b
- Le paramètre de sortie: x

Exemple d'utilisation: $z \leftarrow \mathrm{REPD}(2, 3) + 5$

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