Analyse Numérique I*

Sup'Galilée, Ingénieurs MACS, 1ère année / L3 MIM

François Cuvelier

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

2025/09/07

*Compilé le 2025/09/07 à 07:35:43

Divers

- Volume horaire : 11×3h cours - 11×3h 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, 14 novembre 2025 , (prévision, après 6 cours et 6 TDs)
  - P_2 partiel 2, 7 janvier 2026 ,
- 1 groupe de TD avec Mme Darbas.,
- Un serveur Discord dédié aux cours et aux TDs,
- Un polycopié,
- Une page web dédiée
  - cours : https://www.math.univ-paris13.fr/~cuvelier
- Mail: cuvelier@math.univ-paris13.fr.
- Pas présent sur l'ENT!

Objectifs

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

- Mathématiques
- Numériques
- Algorithmique

Bibliographie I

[GGK14, Dem12, Huc, Cia06, LT04, QSS07]

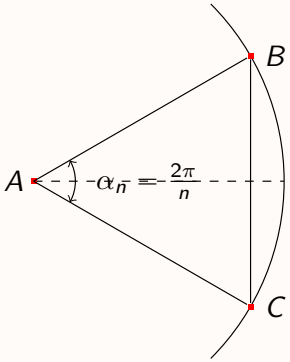
- P.G. Ciarlet, *Introduction à l'analyse numérique matricielle et à l'optimisation*, DUNOD, 2006.
- J.P. Demailly, *Analyse numérique et équations différentielles*, Grenoble Sciences, EDP Sciences, 2012.
- W. Gander, M.J. Gander, and F. Kwok, *Scientific computing : an introduction using maple and matlab*, Springer, Cham, 2014.
- T. Huckle, *Collection of software bugs* *http://wwwzenger.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.

Plan du cours

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 : $\pi \in]3 + \frac{10}{71}, 3 + \frac{1}{7}[$ avec 96 polygônes.
  $(3.14084507042254 < \pi \approx 3.14159265358979) < 3.14285714285714)$
- Algorithme polygônes inscrits ($r = 1$) :
  $\mathcal{A} = \lim_{n\to+\infty} \mathcal{A}_n = \lim_{n\to+\infty} \frac{n}{2}\sin(\alpha_n)$ et $\sin\frac{\alpha_n}{2} = \sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2}}$.

1: $s \leftarrow 1$, $n \leftarrow 4$ ▷ Initialisations
2: **Tantque** $s > 1e-10$ **faire** ▷ Arrêt si $s = \sin(\alpha)$ est petit
3: $s \leftarrow sqrt((1 - sqrt(1 - s * s))/2)$ ▷ 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**

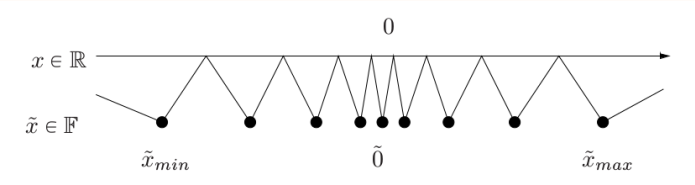
Un exemple : calcul approché de π

n	$\mathcal{A}_n$	$\lvert\mathcal{A}_n - \pi\rvert$	$\sin(\alpha_n)$
4	2.00000000000000	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	3.14159655370482	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

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

où $D \in \{0, 1, ..., b-1\}$ représente un chiffre et b la base.
Mantisse normalisée : 1er D (avant point) est non nul.



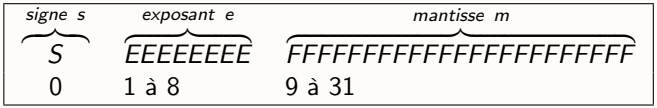
- **Précision machine :** plus petit nombre machine $\mathrm{eps} > 0$ tel que

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

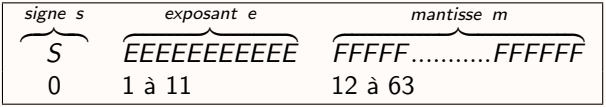
Matlab/Octave, langage C (double), ... : $\mathrm{eps} = 2.2204460492503131e-16$

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	$8 * 10^8 = 800Mo$
$10^5 \times 10^5$	double	8	$8 * 10^{10} = 80Go$

Problèmes arithmétiques en dimension finie

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; ($x > 0$) alors 1+x==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!

Erreurs d'annulation

Il faut être attentifs aux signes sur machine.

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

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

-

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

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

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!

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

1: $s \leftarrow 1$, $n \leftarrow 4$, ▷ Initialisations
2: **Tantque** $s > 1e-10$ **faire** ▷ Arrêt si $s = \sin(\alpha)$ est petit
3: $s \leftarrow s/sqrt(2*(1+sqrt(1-s*s)))$ ▷ 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	$\lvert A_n - \pi \rvert$	$\sin(\alpha_n)$
4	2.00000000000000	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 pousuivre ...

(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

Definition 1.1 : *Petit Robert 97*

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

Exemple 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 Résolution

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 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»

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 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»

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 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
```

Plan

Pour simplier l'écriture des algorithmes

- Identification des tableaux 1D à des vecteurs:
  - $\boldsymbol{x}$ tableau de réels à une dimension et de longueur $n \in \mathbb{N}^* \Rightarrow \boldsymbol{x} \in \mathbb{R}^n$,
  - $\boldsymbol{x}$ tableau d'entier à une dimension et de longueur $n \in \mathbb{N}^* \Rightarrow \boldsymbol{x} \in \mathbb{N}^n$,
  - ...

Choix indiçage: l'indice du premier élément d'un tableau/vecteur est 1

$$\forall i \in [\![1, n]\!],\ \boldsymbol{x}_i \text{ (not. mathématique)} \iff \boldsymbol{x}(i) \text{ (not. algorithmique)}$$

- Identification des tableaux 2D à des matrices:
  - $\mathbb{T}$ tableau de réels à m lignes et n colonnes $\Rightarrow \mathbb{T} \in \mathcal{M}_{m,n}(\mathbb{R})$,
  - $\mathbb{T}$ tableau de complexes à m lignes et n colonnes $\Rightarrow \mathbb{T} \in \mathcal{M}_{m,n}(\mathbb{C})$,
  - ...

$$\forall (i,j) \in [\![1, m]\!] \times [\![1, n]\!],\ \mathbb{T}_{i,j} \text{ (not. mathématique)} \iff \mathbb{T}(i,j) \text{ (not. algorithmique)}$$

Exercice 1

Soient $(a, b) \in \mathbb{R}^2$, $a < b$, $N \in \mathbb{N}$ et $\forall n \in [\![0, N]\!]$, $t^n = a + n\frac{b-a}{N}$. Ecrire une fonction DisReg permettant de retourner l'ensemble des $(t^n)_{n=0}^N$.

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 2 

Ecrire un algorithme permettant de calculer

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

Exercice 3 

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 Fonction

- 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 Fonction

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: $\quad x \leftarrow b/a$
3: **Fin Fonction**

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

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

Exemple

Ecrire une fonction retournant la somme des éléments d'un vecteur donné.

Données : $\boldsymbol{x}$: vecteur de $\mathbb{R}^n$.
Résultat : s : le réel tel que $s = \sum_{i=1}^{n} \boldsymbol{x}_i$.

```
1: Fonction s ← sum( x )
2:     s ← 0
3:     Pour i ← 1 à n faire                ▷ n implicite
4:         s ← s + x(i)
5:     Fin Pour
6: Fin Fonction
```

```
1: Fonction s ← sum( x )
2:     n ← length(x)                       ▷ n explicitement récupéré
3:     s ← 0
4:     Pour i ← 1 à n faire
5:         s ← s + x(i)
6:     Fin Pour
7: Fin Fonction
```

Exemple

Ecrire une fonction retournant le sinus de chacun des éléments d'un vecteur donné.

Données : $\boldsymbol{x}$: vecteur de $\mathbb{R}^n$.
Résultat : $\boldsymbol{y}$: vecteur de $\mathbb{R}^n$ tel que $\boldsymbol{y}_i = \sin(\boldsymbol{x}_i)$, $\forall i \in [\![1, n]\!]$.

```
1: Fonction y ← sinvec( x )
2:     Pour i ← 1 à n faire                ▷ n implicite
3:         y(i) ← sin(x(i))
4:     Fin Pour
5: Fin Fonction
```

```
1: Fonction y ← sinvec( x )
2:     n ← length(x)      ▷ n explicitement récupéré
3:     Pour i ← 1 à n faire
4:         y(i) ← sin(x(i))
5:     Fin Pour
6: Fin Fonction
```

Version écriture algorithmique simplifiée/vectorisée: $\boldsymbol{y} \leftarrow \sin(\boldsymbol{x})$.