CHAPITRE N°2 :
C2 - codage

PLAN :

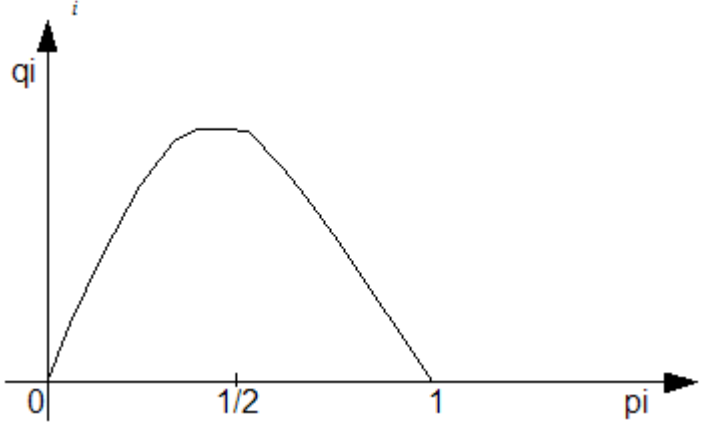
I) Notion d'information

information :
- signal (quelque chose de physique) qui dépend du temps et de l'espace) (concret/physique)
- sens sémantique associé au signal (abstrait)
relation entre les 2 : un code.

ici, le signal est en bits (010...)
- le codage remonte à Claude Shannon : théorie de l'information.
- symbole : . _ ^ : code Morse : les lettres qu'on utilisent souvent : signal cours, et inversement.
- Shannon a étudié la qualité d'information par le bit (binary digit)

$$Q_i = -p_i \log_2(p_i)$$

$$Q = \sum_i -p_i \log_2(p_i)$$



II) Codage des nombres entiers

1) Actions de base pour les entiers naturels

- méthode de la chaussette pour avoir un chiffre en binaire (on commence par la fin des restes pour écrire le chiffre par la gauche)
- pour chaque ordi : machine à 8 bits.
- on divise les mots en groupe de 4 bits
- SAVOIR les 15 premiers chiffres en binaire
- pour multiplier par 2 en base 2 : on décale vers la gauche le nombre, et on met un zéro dans la case vide à droite.

---

2) Arithmétique fixée pour les entiers relatifs

si X = r(x) en binaire et Y=r(y)
X # Y = r(x # y) n'est pas toujours vraie
En codage binaire naturels, si "x" a N bits, alors $x \in [0 ; 2^N-1]$.

---

3) Codage des entiers relatifs

a) Utilisation d'un bit supplémentaire

nombre positif -> 0
négatif -> 1
Le premier bit sur les 4 sera soit 0, soit 1, en fonction du signe du nombre
On aimerait r(x#y) = r(x) # r(y), mais.... pas vraie.

---

b) Codage biaisé

soit b > 0, b = biais.

si x > -b
r(x) = β(x#b) > 0
r(x) # r(y) = x#b#y#b = ... = r(x#y) # b
d'où le résultat :

r(x#y) = r(x) # r(y) - b
si x représenté par N bits, $r(x) \in [0 ; 2b]$
=> $b = 2^{N-1}-1$
=> $x \in [-2^{N-1}+1 ; 2^{N-1}]$.

---

c) Code complément à 2

- complément à 10 :
1502 -> on complémente à 9 -> 8497 -> on ajoute un -> 8498
convention : on s'arrête à 8 :
1000 -> -8
...
1111 -> -1
0000 -> 0
0001 -> 1
... jusqu'à 7 inclu
=> le quatrième chiffre est le bit de signe.

- complément à 2 :
r(-x) = - Complément à 2 (x) = $C_1(x)\#1 = \overline{r(x)}\#1$.

r(x#y) = r(x) # r(y) est vraie si on reste dans les bornes.
r(-x) # r(x) = r(0) = 0
dans le complément à 2 : $r(x_{max})$ = 0111...11 (du rang n à 0) = $2^{N-1}-1$

$r(x_{min})$ = 10...0 = -2^{N-1} (démo : on passe par le complément à 2 de x_{min})

=> $x \in [-2^{N-1} ; 2^{N-1}-1]$.

- en code binaire :

x	r(x)
x positif	β(x)
x strict négatif	$C_2(\beta(x))$

code complément à 2 différent de complément à 2 : en codage comp à 2 : r(x#y) = r(x)#r(y) vrai que si on ne dépasse pas les bornes (ie que la somme ne dépasse pas les bornes)
En binaire : 2 codages possibles :
entier -> r(x)=β(x)
relatif -> voir tableau.

III) Codage des nombres fractionnels

x ∈ [0;1]
car 0 x 2^0 + 1 x 2^-1 + 1 x 2^-2 = 0,11.

1) Codage des nombres à virgule fixe

(fixed point nombre)
(car le point sert de virgule en anglais)
10,11 (nombre décimal en binaire)
=> 1 x 2^1 + 0 x 2^0 + 1 x 2^-1 + 1 x 2^-2 = 0,11

---

2) Codage des nombres à virgule flottante

Avogadro, charge d'un électron, masse d'un électron (avec puissance de 10 et unités) beaucoup de chiffres
en général, on utilise la notation scientifique décimale :
$(-1)^s \times m \times 10^e$
m = mantisse et e = exposant et s = indication de signe
m ∈ [1;10[(mantisse normalisé à 10) ou à [1;1000[(mantisse normalisé à 1000)
e indique une position de virgule (la virgule n'est pas toujours posée au même endroit, d'où le terme de virgule flottante)

on considère la même chose en binaire :
soit x réel, **$x = (-1)^s \times m \times 2^e$**
m ∈ [1;2[(mantisse normalisée à 2).
on utilise la norme IEEE 754, qui spécifie les notations et codages
on ne pourra pas représenter n'importe quoi avec les flottants (on ferra toujours une petite erreur entre le vrai réel, et sa représentation (ce qu'on peut représenter)).

FLOPS : floating point operation (c'est la représentation des nombres flottants) par second.
On utilise en général les giga : G Flops (un microprocesseur puissant), ou les terra : T

Flops (un super calculateur d'un pays).

Norme IEEE 754
On représente les nbres réels par des nbres flottans comme :
un bit pour le signe, μ pour m, et ε pour l'exposant

S (code pour le signe s)	M (code pour la mantisse m)	E (code pour l'exposant e)

Codages :
- signe :

S	Signe s	Signe du réel
1	-	négatif
0	+	positif

- exposant :
$e \in \mathbb{Z}$ (=> codage biaisé)
$E = B(e + \beta)$ avec $\beta = 2^{(\varepsilon-1)} - 1$.
- mantisse :
$m \in [1;2[$
minimum : 1,0000...0
maximum : 1,1111...1
le premier chiffre est toujours « 1 », donc on ne le met pas : on ne représente que la partie fractionnelle

N	σ	μ	ε	
32	1	23	8	Simple précision
64	1	52	11	Double précision
24	1	16	8	
80	1			Précision étendue

FPU : floating point unit (liée à la précision étendue pour faire les opérations sur des grands nombres...).

---

3) Précision

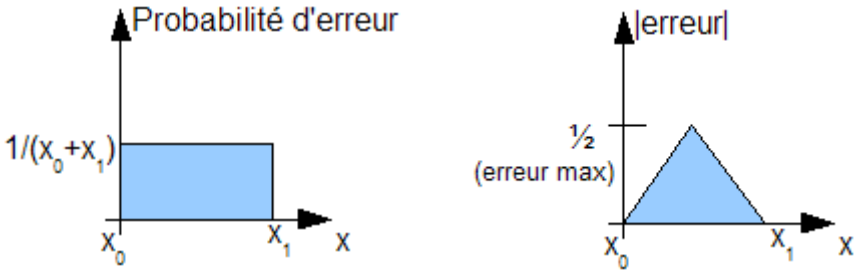
nbre flottant pour représenter un nbre réel.
Un nbre réel est encadré par 2 nombres dénombrables.
Donc le représentant de x réel sera celui qui est le plus proche (avec le moins d'erreur possible).
On appelle ça un arrondi.

$$erreur\ absolue = \frac{intervalle\ entre\ 2\ nombres\ dénombrables\ succecifs}{2}$$

$$erreur\ relative = \frac{erreur\ absolue}{|x|}$$

$$erreur\ relative\ maximum = \frac{|écart\ absolu\ maximal|}{2 \times |x|} = \ldots = 2^{-\mu-1}$$

erreur relative moyenne :



en moyenne : on calcule l'intégrale du triangle :

$$erreur\ relative\ moyenne = \frac{erreur\ relative\ max}{2} = \frac{1}{4} \times \frac{écart}{|x|} = 2^{-\mu-2}$$

l'erreur dépend de la position de x dans l'intervalle.

---

4) Valeurs maximales et minimales

valeur min représentative > 0

$x_{min} = m_{min} \times 2^{e_{min}}$

e_{min} <-> E_{min} = 00000..000

donc $e_{min} = E_{min} - \beta = 0 - (2^{\varepsilon-1} - 1)$

d'où $x_{min} = 1 \times 2^{(2^{(\varepsilon-1)} - 1)} + 1)$

par définition
zéro = codé par : m = 1,0 => M=0
et e = le plus petit => E=0
et s qui dépend du signe.

E_{min} = 00000..001

donc $e_{min} = E_{min} - \beta = 1 - (2^{\varepsilon-1} - 1)$

d'où $x_{min} = 1 \times 2^{(2^{(\varepsilon-1)} + 2)}$.

écart au voisinage de 0 :

$\Delta = (m_1 - m_0).2^e$ (e_{min} -> E=1)

$= 2^{-\mu} \times 2^{(2^{(\varepsilon-1)} + 2)}$

$$\frac{écart}{valeur\ min} = \frac{2^{-\mu} \times 2^{-2^{\varepsilon-1}+2}}{2^{-2^{\varepsilon-1}+2}} = 2^{-\mu}$$

dénormalisation :
matisse = 0,..
dès le moment où on a de très (E=0) petits nombre (en dessous de x_{min}) (=tiny number),
on a des problèmes car bit unité de mantisse = 0 au lieu de 1.
Si E=0, on a dénormalisation => bit unité=0 (x = 0,M etc... et E=0)
pour 0 -> car particulier des tiny number, M = 0.

sur R étendu (avec les infinis) :
il faut mettre un code pour l'infini.
E = 1...1 et M = 0...0 => on a l'infini.

Le plus grand nombre avant l'infini est : $m_{max} \times 2^{(E_{max} - \beta)}$

Emax = 11111..110
Il reste des autres possibilités, qu'on utilise pour construire des tableaux et ensuite détecter les erreurs.