



# INF1005A: Programmation procedurale

## Chapitre 7: Représentation interne des données



# Agenda

- Représentation de nombres
- Conversions, fonctions MATLAB
- Entiers: non signés, signés
- Nombres réels, norme IEEE754
- Les caractères



# Représentation de nombres – notation positionnelle base 10

La valeur de chacun des chiffres formant le nombre est pondérée par la valeur de la base choisie élevée à une puissance correspondant à la position occupée par le chiffre

$$4752 = 4 \times 10^3 + 7 \times 10^2 + 5 \times 10^1 + 2 \times 10^0$$



# Représentation de nombres – notation positionnelle base 2

$$10111_{\text{base } 2} = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 23_{\text{base } 10}$$

Bit le plus significatif: bit ayant la pondération la plus élevée (MSB)

Bit le moins significatif: bit ayant la pondération la moins élevée (LSB)



# Agenda



## Représentation de nombres

- Conversions, fonctions MATLAB
- Entiers: non signés, signés
- Nombres réels, norme IEEE754
- Les caractères

### Conversion – décimale – binaire

Chaque valeur décimale d'un système basé sur des mots (octets) de 8 bits sera une combinaison des 8 poids binaires suivants: 128, 64, 32, 16, 8, 4, 2, 1

↑  
MSB

↑  
LSB

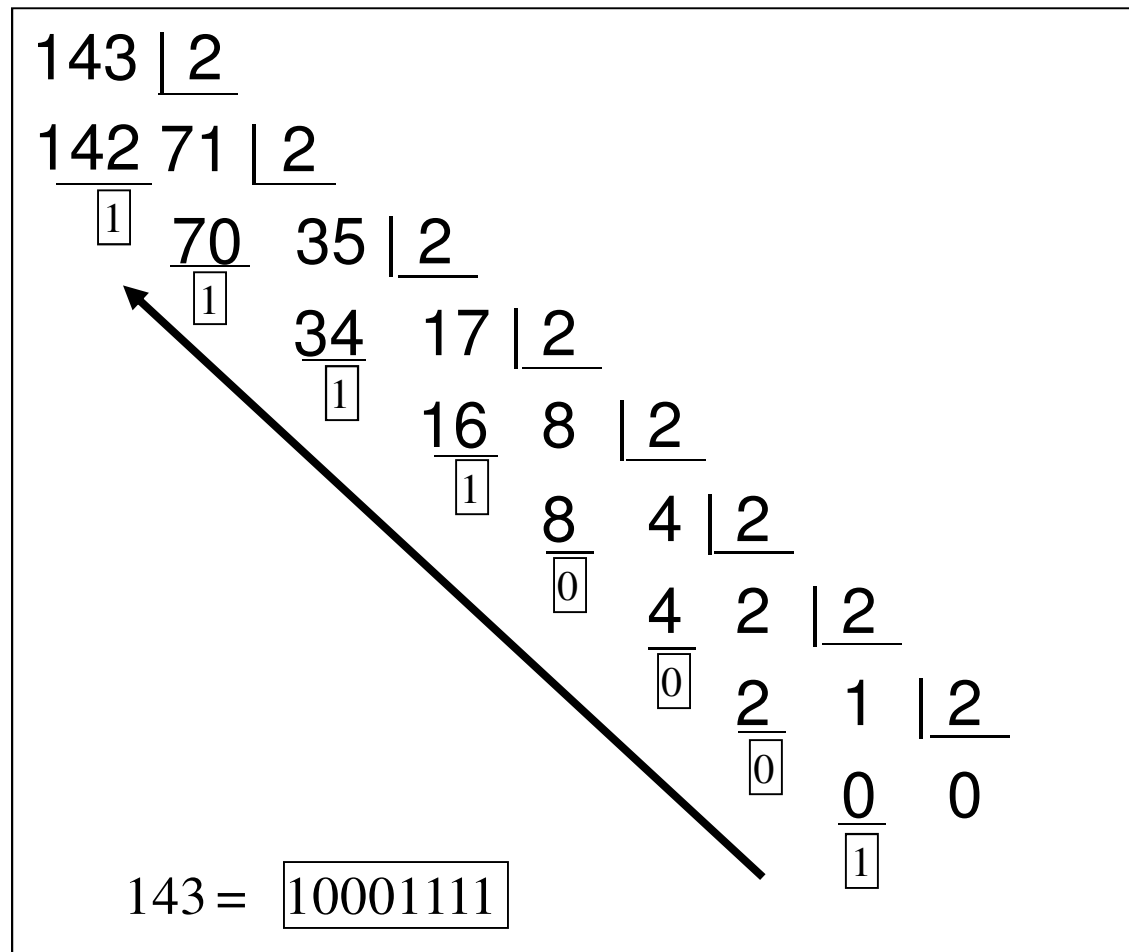
Soit par exemple **143**:

143 est composé de	<b>1</b> fois 128	(MSB)
143 - 128 = 15 est composé de	<b>1</b> fois 8	
15 - 8 = 7 est composé de	<b>1</b> fois 4	
7 - 4 = 3 est composé de	<b>1</b> fois 2	
3 - 2 = 1 est composé de	1 fois 1	(LSB)

Donc  $(1\ 0\ 0\ 0\ 1\ 1\ 1\ 1)_2$  donne 143 décimal



### Conversion – décimale – binaire





### Conversion – **base octale (base 8)**

Chiffres entre 0 et 7

Conversion de la base 2 à la base 8

- Regrouper les bits par paquet de trois à partir du bit le moins significatif (  $8 = 2^3$  )
- Calculer la valeur associée à chaque paquet

$$1010100111 = \begin{matrix} 001 & 010 & 100 & 111 \\ 1 & 2 & 4 & 7 \end{matrix} = 1247_{\text{base } 8}$$





### Conversion – ~~base hexadécimale~~ (base 8)

Comment convertir du octal au décimal?

$$\text{Rép.: } 1 * 8^3 + 2 * 8^2 + 4 * 8^1 + 7 * 8^0 = 679$$

Pour vérifier que le résultat précédent est bon, il s'agit de convertir binaire à décimal et s'assurer que le résultat est le même:

$$1010100111 = 512 + 256 + 64 + 7 = 679$$

### Conversion – **base hexadécimale (base 16)**

Chiffres de 0 à 9 lettres de A à F (pour 10 à 15)

Conversion de la base 2 à la base 16

- Regrouper les bits par paquet de 4 à partir du bit le moins significatif (  $16 = 2^4$  )
- Calculer la valeur associée à chaque paquet

$$1010100111 = \begin{matrix} 0010 & 1010 & 0111 \\ 2 & A & 7 \end{matrix} = 2A7_{\text{base } 16}$$



### Conversion – base hexadécimale (base 16)

Comment convertir du hexadécimal au décimal?

$$\text{Rép.: } 2 * 16^2 + 10 * 16^1 + 7 * 16^0 = 679$$

Pour vérifier que le résultat précédent est bon, il s'agit de convertir binaire à décimal et s'assurer que le résultat est le même:

$$1010100111 = 512 + 256 + 64 + 7 = 679$$

### Conversion – **base numérique avec MATLAB**

MATLAB offre des fonctions qui permettent de passer d'une base à l'autre rapidement. Il s'agit des fonctions:

```
dec2bin(nombre entier positif)
```

```
dec2hex(nombre entier positif)
```

```
bin2dec('Chaîne de caractères représentant les symboles')
```

```
hex2dec('Chaîne de caractères représentant les symboles')
```

#### Exemples :

```
nombre = 45;
```

```
bin = dec2bin(nombre)
```

```
bin = '101101' %Résultat
```

```
A=bin2dec('10101')
```

```
A=21 %Résultat
```



# Agenda



Représentation de nombres



Conversions, fonctions MATLAB

- Entiers: non signés, signés
- Nombres réels, norme IEEE754
- Les caractères

### Entiers – non signés

L'entier non signé est un entier positif.

Dans MATLAB: `uint8`, `uint16`, `uint32`, `uint64`

Il n'y a pas de bit de signe.

Intervalle de représentation:

$[0, 2^n - 1]$  où  $n$  est le nombre de bits

i.e., pour une représentation sur 8 bits

Entier non signé :  $[0, 2^8 - 1] = [0, 255]$



### Entiers – non signés - addition

À chaque étage, les 4 règles de base (i.e. la logique) de l'addition de nombres binaires sont les suivantes:

$0 + 0 = 0$	Somme de 0 et pas de retenue
$0 + 1 = 1$	Somme de 1 et pas de retenue
$1 + 0 = 1$	Somme de 1 et pas de retenue
$1 + 1 = 0$	Somme de 0 et retenue



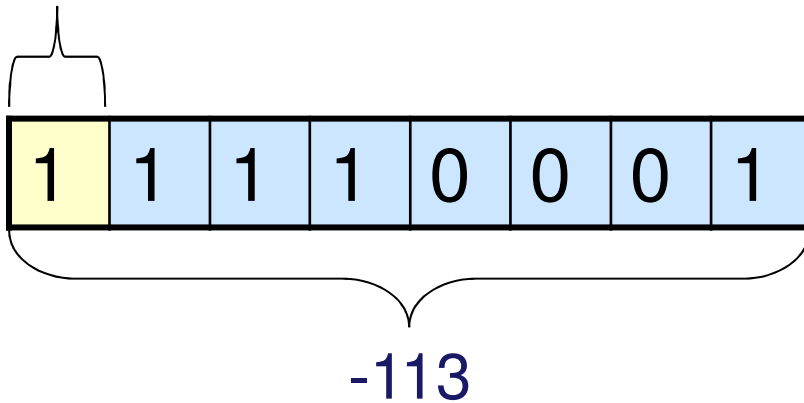
### Entiers – **signés**

Le bit le plus significatif, le plus à gauche précise le signe du nombre:

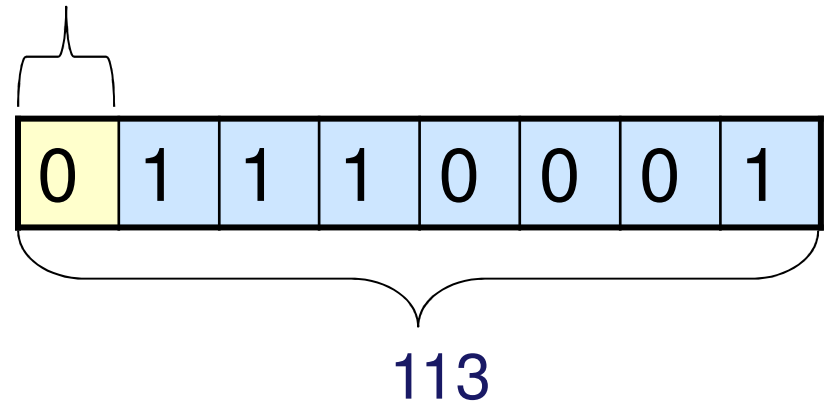
- 0 indique un nombre positif
- 1 indique un nombre négatif

Exemple, représentation de -113 et 113 sur 8 bits:

**Bit de  
signe**



**Bit de  
signe**







### Entiers – **signés en complément**

- Le complément à 2 d'un nombre binaire est important puisqu'il permet la représentation de nombres négatifs.
- Il existe aussi le complément à 1, mais les ordinateurs utilisent surtout le complément à 2.

Nous allons donc approfondir le complément à 2

### Entiers – **signés en complément à 2**

Dans MATLAB: `int8`, `int16`, `int32`, `int64`

Pour obtenir la représentation du nombre négatif il faut inverser tous les bits du nombre positif et y ajouter 1.

i.e. -113 :

	0	1	1	1	0	0	0	1	+ 113
	1	0	0	0	1	1	1	0	<b>Inversion des bits</b>
+	1							<b>Additionne 1</b>	
	1	0	0	0	1	1	1	1	= -113
	↓		↙	↘	↙	↘	↙	↘	
	-128	+	8	+	4	+	2	+	1 = 113



### Entiers – **signés en complément à 2**

Représentation unique de 0

$$+0 = -0 = 00000000\dots000$$

Intervalle de la représentation

$[-(2^{n-1}), (2^{n-1} - 1)]$  où  $n$  est le nombre de bits utilisés

Addition simple selon les règles vues précédemment

# Entiers – **signés en complément à 2** – addition et validation

- Additionner bit à bit
- Ignorer la retenue
- Validité de la réponse
  - ✓ Si les deux opérandes sont de signe opposé, la réponse est exacte
  - ✓ Si les deux opérandes sont de même signe, mais que la réponse est de signe opposé, on dit qu'il y a *débordement* et la réponse est alors inexacte

$$\begin{array}{r}
 \begin{array}{ccccccc} & 1 & & 1 & & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & -85 \\ + \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 43 \\ \hline 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & -42 \end{array}
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{ccccccc} & 1 & & 1 & & 1 & \\ \hline 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 43 \\ + \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 11 \\ \hline 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 54 \end{array}
 \end{array}$$

**Deux sommes correctes**



# Entiers – signés en complément à 2 – addition et validation

$$\begin{array}{r} \phantom{+} \begin{array}{ccccccc} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{array} 43 \\ + \begin{array}{ccccccc} 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{array} 107 \\ \hline \begin{array}{ccccccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array} -106 \end{array}$$

Somme  
incorrecte et  
donc rejetée

Sur 8 bits l'intervalle des valeurs possibles est

$$[-(2^{8-1}), (2^{8-1} - 1)] = [-128, 127]$$

Puisque  $43 + 107 = 150$  est une valeur à l'extérieur de l'intervalle, nous sommes en présence d'un cas de *dépassement de valeur* ou plus simplement un cas de **débordement**?

La valeur obtenue -106 est inexacte.



# Entiers – signés en complément à 2 – addition et validation

➤ **Attention!** Ne pas confondre débordement (overflow) et retenue (carry).



# Agenda



Représentation de nombres



Conversions, fonctions MATLAB



Entiers: non signés, signés

- Nombres réels, norme IEEE754

- Les caractères

### Les nombres réels

Les nombres réels s'expriment principalement sous deux formes:

- La notation décimale
- La notation scientifique

La notation scientifique est celle qui permet un format de mémorisation standardisé. Il incorpore quatre éléments: le **signe**, la **mantisse**, la **base** et l'**exposant**.

Pour  $-2,7 \times 10^5$

- est le signe
- 2,7** est la mantisse
- 10** est la base
- 5** est l'exposant



### Les nombres réels

La mantisse doit respecter la règle:

$$1 \leq |\text{Mantisse}| < \text{Base}$$

Ainsi:

$0,1245 \times 10^{-23}$  est inacceptable

$12,45 \times 10^{-25}$  est inacceptable

$1,245 \times 10^{-24}$  est la seule et unique représentation acceptable

Connaissant la base, il est possible de représenter le nombre à l'aide du triplet:

(Signe, Exposant, Mantisse)

### Les nombres réels binaires – norme IEEE754


Un nombre réel binaire est mémorisé en respectant la norme IEEE754

Dans MATLAB: `double`, `float64`, `float32`

	Nombre bit du signe	Nombre de bits de l'exposant	Nombre de bits de la mantisse
Simple précision 32 bits	1	8	23
Double précision 64 bits	1	11	52

# Les nombres réels binaires – norme IEEE754

## Règles

- Le bit de signe est à 0 pour un nombre positif ou à 1 pour un nombre négatif
- La mantisse est représentée par notation positionnelle. 
- Le premier bit de la partie entière est sous-entendu puisque toujours 1.
- L'exposant est représenté par excès de  $2^{n-1}-1$ . Il s'agit de l'astuce utilisée pour combler l'absence du signe de l'exposant. En simple précision l'excès est de 127 tandis qu'en double précision l'excès est de 1023.



## Les nombres réels binaires – norme IEEE754, exemple

Conversion de 132.147 en format simple précision

= 0 10000110 00001000010010110100010

Dernier bit est  
arrondi à 1

Partie entière = 132		Partie fractionnaire=147	
nbre	nbre bin	nbre	Partie ent. de (2×nb)
		0.147	0
132	0	0.294	0
66	0	0.588	1
33	1	0.176	0
16	0	0.352	0
8	0	0.704	1
4	0	0.408	0
2	0	0.816	1
1	1	0.632	1
0		0.264	0
		0.528	1
		0.056	0
		0.112	0
		0.224	0
		0.448	0
		0.896	1
		0.792	1

Pour arrondi.

Si 1, fait +1,  
sinon le dernier  
bit reste  
comme il est.

132 . 147

10000100 . 0010010110100001

10000100 = 1.0000100 × 2<sup>7</sup>

1. 00001000010010110100010 × 2<sup>7</sup>

Signe = 0

Exposant = 7 + 127 = 134

= 10000110

Mantisse = 00001000010010110100010

S	Exposant	Mantisse
0	1000 0110	0000 1000 0100 1011 0100 010

On a 23 bits pour la mantisse

# Les nombres réels binaires – norme IEEE754, exceptions

Dans les deux formats, la norme prévoit des exceptions:

Signe	Exposant	Mantisse
1 ou 0	0000...00	0000...000

Représente 0

Signe	Exposant	Mantisse
1 ou 0	1111...11	000...000

Débordement (overflow). inf dans MATLAB.

Signe	Exposant	Mantisse
1 ou 0	0000...00	xxxx...xx (!=0)

Débordement (underflow). MATLAB donne 0.

Signe	Exposant	Mantisse
1 ou 0	1111...11	xxxx...xx (!=0)

NAN (Not A Number), par ex. zéro divisé par zéro. NaN dans MATLAB.

# Les nombres réels binaires – IEEE754, caractéristiques

Réel maximal

Signe	Exposant	Mantisse
1 ou 0	1111...10	1111...11

$$= 1.111...11 \times 2^{(111...10) - \text{Excès}}$$

$$1.111...11 = 1 + 2^{-1} + 2^{-2} + 2^{-3} + \dots + 2^{-M}$$

où M=nbre de bits de la mantisse

La M ième somme partielle de la série géométrique avec  $|c| < 1$  est

$$1 + c^1 + c^2 + c^3 + \dots + c^M = \frac{1 - c^{M+1}}{1 - c}$$

Dans notre cas,  $c = 1/2$

$$1.111...11 = \frac{1 - (1/2)^{M+1}}{1/2} = 2 - (1/2)^M$$

$$111...10 - \text{Excès} = [(2^E - 1) - 1] - (2^{E-1} - 1)$$

où E=nbre de bits de l'exposant

$$\begin{aligned} 111...10 - \text{Excès} &= (2^E - 2) - (2^{E-1} - 1) \\ &= 2(2^{E-1} - 1) - (2^{E-1} - 1) \\ &= (2^{E-1} - 1) \end{aligned}$$

Finalement,

$$\begin{aligned} &1.111...11 \times 2^{(111...10) - \text{Excès}} \\ &= (2 - 2^{-M}) \times 2^{(2^{E-1} - 1)} \end{aligned}$$

# Les nombres réels binaires – IEEE754, caractéristiques

Réel minimal

Signe	Exposant	Mantisse
1 ou 0	0000...01	0000...00

$$= 1.0 \times 2^{(0000...01) - \text{Excès}}$$

$$0000...01 - \text{Excès} = 1 - (2^{E-1} - 1)$$

où E=nombre de bits de l'exposant

$$0000...01 - \text{Excès} = 2 - 2^{E-1}$$

Finalement,

$$\begin{aligned}
 & 1.0 \times 2^{(0000...01) - \text{Excès}} \\
 &= 1.0 \times 2^{(2 - 2^{E-1})} \\
 &= 2^{(2 - 2^{E-1})}
 \end{aligned}$$

# Les nombres réels binaires – IEEE754, caractéristiques

La granularité est la différence existant entre un nombre réel et le nombre réel suivant pour une précision donnée (i.e. pour un exposant donné).

La granularité est le dernier bit de la mantisse \* exposant

Exemple #1 en simple précision:

Nombre réel (1.0):

S	Exposant	Mantisse
0	01111111	0000000000000000000000000

Nombre réel suivant  
(1,000000119):

S	Exposant	Mantisse
0	01111111	0000000000000000000000001

$$\text{La granularité} = 2^{-M} * 2^{(127-127)} = 2^{-23} * 1 = 1.19 \times 10^{-7}$$

Avec MATLAB, essayer: `eps(single(1))`





## Les nombres réels binaires – IEEE754, caractéristiques

Exemple #2: Granularité pour l'exposant le plus petit en simple précision:

Nombre réel ( $1,2 \times 10^{-38}$ ):

S	Exposant	Mantisse
0	00000001	000000000000000000000000 <b>0</b>

## Nombre réel suivant

$(1,200000140129846432481707092373 \times 10^{-38})$ :

S	Exposant	Mantisse
0	00000001	000000000000000000000000 <b>1</b>

La granularité =  $2^{-M} * 2^{(1-127)} = 2^{-23} * 2^{-126} = 1.4 \times 10^{-45}$

Avec MATLAB, essayer: `eps(single(realmin))`

# Les nombres réels binaires – IEEE754, caractéristiques

	Simple précision	Double précision
Réel Maximal <code>realmax()</code>	$3,4 \times 10^{38}$	$1,8 \times 10^{308}$
Réel Minimal <code>realmin()</code>	$1,2 \times 10^{-38}$	$2,2 \times 10^{-308}$



# Agenda



Représentation de nombres



Conversions, fonctions MATLAB



Entiers: non signés, signés



Nombres réels, norme IEEE754

- Les caractères

### Les caractères

- Les nombres constituent le langage des ordinateurs. Pour obtenir une lettre il faut un système de conversion du nombre à la lettre.
- Création du code ASCII (American Standard Code for Information Interchange).
- Le jeu de caractères ASCII standard correspond à un tableau de correspondance qui associe un nombre entier à une lettre, un chiffre, un symbole ou un code de contrôle.

Dans MATLAB: char et chaînes de caractères.

# Les caractères – code ASCII

## Lettres minuscules

<b>a</b> = 97	<b>b</b> = 98	<b>c</b> = 99	<b>d</b> = 100	<b>e</b> = 101
<b>f</b> = 102	<b>g</b> = 103	<b>h</b> = 104	<b>i</b> = 105	<b>j</b> = 106
<b>k</b> = 107	<b>l</b> = 108	<b>m</b> = 109	<b>n</b> = 110	<b>o</b> = 111
<b>p</b> = 112	<b>q</b> = 113	<b>r</b> = 114	<b>s</b> = 115	<b>t</b> = 116
<b>u</b> = 117	<b>v</b> = 118	<b>w</b> = 119	<b>x</b> = 120	<b>y</b> = 121
<b>z</b> = 122				

## Lettres majuscules

<b>A</b> = 65	<b>B</b> = 66	<b>C</b> = 67	<b>D</b> = 68	<b>E</b> = 69
<b>F</b> = 70	<b>G</b> = 71	<b>H</b> = 72	<b>I</b> = 73	<b>J</b> = 74
<b>K</b> = 75	<b>L</b> = 76	<b>M</b> = 77	<b>N</b> = 78	<b>O</b> = 79
<b>P</b> = 80	<b>Q</b> = 81	<b>R</b> = 82	<b>S</b> = 83	<b>T</b> = 84
<b>U</b> = 85	<b>V</b> = 86	<b>W</b> = 87	<b>X</b> = 88	<b>Y</b> = 89
<b>Z</b> = 90				

Lettre	H	e	I	I	o
Binaire	0100 1000	0110 0101	0110 1100	0110 1100	0110 1111
Decimal	72	101	108	108	111

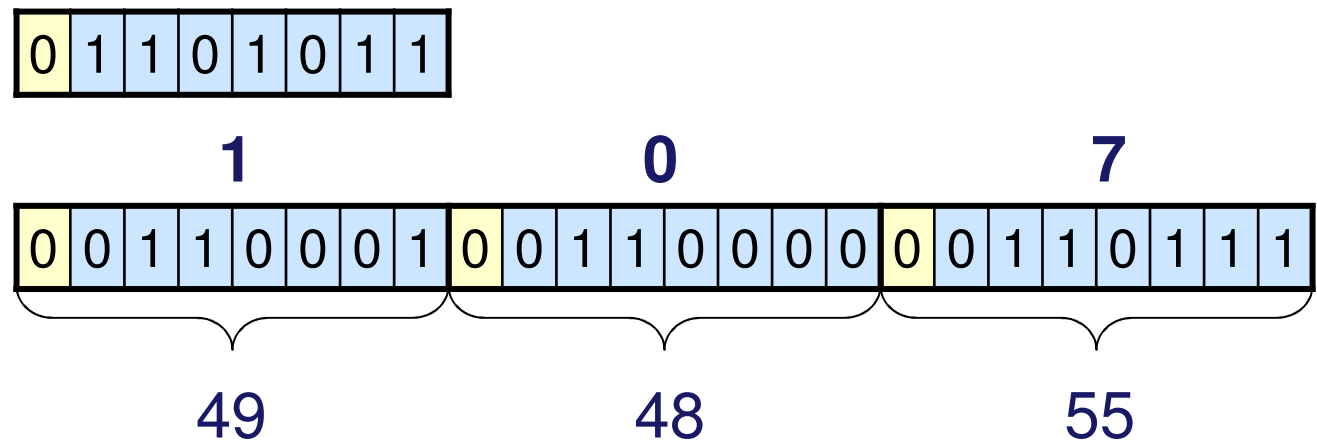
# Les caractères – code ASCII

Les chiffres existe aussi en caractère.

chiffres	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>
valeur ASCII	48	49	50	51	52	53	54	55	56	57

Le nombre 107 en  
complément à 2

s'écrit en  
caractères ASCII



# Les caractères – **Unicode**

Unicode, dont la première publication remonte à 1991, a été développé dans le but de remplacer l'utilisation de pages de code nationales.

Une **page de code** est un standard informatique national/local qui vise à donner à tout caractère national /local un numéro. Cependant les pages de code ayant été développées sur des bases nationales, elles ne permettent pas l'échange de document entre différents pays. Les pages de code les plus connues sont :

**page de code 437 américain, graphique**

**page de code 850 multilingue européen**

**ANSI (iso-latin-1)**

**VISCII vietnamien**

**CP1258**

Tous les systèmes les plus utilisés dans le monde sont représentés, ainsi que des règles sur la sémantique des caractères, leurs compositions et la manière de combiner ces différents systèmes (par exemple, comment insérer un système d'écriture de droite à gauche dans un système d'écriture de gauche à droite ?)

# Les caractères – Unicode

L'ASCII utilise 8 bits, l'Unicode utilise entre 20 et 21 bits par caractère.

Unicode attribue un numéro à des symboles. Il ne code pas les descriptions des caractères, les *glyphes*, c'est-à-dire la représentation graphique du caractère. Il n'y a donc pas une bijection entre la représentation du caractère et son numéro comme c'est le cas dans une police ASCII classique. Le caractère français é peut être décrit de deux manières : soit en utilisant directement le numéro correspondant au é, soit en faisant suivre le numéro du 'e' par celui de l'accent aigu sans chasse. Quelle que soit l'option choisie le même glyphe sera affiché.

Selon les polices installées sur votre ordinateur et la configuration de votre navigateur, le caractère s'affichera correctement ou non.





## Les caractères – Unicode

- Unicode classe plus de 1 10 000 caractères, symboles couvrant 100 écritures.
- La version actuelle est la version 6.3 de septembre 2013.
- Des problèmes semblent cependant exister, pour le codage des caractères chinois (en cours d'étude).



# Les caractères – **Unicode: caractères 200 à 309 pour la police Arial**

	0	1	2	3	4	5	6	7	8	9
200	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ
210	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û
220	Ü	Ý	Þ	ß	à	á	â	ã	ä	å
230	æ	ç	è	é	ê	ë	ì	í	î	ï
240	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù
250	ú	û	ü	ý	þ	ÿ	Ā	ā	Ă	ă
260	Ą	ą	Ć	ć	Ĉ	ĉ	Ċ	ċ	Č	č
270	Ď	ď	Đ	đ	Ē	ē	Ĕ	ĕ	Ė	ė
280	Ę	ę	Ė	ė	Ĝ	ĝ	Ğ	ğ	Ġ	ġ
290	Ģ	ģ	Ĥ	ĥ	Ħ	ħ	Ĩ	ĩ	Ī	ī
300	Ĳ	ķ	Ļ	ļ	İ	ı	Ĳ	ĳ	Ĵ	ĵ



# Agenda



Représentation de nombres



Conversions, fonctions MATLAB



Entiers: non signés, signés



Nombres réels, norme IEEE754



Les caractères

# Sommaire

- 1 Représentation de nombres
- 2 Conversions, fonctions MATLAB
- 3 Entiers: non signés, signés
- 4 Nombres réels, norme IEEE754
- 5 Les caractères