# 1.2. Nombre de solution d'un système

April 20, 2020

## 1 Concept(s)-clé(s) et théorie

On considère un système d'équations linéaires aux inconnues $x_1, \ldots, x_n$

$$S = \begin{cases} a_{11}x_1 & +a_{12}x_2 & + & \cdots & +a_{1n}x_n & = & b_1 \\ a_{21}x_1 & +a_{22}x_2 & + & \cdots & +a_{2n}x_n & = & b_2 \\ \vdots & & & & \vdots & \vdots & \vdots \\ a_{m1}x_1 & +a_{m2}x_2 & + & \cdots & +a_{mn}x_n & = & b_m \end{cases},$$

où $a_{ij}, b_i \in \mathbb{R}$ pour tout $1 \leq i \leq m$ et tout $1 \leq j \leq n$.

---

Un système d'équations linéaires à coefficients réels satisfait précisément une des conditions suivantes.

1.Le système ne possède aucune solution.

2.Le système possède une solution unique.

3.Le système possède une infinité de solutions.

[ ]:
```
import Librairie.AL_Fct as al
import numpy as np
import matplotlib.pyplot as plt
import plotly as py
import plotly.graph_objs as go
import pandas as pd
import random
from IPython.core.magic import register_cell_magic
from IPython.display import HTML, display
from ipywidgets import interactive, HBox, VBox, widgets, interact, FloatSlider

py.offline.init_notebook_mode(connected=True)
```

### 1.0.1 EXEMPLE 1

Nous allons travailler avec un système de 2 inconnues, 2 équations:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2. \end{cases}$$

On utilise *la syntaxe suivante pour entrer les coefficients du système*

$$\begin{array}{lllll} A = & [ & [a_{11}, a_{12}, \ldots, a_{1n}], & [a_{21}, a_{22}, \ldots, a_{2n}] \quad , \ldots , & [a_{m1}, a_{m2}, \ldots, a_{mn}] \quad ] \\ b = & [ & b_1, b_2, \ldots, b_m \quad ] & & \end{array}$$

---

Entrer le système ci-dessous - ou n'importe quel système de votre choix. Après avoir étudié les deux droites, entrer une solution du système

$$\begin{cases} -2x_1 + x_2 = 2 \\ 2x_1 - 2x_2 = 1. \end{cases}$$

[ ]:
```
al.bgc('seashell')
#Par défaut, les coefficients sont fixés à 1 et n=2, m=2

A = [ [-2,1] , [2,-2]]
b = [2,1]
```

[ ]:
```
al.printSyst(A,b)

al.Plot2DSys(-7,7,15,A,b)
```

[ ]:
```
al.bgc('seashell')

alpha=[-2.5, -3]
```

[ ]:
```
al.SolOfSyst(alpha, A,b)
```

### 1.0.2 EXERCICE 1

Pour chacun des cas ci-dessous, déterminer la/les solution(s) des systèmes.

$$a) \quad \begin{cases} \frac{3}{2}x_1 - 2x_2 = 0 \\ -3x_1 + 5x_2 = 3 \end{cases} \qquad b) \quad \begin{cases} x_1 - x_2 = 2 \\ -3x_1 - 2x_2 = -1 \\ 2x_1 + 3x_2 = 4 \end{cases}$$

[ ]:
```
al.bgc('seashell')
#Par défaut, les coefficients sont fixés à 1
#Attention d'adapter les valeurs ci-dessous pour a) et b)
```

```
A = [[1, 1], [1, 1]]
b =[1,1]
alpha=[1,1]
```

[ ]:
```
al.printSyst(A,b)
al.SolOfSyst(alpha, A,b)
al.Plot2DSys(-7,7,15,A,b)
```

### 1.0.3 EXERCICE 2

Pour chacun des cas ci-dessous, déterminer la/les solution(s) des systèmes.

$$a) \begin{cases} x_1 - x_2 + 2x_3 & = 0 \\ -3x_1 & = 1 \\ 2x_1 + 5x_2 & = -2 \end{cases} \quad b) \begin{cases} x_1 + x_2 - 2x_3 & = 0 \\ 3x_2 - x_3 & = 1 \\ -2x_1 - 2x_2 + 4x_3 & = 2 \end{cases} \quad c) \begin{cases} x_1 + x_2 - 2x_3 & = -1 \\ 3x_2 - x_3 & = 1 \\ -2x_1 - 2x_2 + 4x_3 & = 2 \end{cases}$$

[ ]:
```
al.bgc('seashell')
#Par défaut, les coefficients sont fixés à 1

A = [[ 1, 1, 1] ,[1, 1, 1] ,[1, 1, 1]]
b= [1, 1, 1]
```

[ ]:
```
al.printSyst(A,b)
al.Plot3DSys(-10,10,100,A,b)
```

[ ]:
```
alpha = [1, 1, 1]
al.SolOfSyst(alpha, A,b)
```

### 1.0.4 EXERCICE 3

À l'aide des graphes interactifs ci-dessous, trouver les valeurs des paramètres $h$ et $k$ pour que les systèmes a), b), c) et d) admettent, si possible, 1. aucune solution 2. une unique solution 3. une infinité de solution

$$a) \begin{cases} x_1 + 5x_2 & = h \\ -2x_1 - 10x_2 & = 18 \end{cases} \quad b) \begin{cases} hx_1 + x_2 & = 3 \\ -2x_1 - x_2 & = -1 \end{cases} \quad c) \begin{cases} 3x_1 + hx_2 & = 1 \\ x_1 + 3x_2 & = h \end{cases} \quad d) \begin{cases} kx_1 + x_2 & = h \\ 3x_1 - 5x_2 & = 2 \end{cases}$$

### 1.1 CASE a)

[ ]:
```
A=[[1, 5], [-2, -10]] # we initialize the problem. The values of h and k must␣
 ↪be fixed to one specific value here
b=[0, 18]
al.printSyst(A, b) # the system printed below shows the values of h and k␣
 ↪inserted here!!

data=[]
x=np.linspace(-5,5,11)
m=len(A)
MatCoeff = [A[i]+[b[i]]for i in range(m)] #becomes augmented matrix
MatCoeff=np.array(MatCoeff)

for i in range(len(MatCoeff)):
    trace=go.Scatter(x=x,  y=(MatCoeff[i,2]-MatCoeff[i,0]*x)/MatCoeff[i,1],␣
 ↪name='a) Droite %d'%(i+1))
    data.append(trace)

f=go.FigureWidget(data=data,
    layout=go.Layout(xaxis=dict(
        range=[-5, 5]
    ),
    yaxis=dict(
        range=[-10, 10]
    ) )

)

def update_y(h):
    MatCoeff = [[1, 5, h]]
    MatCoeff=np.array(MatCoeff)
    f.data[0].y=(MatCoeff[0,2]-MatCoeff[0,0]*x)/MatCoeff[0,1]

freq_slider = interactive(update_y, h=(-15, 15, 1/2))
vb = VBox((f, freq_slider))
vb.layout.align_items = 'center'
vb
```

### 1.2 CASE b)

[ ]:
```
A=[[1, 1], [-2,-1]] # we initialize the problem. The value of h is fixed
b=[3, -1]
al.printSyst(A, b) # the system printed below shows the value of h inserted here
```

```
m=len(A)
MatCoeff = [A[i]+[b[i]]for i in range(m)] #becomes augmented matrix
MatCoeff=np.array(MatCoeff)
data=[]
x=np.linspace(-5,5,11)
MatCoeff=np.array(MatCoeff)
for i in range(len(MatCoeff)):
    trace=go.Scatter(x=x,  y= (MatCoeff[i,2]-MatCoeff[i,0]*x)/MatCoeff[i,1],␣
 ↪name='b) Droite %d'%(i+1))
    data.append(trace)

f=go.FigureWidget(data=data,
    layout=go.Layout(xaxis=dict(
        range=[-5, 5]
    ),
    yaxis=dict(
        range=[-10, 10]
    ) )

)

def mat(k):
    h=k
    MatCoeff=[[h,1,3]]
    return MatCoeff
def update_y(h):
    MatCoeff= mat(h)
    MatCoeff=np.array(MatCoeff)
    f.data[0].y=(MatCoeff[0,2]-MatCoeff[0,0]*x)/MatCoeff[0,1]

freq_slider = interactive(update_y, h=(-10, 10, 1/2))

vb = VBox((f, freq_slider))
vb.layout.align_items = 'center'
vb
```

### 1.3 CASE c)

[ ]:
```
A=[[3, 1], [1, 3]] # we initialize the problem. The values of h and k are fixed
b=[1, 1]
al.printSyst(A, b) # the system printed below shows the values of h and k␣
 ↪inserted here

m=len(A)
MatCoeff = [A[i]+[b[i]]for i in range(m)] #becomes augmented matrix
```

```
MatCoeff=np.array(MatCoeff)
data=[]
x=np.linspace(-25,25,101)
MatCoeff=np.array(MatCoeff)
for i in range(len(MatCoeff)):
    trace=go.Scatter(x=x,  y= (MatCoeff[i,2]-MatCoeff[i,0]*x)/MatCoeff[i,1],␣
 ↪name='c) Droite %d'%(i+1))
    data.append(trace)

f=go.FigureWidget(data=data,
    layout=go.Layout(xaxis=dict(
        range=[-25, 25]
    ),
    yaxis=dict(
        range=[-50, 50]
    ) )

)

def mat(h):
    MatCoeff= [[h, 3, 1],[3, 1, h] ]
    return MatCoeff

def update_y(h):
    MatCoeff= mat(h)
    MatCoeff=np.array(MatCoeff)
    f.data[0].y=(MatCoeff[0,2]-MatCoeff[0,0]*x)/MatCoeff[0,1]
    f.data[1].y=(MatCoeff[1,2]-MatCoeff[1,0]*x)/MatCoeff[1,1]

freq_slider = interactive(update_y, h=(-20, 20, 1))

vb = VBox((f, freq_slider))
vb.layout.align_items = 'center'
vb
```

### 1.4 CASE d)

[ ]:
```
A=[[1, 1], [3, -5]] # we initialize the problem. The values of h and k are fixed
b=[1, 2]
al.printSyst(A, b) # the system printed below shows the values of h and k␣
 ↪inserted here

m=len(A)
MatCoeff = [A[i]+[b[i]]for i in range(m)] #becomes augmented matrix
MatCoeff=np.array(MatCoeff)
```

```
data=[]
x=np.linspace(-15,15,101)
MatCoeff=np.array(MatCoeff)
for i in range(len(MatCoeff)):
    trace=go.Scatter(x=x,  y= (MatCoeff[i,2]-MatCoeff[i,0]*x)/MatCoeff[i,1],␣
 ↪name='d) Droite %d'%(i+1))
    data.append(trace)

f=go.FigureWidget(data=data,
    layout=go.Layout(xaxis=dict(
        range=[-15, 15]
    ),
    yaxis=dict(
        range=[-10, 10]
    ) )

)

def update_y(h, k):
    MatCoeff= [[k, 1, h],[2, -5, 5] ]
    MatCoeff=np.array(MatCoeff)
    f.data[0].y=(MatCoeff[0,2]-MatCoeff[0,0]*x)/MatCoeff[0,1]


    f.data[1].y=(MatCoeff[1,2]-MatCoeff[1,0]*x)/MatCoeff[1,1]

freq_slider = interactive(update_y, h=(-10, 10, 1/10),k=(-10, 10, 1))

vb = VBox((f, freq_slider))
vb.layout.align_items = 'center'
vb
```

### 1.5 Exercice 4

À l'aide des graphes interactifs ci-dessous, trouver les valeurs des paramètre $h$ pour que le système admette, si possible, 1. aucune solution 2. une unique solution 3. une infinité de solution

$$\begin{cases} hx_1 - 2x_2 & = 4 \\ -2x_1 + hx_2 & = -4 \\ x_1 - x_2 & = h \end{cases}$$

[ ]:
```
np.seterr(divide='ignore', invalid='ignore')

A=[[1, -2], [-2, 1], [1, -1]] # we initialize the problem. The values of h and␣
 ↪k are fixed
```

```
b=[4, -4, 0]
al.printSyst(A, b) # the system printed below shows the values of h and k␣
 ↪inserted here

m=len(A)
MatCoeff = [A[i]+[b[i]]for i in range(m)] #becomes augmented matrix
MatCoeff=np.array(MatCoeff)
data=[]
x=np.linspace(-25,25,101)
MatCoeff=np.array(MatCoeff)
for i in range(len(MatCoeff)):
    if MatCoeff[i,1] == 0:
        MatCoeff[i,1] += 1e-3
    trace=go.Scatter(x=x,  y=(MatCoeff[i,2]-MatCoeff[i,0]*x)/MatCoeff[i,1],␣
 ↪name='c) Droite %d'%(i+1))
    data.append(trace)

f=go.FigureWidget(data=data,
    layout=go.Layout(xaxis=dict(
        range=[-25, 25]
    ),
    yaxis=dict(
        range=[-100, 100]
    ) )

)

def mat(h):
    MatCoeff= [[h, -2, 4], [-2, h, -4], [1, -1, h]]
    return MatCoeff

def update_y(h):
    MatCoeff= mat(h)
    MatCoeff=np.array(MatCoeff)
    if MatCoeff[0,1] == 0:
        MatCoeff[0,1] += 1e-3
    f.data[0].y=(MatCoeff[0,2]-MatCoeff[0,0]*x)/MatCoeff[0,1]
    if MatCoeff[1,1] == 0:
        MatCoeff[1,1] += 1e-3
    f.data[1].y=(MatCoeff[1,2]-MatCoeff[1,0]*x)/MatCoeff[1,1]
    if MatCoeff[2,1] == 0:
        MatCoeff[2,1] += 1e-3
    f.data[2].y=(MatCoeff[2,2]-MatCoeff[2,0]*x)/MatCoeff[2,1]

freq_slider = interactive(update_y, h=(-20, 20, 0.5))

vb = VBox((f, freq_slider))
```

```
vb.layout.align_items = 'center'
vb
```

Passez au notebook du chapitre 1.3-4: Notation Matricielle