Analyse Numérique et Algorithmique

Bouchaib FERRAHI

2023/2024

Présentation du module

- Filière: SMP
- Semestre:S3
- Code: M20
- intitulé: Analyse Numérique et Algorithmique
- Objectifs:
  - Maîtriser quelques outils Mathématiques utilisés en sciences appliquées
  - Apprendre à utiliser les démarches scientifiques face à des problèmes théoriques et expérimentaux variés
  - Comprendre les méthodes numériques permettant la résolution de quelques problèmes Mathématiques
  - S'initier à l'algorithmique
- Pré-requis:Analyse 1 et 2, Algèbre 1 et 2
- Volume horaire: 21H de cours + 21H de TD

Contact

Bouchaib FERRAHI

- Département de Mathématiques - Faculté des Sciences - Tétouan
- bferrahi@uae.ac.ma et ferrahi@yahoo.com
- http://moodle.uae.ac.ma/course/view.php?id=875
- www.ferrahi.ma
- www.facebook.com/bouchaib.ferrahi
- https://www.youtube.com/c/BouchaibFERRAHI
- www.linkedin.com/in/ferrahi

Plan

Introduction

- Pourquoi un cours de analyse numérique pour les étudiants de la filière SMP?
- Quelle approche pédagogique durant la situation actuelle?

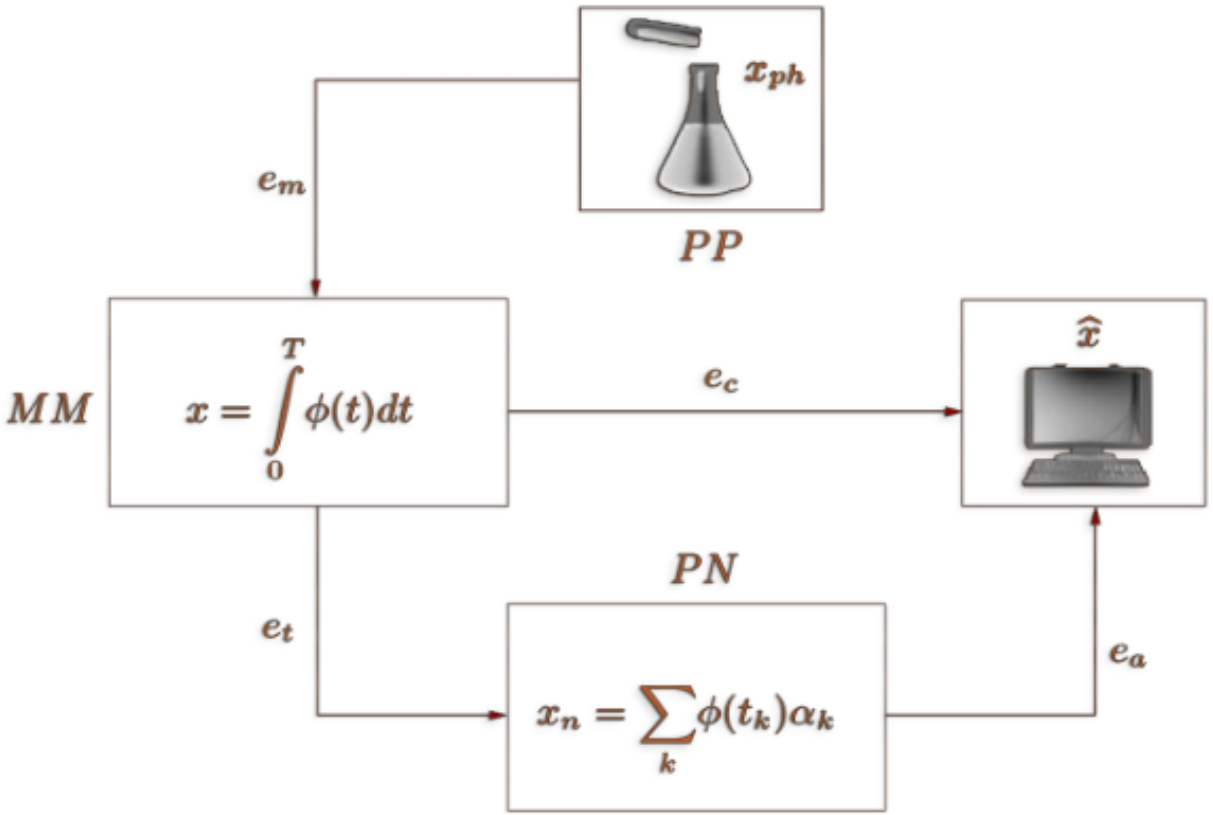


Figure 1.1 – Source : A.Quarteroni and all : Calcul Scientifique Cours, exercices corrigés et illustrations en Matlab et Octave.

Introduction

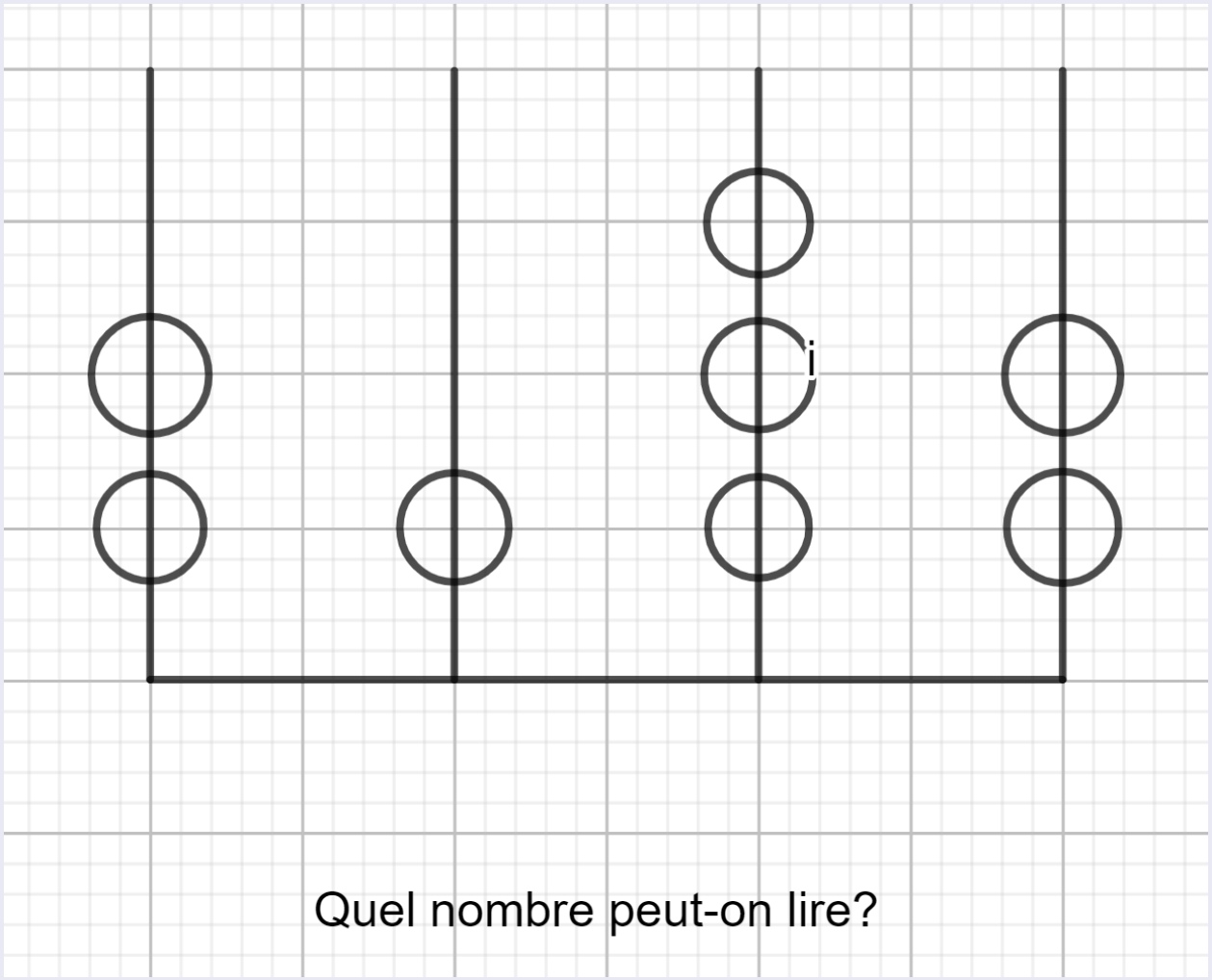
L'analyse numérique est une discipline Mathématique qui peut être considérée comme partagée en deux grands thèmes:

- Méthodes numériques: résolution de grands systèmes linéaires, intégration numérique, résolution numérique des équations différentielles, optimisation,...
- Approximation numérique des équations aux dérivées partielles, éléments finis, volumes finis, méthodes spectrales,$\cdots$

Introduction

Introduction à l'arithmétique des ordinateurs

Représentation des nombres



Réponse:

2132 = "Deux" milles + "un" cent + "trois" dizaines + "deux" unités

$$2132 = 2 \times 1000 + 1 \times 100 + 3 \times 10 + 2 \times 1$$

$$2132 = 2 \times 10^3 + 1 \times 10^2 + 3 \times 10^1 + 2 \times 10^0$$

Remarques:

- Écriture habituelle suivant la base décimale (base 10)
- Tout entier $a_n a_{n-1} \ldots . a_1 a_0$ s'écrit d'une manière naturelle:
  $a_n \times 10^n + a_{n-1} \times 10^{n-1} + \ldots . + a_1 \times 10^1 + a_0 \times 10^0$

D'une manière générale l'écriture suivant une base b (entier naturel) est définie par:

Définition:

Soit b un entier positif, tout entier naturel m admet une écriture unique suivant la base b donée par:

$$m = a_n \times b^n + a_{n-1} \times b^{n-1} + ... + a_1 \times b^1 + a_0 \times b^0$$

avec: $a_n \neq 0$ et pout tout i on a: $a_i = 0, 1, 2, ..., b-1$

NOTATION

- on note :

$$m = (a_n a_{n-1} \ldots . a_1 a_0)_b$$

ou

$$m = \overline{a_n a_{n-1} \ldots . a_1 a_0}^b$$

- Si $b = 10$ on écrit habituellement, lorsque aucune confusion n'est possible, $a_n a_{n-1} \ldots . a_1 a_0$ au lieu de

$$(a_n a_{n-1} \ldots . a_1 a_0)_{10}$$

ou

$$\overline{a_n a_{n-1} \ldots . a_1 a_0}^{10}$$

Exemples de bases

- Base binaire: $b = 2$, tout entier naturel s'écrit $\overline{a_n a_{n-1} \ldots . a_1 a_0}^2$ avec $a_i = 0$ ou $a_i = 1$, utilisée en Électronique et Informatique
- Base octale: $b = 8$, tout entier naturel s'écrit $\overline{a_n a_{n-1} \ldots . a_1 a_0}^8$ avec $a_i = 0, 1, \ldots, 7$, utilisée en Informatique
- Base hexadécimale, $b = 16$, tout entier naturel s'écrit $\overline{a_n a_{n-1} \ldots . a_1 a_0}^{16}$ avec $a_i = 0, 1, \ldots, 9$, $a = 10$, $b = 11, \ldots f = 15$, utilisée en Informatique

Exemple1:

- $1830 = (1830)_{10} = \overline{1830}^{10} = 1 \times 10^3 + 8 \times 10^2 + 3 \times 10^1 + 0 \times 10^0$;
- $1830 = (11100100110)_2 = \overline{11100100110}^2 = 1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^8 + 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$;
- $1830 = (726)_{16} = \overline{726}^{16} = 7 \times 16^2 + 2 \times 16^1 + 6 \times 16^0$.

Exemple 2:

Décimale (10)	Binaire (2)	Octale (8)	Hexadécimale (16)
0	00000	00	0
1	00001	01	1
2	00010	02	2
3	00011	03	3
4	00100	04	4
5	00101	05	5
6	00110	06	6
7	00111	07	7
8	01000	10	8
9	01001	11	9
10	01010	12	A

Exemple 3:

Décimale (10)	Binaire (2)	Octale (8)	Hexadécimale (16)
11	01011	13	B
12	01100	14	C
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10

Remarque:

Il est nécessaire de préciser la base utilisée, par exemple:

- 1101 en base $b = 10$: $(1101)_{10} = 1101$
- 1101 en base $b = 8$: $(1101)_8 = 577$
- 1101 en base $b = 2$: $(1101)_2 = 13$

Conversion des écritures entre deux bases:

base B $\rightarrow$ base décimale:

La conversion d'une écriture $(a_n a_{n-1} ... a_1 a_0)_b$ en base b quelque à une écriture en base décimale s'obtient en calculant la somme:

$$a_n \times b^n + a_{n-1} \times b^{n-1} + ... + a_1 \times b^1 + a_0 \times b^0$$

Exemples:

$(11001)_2 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^0 = 16 + 8 + 1 = 25$
$(1201)_3 = 1 \times 3^3 + 2 \times 3^2 + 1 \times 3^0 = 27 + 18 + 1 = 46$
$(7A9E)_{16} = 7 \times 16^3 + 10 \times 16^2 + 9 \times 16^1 + 14 \times 16^0 =$
$7 \times 4096 + 10 \times 256 + 9 \times 16 + 14 = 28672 + 2560 + 144 + 14 = 31390$

base décimale $\rightarrow$ base B:

La conversion s'obtient en effectuant des divisions successives du nombre, écrit en base décimale, par b et en classant les restes dans le sens inverse: $m = bq_0 + r_0$, $q_0 = bq_1 + r_1$, ...,$q_{n-2} = bq_{n-1} + r_{n-1}$, $q_{n-1} = b \times 0 + r_n$, avec $q_{n-1} = r_n < b$ et $q_n = 0$, alors:

$$m = (r_n r_{n-1} .... r_1 r_0)_b$$

Exemples:

- $41 = 2 \times 20 + 1$, $20 = 2 \times 10 + 0$, $10 = 2 \times 5 + 0$, $5 = 2 \times 2 + 1$, $2 = 2 \times 1 + 0$ et $1 = 2 \times 0 + 1$ donc: $(41)_{10} = (101001)_2$
- $1830 = 16 \times 114 + 6$, $114 = 16 \times 7 + 2$, $7 = 16 \times 0 + 7$, donc: $(1830)_{10} = (726)_{16}$
- $479 = 7 \times 68 + 3$, $68 = 7 \times 9 + 5$, $9 = 7 \times 1 + 2$, $1 = 7 \times 0 + 1$, donc: $479_{10} = (1253)_7$

Base $b \to$ base b':

La conversion s'obtient en passant par la base décimale:
Base $b \to$ base 10 $\to$ base b'.

Cas important: base puissance de l'autre ($b \to b^k$)

On découpe la représentation de m en base b en tranches de k chiffre, en commençant par la droite et en rajoutant des 0 à gauche si le nombre de chiffres de m n'est pas un multiple de k. Chaque tranche de k chiffres est alors transformée en un chiffre en base b^k. Ces chiffres, écrits dans cet ordre, constituent l'écriture de m en base b^k.

Exemples:

- Écriture de 1022102_3 en base $9 = 3^2$, on coupe l'écriture à des tranches de 2 en commençant par la droite et en ajoutant 0 à gauche:

$$\underbrace{\mathbf{0}1}_{3}\ \underbrace{02}_{3}\ \underbrace{21}_{3}\ \underbrace{02}_{3} = 1_9.2_9.7_9.2_9 = (1272)_9$$

- Écriture de 10101101110_2 en base $16 = 2^4$:

$$\underbrace{\mathbf{0}101}_{2}\ \underbrace{0110}_{2}\ \underbrace{1110}_{2} = 5_{16}.6_{16}.E_{16} = (56E)_{16}$$

Cas important: base puissance de l'autre ($b^k \to b$)

On exprime chaque chiffre en base b^k comme un nombre écrit en base b sur k chiffres, en rajoutant des 0 (à gauche) si l'écriture du nombre obtenu comporte moins de k chiffres en base b.

Exemples:

- Écriture de $1A2F_{16=2^4}$ en base $b = 2$:

$$1_{16}.A_{16}.2_{16}.F_{16} = \underbrace{\mathbf{000}1}_{2}.\underbrace{1010}_{2}.\underbrace{\mathbf{00}10}_{2}.\underbrace{1111}_{2}$$

$$= (0001101000101111)_2 = (1101000101111)_2$$

- Écriture de $156_{8=2^3}$ en base $b = 2$:

$$1_8.5_8.6_8 = \underbrace{\mathbf{00}1}_{2}.\underbrace{101}_{2}.\underbrace{110}_{2} = (001101110)_2 = (1101110)_2$$

Opérations habituelles dans une base b

Opérations habituelles dans une base b

Les opérations habituelles (somme, soustraction et multiplication) peuvent être effectuées dans une base b en prenant en considération que $0 \leq a_i < b$

Opérations habituelles dans une base b

Opérations habituelles dans une base b

Les opérations habituelles (somme, soustraction et multiplication) peuvent être effectuées dans une base b en prenant en considération que $0 \leq a_i < b$

Opérations binaires

Addition: $0+0=0$, $1+0=0+1=1$ et $1+1=0$ avec une retenue de 1.

Exemple:

Effectuons en binaire l'addition suivante: $(34)_{10} + (27)_{10}$

Opérations habituelles dans une base b

Opérations habituelles dans une base b

Les opérations habituelles (somme, soustraction et multiplication) peuvent être effectuées dans une base b en prenant en considération que $0 \leq a_i < b$

Opérations binaires

Addition: $0+0=0$, $1+0=0+1=1$ et $1+1=0$ avec une retenue de 1.

Exemple:

Effectuons en binaire l'addition suivante: $(34)_{10}+(27)_{10}$

$$(34)_{10}+(27)_{10}=(100010)_2+(11011)_2=(111101)_2$$

Opérations habituelles dans une base b

Opérations habituelles dans une base b

Les opérations habituelles (somme, soustraction et multiplication) peuvent être effectuées dans une base b en prenant en considération que $0 \leq a_i < b$

Opérations binaires

Addition: $0+0=0$, $1+0=0+1=1$ et $1+1=0$ avec une retenue de 1.

Exemple:

Effectuons en binaire l'addition suivante: $(34)_{10}+(27)_{10}$

$$(34)_{10}+(27)_{10}=(100010)_2+(11011)_2=(111101)_2$$

Vérification: $(34)_{10}+(27)_{10}=(61)_{10}$ et $(61)_{10}=(111101)_2$

Représentation des nombres en machine

Les informations traitées par un ordinateur sont de différents types (nombres, instructions, images, vidéo...) mais elles sont toujours représentées sous un format binaire. Seul le codage changera suivant les différents types de données à traiter. Elles sont représentées physiquement par 2 niveaux de tensions différents. En binaire, une information élémentaire est appelé bit et ne peut prendre que deux valeurs différentes : 0 ou 1 . Une information plus complexe sera codée sur plusieurs bit. On appelle cet ensemble un mot. Un mot de 8 bits est appelé un octet. Pour représenter les nombres en machine on choisira une base b et une norme de codage. Les ordinateurs emploient en général trois bases :

- la base 2 ou binaire,
- la base 8 ou octale et
- la base 16 ou hexadécimale.

Représentation des nombres en machine

Dans la représentation binaire, les chiffres se réduisent aux deux symboles 0 et 1 , appelés bits (de l'anglais binary digits). En hexadécimal les symboles utilisés pour la représentation des chiffres sont $0, 1, \ldots, 9, A, B, C, D, E, F$.
Toute opération qu'effectue un ordinateur ("opération machine") est entachée par des erreurs d'arrondi. Elles sont dues au fait qu'on ne peut représenter dans un ordinateur qu'un sous-ensemble fini de l'ensemble des nombres réels.

Représentation des nombres en machine - Les nombres entiers

Sur machine le terme nombres entiers désigne l'ensemble **Z**. Ils sont exprimés par des chiffres pouvant prendre deux valeurs 0 ou 1.
A chaque chiffre est affecté un poids exprimé en puissance de 2
- Exemples (Écriture suivant différentes bases)
$10 : (2019)_{10} = 2.10^3 + 0.10^2 + 1.10^1 + 9.10^0$

$$(101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (5)_{10}$$

$$(39)_{10} = 32 + 4 + 2 + 1 = 2^5 + 2^2 + 2^1 + 2^0 = (100111)_2$$

Le nombre représentable dépend du nombre d'octets utilisés.

Représentation des nombres en machine - Les nombres entiers

Le nombre représentable dépend du nombre d'octets utilisés :

- bit $\rightarrow$

Représentation des nombres en machine - Les nombres entiers

Le nombre représentable dépend du nombre d'octets utilisés :

- bit $\rightarrow$ 2 états possibles: 0 ou 1
- octet $\rightarrow$

Représentation des nombres en machine - Les nombres entiers

Le nombre représentable dépend du nombre d'octets utilisés :

- bit $\rightarrow$ 2 états possibles: 0 ou 1
- octet $\rightarrow$
  $2\times2\times2\times2\times2\times2\times2\times2 = 2^8 = 256$ états possibles
- 2 octets $\rightarrow$

Représentation des nombres en machine - Les nombres entiers

Le nombre représentable dépend du nombre d'octets utilisés :

- bit $\rightarrow$ 2 états possibles: 0 ou 1
- octet $\rightarrow$
  $2\times2\times2\times2\times2\times2\times2\times2 = 2^8 = 256$ états possibles
- 2 octets $\rightarrow 256 \times 256 = 65536$ états possibles
- 3 octets $\rightarrow$

Représentation des nombres en machine - Les nombres entiers

Le nombre représentable dépend du nombre d'octets utilisés :

- bit $\rightarrow$ 2 états possibles: 0 ou 1
- octet $\rightarrow$
  $2\times2\times2\times2\times2\times2\times2\times2 = 2^8 = 256$ états possibles
- 2 octets $\rightarrow$ $256 \times 256 = 65536$ états possibles
- 3 octets $\rightarrow$
  $256 \times 256 \times 256 = 16777216$ états possibles

- avec deux (2) octets, on peut représenter les entiers compris entre -32768 et 32767
- avec quatre (4) octets on peut représenter les entiers compris entre -2147483648 et 2147483647

Représentation des nombres en machine - Les réels à virgule fixe

Soit une base fixée $b \in \mathbb{N}$ avec $b \geqslant 2$, et soit x un nombre réel comportant un nombre fini de chiffres a_k avec $0 \leqslant a_k < b$. On se propose de travailler avec un ordinateur disposant de N cases mémoires pour représenter x. Une case mémoire sera réserver pour le signe, $N - k - 1$ pour les chiffres entiers et k pour les chiffres situés après la virgule, de sorte que :

$$x = (-1)^s \times [a_N a_{N-1} \dots a_k, a_{k-1} \dots a_0], \quad a_N \neq 0$$

Qu'on peut écrire :

$$x = (-1)^s \times b^{-k} \times \left(\sum_{j=0}^{N} a_j b^j \right).$$

où s dépend du signe de $\times$($s = 0$ si $\times$ est positif, 1 si $\times$ est négatif).

Représentation des nombres en machine - Les réels à virgule fixe

L'ensemble des nombres de ce type est appelé système à virgule fixe.

- Exemple :

- $(-526,73)_{10} = (-1)^1 \times 10^{-2} \times (3 \times 10^0 + 7 \times 10^1 + 6 \times 10^2 + 2 \times 10^3 + 5 \times 10^4)$.
- $(425,31)_6 = (-1)^0 \times 6^{-2} \times (1 \times 6^0 + 3 \times 6^1 + 5 \times 6^2 + 2 \times 6^3 + 4 \times 6^4)$.
  L'utilisation de la virgule fixe limite considérablement les valeurs minimales et maximales des nombres pouvant être représentés par l'ordinateur, à moins qu'un très grand nombre N de cases mémoires ne soit employé.

Représentation des nombres en machine - Les réels à virgule flottante

Soit x un nombre réel non nul, sa représentation en virgule flottante est donnée par:
$x = (-1)^s \times (0, a_1 a_2 \ldots\ldots a_t) \times b^e = (-1)^s \times m \times b^{e-t}$ avec

- $t \in \mathbb{N}$ est le nombre de chiffres significatifs a_i (avec $0 \leq a_i < b$),
- $m = a_1 a_2 \ldots a_t$ un entier vérifiant $0 \leq m \leq b^{t-1}$ appelé **mantisse**,
- e un entier appelé **exposant**.

L'exposant ne peut varier que dans un intervalle fini de valeurs admissibles : posons

$$L = e_{min} \leq e \leq U = e_{max}.$$

Représentation des nombres en machine - Les réels à virgule flottante

Les N cases mémoires sont à présent réparties ainsi :

- une case pour le signe,
- t cases pour les chiffres significatifs et
- $N - t - 1$ cases pour les chiffres de l'exposant.

Représentation des nombres en machine - Les réels à virgule flottante

Les N cases mémoires sont à présent réparties ainsi :

- une case pour le signe,
- t cases pour les chiffres significatifs et
- $N - t - 1$ cases pour les chiffres de l'exposant.

Remarque: Le nombre zéro a une représentation à part.

Représentation des nombres en machine - Les réels à virgule flottante

Il y a typiquement sur un ordinateur deux formats disponibles pour les nombres à virgule flottante : les représentations en simple et en double précision. Dans le cas de la représentation binaire, ces formats sont codés dans les versions standards avec $N = 32$ bits (simple précision)

Dans le standard *IEEE* 754 utilisé par Matlab, on a $b = 2$ et:

- en simple précision : $t = 24$, $e_{min} = -125$, $e_{max} = 128$

Représentation des nombres en machine - Les réels à virgule flottante

et avec $N = 64$ bits (double précision)

Dans le standard *IEEE* 754 utilisé par Matlab, on a $b = 2$ et:

- en double précision : $t = 53$, $e_{min} = -1021$, $e_{max} = 1024$

Représentation des nombres en machine - Arrondi d'un nombre réel en représentation machine

Sur tout ordinateur, seul le sous-ensemble $\mathcal{F} \subset \mathbb{R}$ soit effectivement disponible. Ainsi un $x \in \mathbb{R}$, sera remplacer par son représentant dans $\mathcal{F}$ qu'on notera $fl(x)$. Par exemple si $x = \frac{1}{3} = 0,33333333\ldots$ et $t = 4$ alors $fl(x) = 0,3333$.
D'une manière générale pour calculer $fl(x)$ on procède soit par **Arrondi** soit **Troncature** :

Représentation des nombres en machine - Arrondi d'un nombre réel en représentation machine

Arrondi

Soit $x \in \mathbb{R}$ en notation normalisée, en **arrondi** $fl(x)$ est défini par :

$$fl(x) = (-1)^s \times (0.a_1 a_2 \dots \tilde{a}_t) \times b^e, \quad \tilde{a}_t = \begin{cases} a_t & \text{si } a_{t+1} < \dfrac{b}{2} \\ a_t + 1 & \text{si } a_{t+1} \geq \dfrac{b}{2}. \end{cases}$$

Représentation des nombres en machine - Arrondi d'un nombre réel en représentation machine

Arrondi

Soit $x \in \mathbb{R}$ en notation normalisée, en **arrondi** $fl(x)$ est défini par :

$$fl(x) = (-1)^s \times (0.a_1 a_2 \dots \tilde{a}_t) \times b^e, \quad \tilde{a}_t = \begin{cases} a_t & \text{si } a_{t+1} < \dfrac{b}{2} \\ a_t + 1 & \text{si } a_{t+1} \geq \dfrac{b}{2}. \end{cases}$$

Exemple

Avec 3 chiffres pour $x = 0,8573$ et $y = 0,8576$ on a en arrondi

Représentation des nombres en machine - Arrondi d'un nombre réel en représentation machine

Arrondi

Soit $x \in \mathbb{R}$ en notation normalisée, en **arrondi** $fl(x)$ est défini par :

$$fl(x) = (-1)^s \times (0.a_1 a_2 \dots \tilde{a}_t) \times b^e, \quad \tilde{a}_t = \begin{cases} a_t & \text{si } a_{t+1} < \dfrac{b}{2} \\ a_t + 1 & \text{si } a_{t+1} \geq \dfrac{b}{2}. \end{cases}$$

Exemple

Avec 3 chiffres pour $x = 0,8573$ et $y = 0,8576$ on a en arrondi

- $fl(x) = 0,857$
- $fl(y) = 0,858.$

Représentation des nombres en machine - Arrondi d'un nombre réel en représentation machine

Troncature Dans l'approche **troncature** on prendrait $\tilde{a_t} = a_t$.

$$fl(x) = (-1)^s \times (0.a_1 a_2 \dots a_t) \times b^e.$$

Exemple

Avec 3 chiffres, pour $x = 0,8573$ et $y = 0,8576$ on a en troncature :

Représentation des nombres en machine - Arrondi d'un nombre réel en représentation machine

Troncature Dans l'approche **troncature** on prendrait $\tilde{a_t} = a_t$.

$$fl(x) = (-1)^s \times (0.a_1 a_2 \dots a_t) \times b^e.$$

Exemple

Avec 3 chiffres, pour $x = 0,8573$ et $y = 0,8576$ on a en troncature :

- $fl(x) = 0,857$
- $fl(y) = 0,857$.

Représentation des nombres en machine - Estimation d'erreurs

Soit x un réel et $\overline{x}$ une valeur approchée de x.

- L'erreur absolue $E = E(x)$ est défini par $E = |x - \overline{x}|$.
- L'erreur relative est $E_{rel} = \dfrac{|x - \overline{x}|}{|x|}$.

Représentation des nombres en machine - Estimation d'erreurs

Soit x un réel. Si x est dans $\mathcal{F}$, alors

$$fl(x) = x(1+\delta) \quad \text{avec } |\delta| \leq u,$$

où

$$u = \frac{1}{2}b^{1-t} = \frac{1}{2}\varepsilon_M$$

Le nombre u est l'erreur relative maximale que l'ordinateur peut commettre en représentant un nombre réel en arithmétique finie. Pour cette raison, on l'appelle **unité d'arrondi.**
On déduit immédiatement la majoration suivante de l'erreur relative :

$$E_{rel} = \frac{|x - fl(x)|}{|x|} \leq u$$

Représentation des nombres en machine - Estimation d'erreurs

Exemple:

En base $b = 10$, $x = 1/15 = 0,066666666.....$

1. Dans le cas d'une **représentation tronquée** nous aurons, pour $s = 5$, $fl(x) = 0,66666 \times 10^{-1}$.
   - L'erreur absolue $x - fl(x)$ est de 6×10^{-7}.
   - L'erreur relative est de l'ordre de 10^{-5}
   - Dans une représentation tronquée à s chiffres, l'erreur relative maximale est de l'ordre de 10^{-s}
2. Dans le cas d'une **représentation arrondie** nous aurons $fl(x) = 0,66667 \times 10^{-1}$
   - L'erreur absolue serait alors $3,333 \times 10^{-7}$.
   - L'erreur relative serait 5×10^{-6}
   - En général, l'erreur relative dans une représentation arrondie à s chiffres est de $5 \times 10^{-(s+1)}$ soit la moitié de celle d'une représentation tronquée.

Représentation des nombres en machine - Estimation d'erreurs

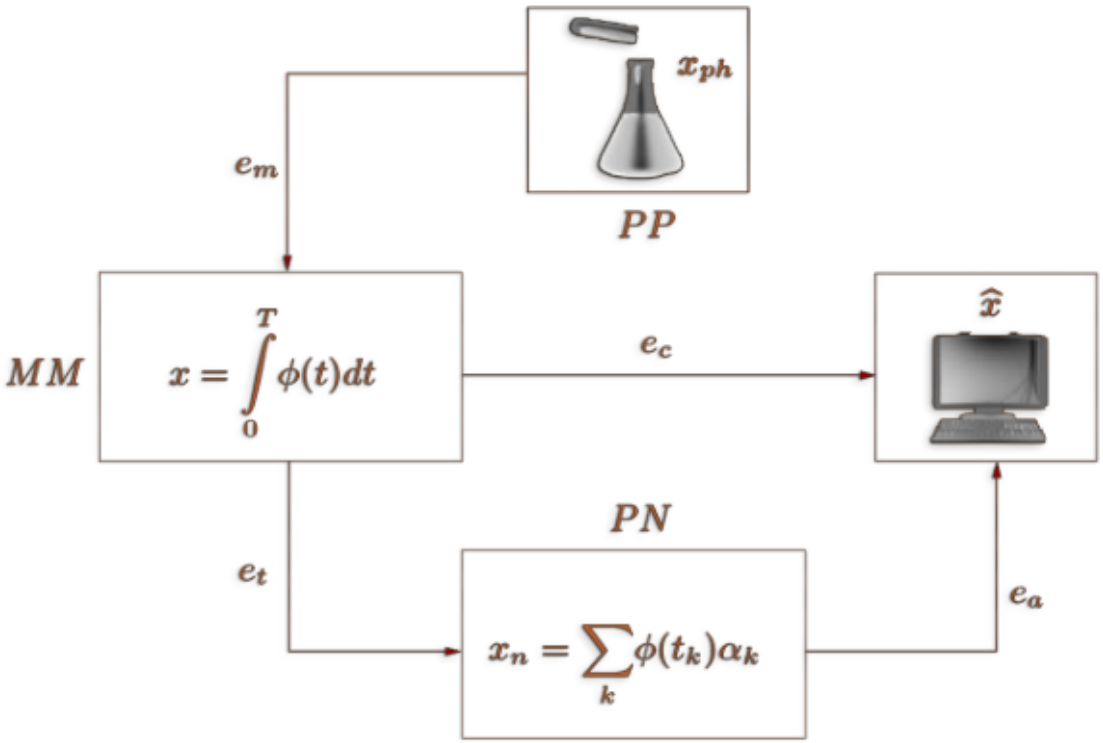


Figure 1.1 – Source : A.Quarteroni and all : Calcul Scientifique Cours, exercices corrigés et illustrations en Matlab et Octave.