TD 13 -
Représentation des nombres réels.

Informatique
MPSI-PCSI - Lycée Thiers

Exercice 1

Exercice 1. Dans la représentation normalisée des nombres flottant sur 32 bits :

$$x = \varepsilon \times m \times 2^e$$
$$\varepsilon = \pm 1 \quad ; \quad m \in [1, 2[\quad ; \quad e \in \mathbb{Z}$$

- Le bit de poids fort représente le signe : 1 pour négatif, 0 pour positif,
- Les 8 bits suivants représentent l'écriture binaire de $e + 127$.
- les 23 bits suivants représentent l'écriture binaire après la virgule de `m-1`.

Corrigé

1. Comment s'écrit le nombre `-1.0` ?
   : $-1,0 = -1 \times 2^0$
   Signe : $-$; exposant 0 $\Longrightarrow$ $0 + 127 = 127$; mantisse $m = 1$

   1 01111111 00000000000000000000000

2. Comment s'écrit le nombre `1.5` ?
   $1.5 = +1.5 \times 2^0$
   Signe : $+$; exposant 0 $\Longrightarrow$ $0 + 127 = 127$; mantisse $m = 1.5$
   $\Longrightarrow$ $m - 1 = \overline{0,1}$

   0 01111111 10000000000000000000000

Corrigé

1. Comment s'écrit le nombre 4.75 ?
   $4.75 = 4 + 1/2 + 1/4 = 2^2 + 2^{-1} + 2^{-2} = +(1 + 2^{-3} + 2^{-4}) \times 2^2$
   Signe : $+$; exposant $2 \Longrightarrow 2 + 127 = 129$;
   mantisse $m = 1 + 2^{-3} + 2^{-4} \Longrightarrow m - 1 = \overline{0,0011}$

   $$0\ 10000001\ 00110000000000000000000$$

2. Quel nombre représente :

   $$1\ 01111111\ 11000000000000000000000$$

   Signe : $-$; Exposant : $e + 127 = 127 \Longrightarrow e = 0$
   mantisse : $m = \overline{1,11} = 1 + 1/2 + 1/4 = 1,75.$
   C'est le nombre $-1,75$.

Corrigé

1. Quel nombre représente :

$$0\,10000001\,00100000000000000000000$$

Signe : $+$; Exposant : $e + 127 = 129 \Longrightarrow e = 2$
mantisse : $m = \overline{1,001} = 1 + 1/8 = 1,125$.
C'est le nombre $(1 + 1/8) \times 2^2 = 4 + 1/2 = +4,5$.

2. Quel nombre représente :

$$1\,01111110\,10000000000000000000000$$

Signe : $-$; Exposant : $e + 127 = 126 \Longrightarrow e = -1$
mantisse : $m = \overline{1,1} = 1 + 1/1 = 1,5$.
C'est le nombre $-(1 + 1/2) \times 2^{-1} = -(0,5 + 1/4) = -0,75$.

Exercice 2

Racines d'un polynôme de degré 2

1. Reprendre la fonction `trinome(a,b,c)` du TD 3 prenant en paramètres 3 nombres à virgule flottante `a`, `b`, `c` et qui à l'aide du discriminant retourne la ou les racines du polynôme de degré 2 : $P(X) = aX^2 + bX + c$.
2. Appliquer la recherche de racines au polynôme :

$$X^2 + X + \frac{1 + 2^{-53}}{4}$$

en appelant `racine_trinome(1,1,(1+2**-53)/4)`.

3. Définir la variable `e = -53`, et lancer l'instruction de comparaison avec 0 :

```
>>> e = - 53
>>> 1 + 2**e - 1 == 0
```

Recommencer après l'affectation : `e = -52`. Discuter de la validité du résultat trouvé à la question précédente.

4. Faire de même avec le polynôme :

$$X^2 - \sqrt{2}X + \frac{1}{2}$$

Le résultat retourné est-il correct ?

Exercice 2 : Réponse

1)

```
def trinome(a,b,c):
    assert a != 0
    D = b**2 - 4*a*c
    if D == 0 :
        print("Une seule racine")
        return -b/(2*a)
    elif D > 0 :
        print("Deux racines réelles")
        return (-b-D**0.5)/(2*a), (-b+D**0.5)/(2*a)
    else:
        print("Deux racines complexes conjuguées")
        return (-b-1j*(-D)**0.5)/(2*a), (-b+1j*(-D)**0.5)/(2*a)
```

Exercice 2 : Réponse

2)

```
In [1] : e = -53
In [2] : a, b, c = 1, 1, (1.0+2**e)/4.0
In [3] : trinome(a,b,c)
Une seule racine
Out [3] : -0.5
```

```
In [4] : e = -53
In [5] : 1.0 + 2**e == 1
Out [5] : True
In [6] : 2**e == 0
Out [6] : False
In [7] : e = -52
In [8] : 1.0 + 2**e == 1
Out [8] : False
```

Problème dans la comparaison d'un nombre à virgule flottante avec 0.

Exercice 2 : Réponse

3) De même

$$X^2 - \sqrt{2}X + \frac{1}{2}$$

a discriminant $\Delta = (\sqrt{2})^2 - 2 = 0$.
On obtient pourtant deux racines distinctes.

Exercice 3

1. Quel est le plus petit nombre > 1 qui peut être représenté sur un ordinateur 64 bits ? Quels sont les 2 suivants ?
2. Quel est le plus petit nombre > 2 qui peut être représenté sur un ordinateur 64 bits ? Quels sont les 2 suivants ?
3. Quel est le plus petit nombre > 4 qui peut être représenté sur un ordinateur 64 bits ? Quels sont les 2 suivants ?
4. Quel est le plus petit nombre $> 1/2$ qui peut être représenté sur un ordinateur 64 bits ? Quels sont les 2 suivants ?
5. Combien de nombres entre 1 inclu et 2 exclu peut -on représenter ? Combien entre 2 et 4 ? Combien entre $1/2$ et 1 ?

Exercice 3

1. Quel est le plus petit nombre > 1 qui peut être représenté sur un ordinateur 64 bits ? Quels sont les 2 suivants ?
   Réponse : $1 + 2^{-52}$; $1 + 2 \times 2^{-52}$; $1 + 3 \times 2^{-52}$.
2. Quel est le plus petit nombre > 2 qui peut être représenté sur un ordinateur 64 bits ? Quels sont les 2 suivants ?
   Réponse : $2 + 2^{-51}$; $2 + 2 \times 2^{-51}$; $2 + 3 \times 2^{-51}$.
3. Quel est le plus petit nombre > 4 qui peut être représenté sur un ordinateur 64 bits ? Quels sont les 2 suivants ?
   Réponse : $4 + 2^{-50}$; $4 + 2 \times 2^{-50}$; $4 + 3 \times 2^{-50}$.
4. Quel est le plus petit nombre $> 1/2$ qui peut être représenté sur un ordinateur 64 bits ? Quels sont les 2 suivants ?
   Réponse : $0.5 + 2^{-53}$; $0.5 + 2 \times 2^{-53}$; $0.5 + 3 \times 2^{-53}$.
5. Combien de nombres entre 1 inclu et 2 exclu peut -on représenter ? Combien entre 2 et 4 ? Combien entre $1/2$ et 1 ?
   Réponse : chaque fois : exactement 2^{52}.

Exercice 4

Dans la représentation des réels en virgule flottante sur 8 bits, on utilise :
– 1 bit, celui de poids le plus fort, pour le signe,
– les 2 bits suivants pour l'exposant $e \in [[-1, 2]]$. On code $e + 1$ sur 2 bits en binaire.
– les 5 bits restants pour la mantisse.
Le principe du codage est par ailleurs le même que dans l'écriture normalisée des nombres à virgules flottantes.

1. Comment représenter sur 8 bits le nombre $2,75$?
2. Quel nombre représente `'00111111'` ?
3. Quel est le plus petit nombre que l'on peut représenter ainsi ? Le plus grand ?

Exercice 4

1)

$2,75 = 2 \times 1,375 = 2 \times (1 + 0,25 + 0,125) = 2^1 \times (1 + 2^{-2} + 2^{-3})$

signe : positif ;

exposant : $e = 1$ (on code $e + 1 = 2 = \overline{10}$) ;

mantisse $1,375 = \overline{1,011}$ (on code 011 sur 5 bits).

$$\boxed{0\,10\,01100}$$

2)

$$0\,01\,11111$$

bit de signe : 0 $\Longrightarrow$ signe positif

bits d'exposants : 10 $\Longrightarrow$ $e + 1 = 2$ $\Longrightarrow$ exposant 1

bits de mantisse : 11111

$\Longrightarrow$ $m = 1 + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} = 2 - 2^{-5} = 1.96875$

Le nombre représenté est :

Exercice 4

3)

Le plus petit nombre représenté en valeur absolue est :

$$0\,00\,00000$$

C'est $2^{-1} \times 1 = \boxed{0,5}$.

Le plus grand nombre représenté en valeur absolue est :

$$0\,11\,11111$$

C'est $2^2 \times (1 + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5}) = 4 \times 1.96875 = \boxed{7.875}$.

Exercice 5

Le code qui suit permet de tracer graphiquement tous les nombres réels se représentant graphiquement sur 8 bits (voir description à l'exo précédent).

1. Expliquer le rôle de chaque fonction.
2. L'exécuter. Après avoir "zoomé" sur certaines parties du graphiques, quelles constatations faites-vous ?

Exercice 5

1)

```
def binaire(n,N):
    s = ''
    while n>0:
        s = str(n%2) + s
        n = n//2
    return '0'*(N-len(s)) + s
```

Retourne l'écriture binaire sur N bits (str) de l'entier positif n.

```
def bin2dec(s):
    r = 0
    for c in s:
        r *= 2
        r += int(c)
    return r
```

Prend en paramètre une chaine de 0 et 1, représentant un nombre en binaire, et retourne l'entier représenté.

Exercice 5

1)

```
S = bin2dec('10000000') # pour récupérer le bit de poids
fort
E = bin2dec('01100000') # pour récupérer les bits de
l'exposant
M = bin2dec('00011111') # pour récupérer les bits de la
mantisse
```

Définit des "filtres" pour récupérer bit de signe, exposant et mantisse.

```
def signe(n):
    if n & S == 0:
        return +1
    else:
        return -1

def exposant(n):
```

Exercice 5

Même chose pour la mantisse :

```
def mantisse(n):
    m = M & n
    s = binaire(m,5)
    r = 1
    p = 1
    for i in range(5):
        p /= 2
        r += int(s[i]) * p
    return r
```

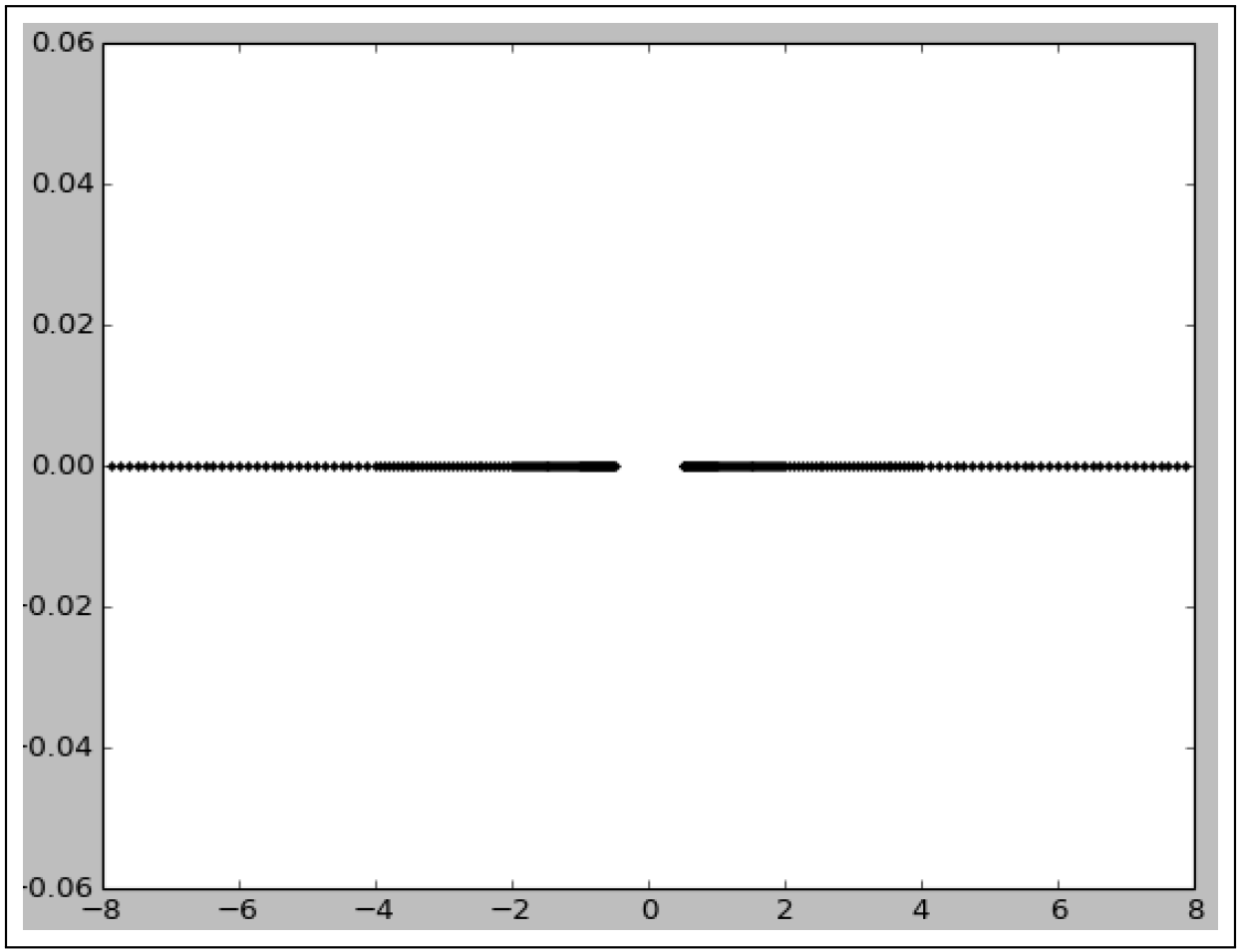
Exercice 5

On énumère tous les entiers de 0 à 255. Pour chacun on récupère signe, mantisse et exposant de son écriture sur 8 bits, pour obtenir le flottant ayant même représentation sur 8 bits. On stocke ces éléments dans une liste, puis on éxecute leur tracé.

```
L = []
for n in range(256):
    L.append(signe(n)*mantisse(n)*2**exposant(n))
Y = [0] * len(L)
import matplotlib.pyplot as plt
plt.plot(L,Y,'.k')
plt.show()
```

Exercice 5

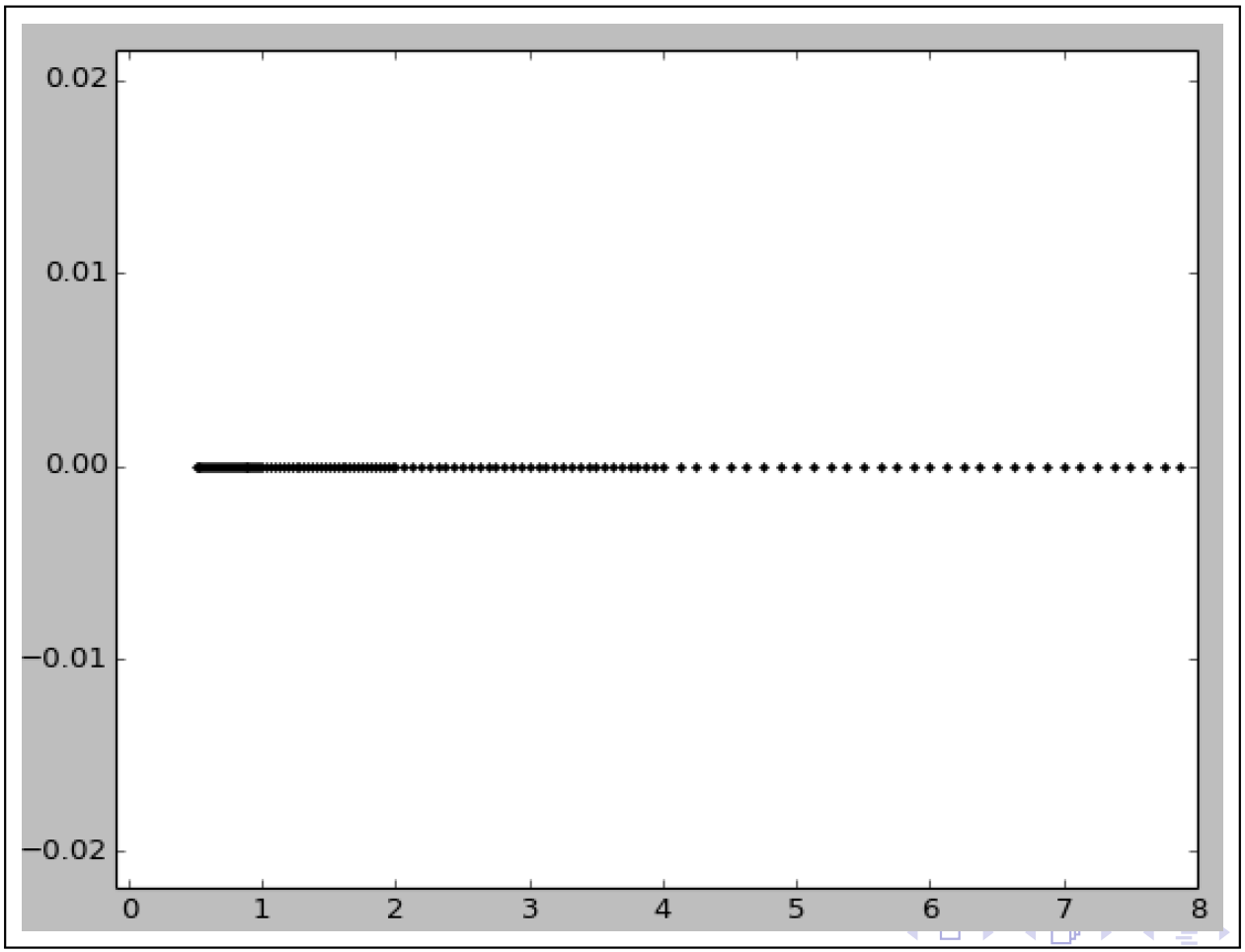
2) Les nombres représentés sont symétriques par rapport à 0 :



Dans l'écriture "normalisé" le zéro n'est pas représenté.

Exercice 5

En valeur absolue le plus petit nombre représenté est 0,5, le plus grand 7,875



Exercice 5

Il y a autant de nombres représentés dans $[0.5, 1[$, que dans $[1, 2[$, que dans $[2, 4[$ et $[4, 8[$ (2^5).