TP9/10 : Chaînes de Markov

Pré-requis : je vous invite à consulter les chapitres de cours correspondants sur ma page (support informatique). Pour ce TP, on pourra en particulier se reporter à la section « Tracés d'histogrammes » du CH 8.

▶ Dans votre dossier `Info_2a`, créer le dossier `TP_9`.

I. Notion de chaîne de Markov : illustration sur un exemple

I.1. Énoncé du problème

Doudou le hamster passe son temps entre ses trois activités favorites : dormir, manger et faire de la roue. Au début de la journée, il mange, et à chaque heure, il change d'activité selon les critères suivants.

1) Si, à l'heure n, il est en train de manger, alors il va dormir l'heure suivante avec probabilité 0.7 et faire de l'exercice avec probabilité 0.3.

2) Si, à l'heure n, il est en train de dormir, alors il continue à dormir l'heure $n+1$ avec probabilité 0.4, il va manger avec probabilité 0.3 et il va faire de l'exercice avec probabilité 0.3.

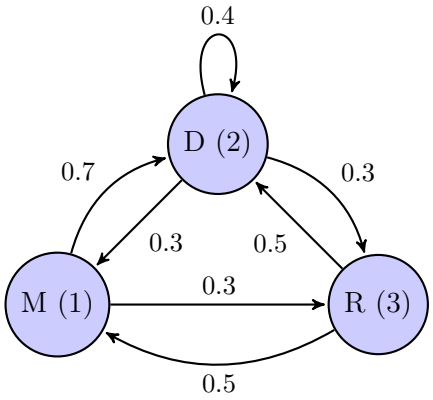
3) Si, à l'heure n, il est en train de faire de la roue, il va manger l'heure suivante avec probabilité 0.5 et il va dormir avec probabilité 0.5.

On s'intéresse ici à l'évolution du comportement de Doudou. On souhaite notamment déterminer si l'une de ses activités prendra, à terme, le dessus sur les autres.

I.2. Modélisation mathématique

On modélise ce problème comme suit.

- On commence par numéroter les activités par un entier entre 1 et 3.
- On note X_n la v.a.r. égale à l'état du hamster à l'heure n.
  Ainsi, la suite de v.a.r. (X_n) représente l'évolution des activités du hamster.
- Cette évolution peut être modélisée par le graphe suivant.



- On définit enfin la `matrice de transition` A associée au problème. Il s'agit de la matrice :

$$A = (a_{i,j}) \quad \text{où} \quad a_{i,j} = \mathbb{P}_{[X_n=j]}([X_{n+1} = i])$$

($a_{i,j}$ représente la probabilité de passage de l'état j à l'état i)

I.3. Étude de la matrice de transition

► Déterminer la matrice de transition du problème précédent.
Écrire l'appel permettant de la stocker dans une variable `A`.

$$A = \begin{pmatrix} 0 & 0.3 & 0.5 \\ 0.7 & 0.4 & 0.5 \\ 0.3 & 0.3 & 0 \end{pmatrix}$$

En **Scilab** : `A = [0, 0.3, 0.5 ; 0.7, 0.4, 0.5 ; 0.3, 0.3, 0]`

► Déterminer $\mathbb{P}_{[X_0=j]}([X_1 = i])$. À quel cœfficient de la matrice A cela correspond-il ?

On a :
$$\mathbb{P}_{[X_0=j]}([X_1 = i]) = \mathbb{P}_{[X_n=j]}([X_{n+1} = i])$$
Ceci permet de démontrer que la matrice de transition est indépendante de n.
Cette propriété est appelée **homogénéité**.

► Soit $(i_0, \dots, i_n, i_{n+1}) \in [\![1, 3]\!]^{n+2}$. Que vaut $\mathbb{P}_{[X_0=i_0]\cap\dots\cap[X_n=i_n]}([X_{n+1} = i_{n+1}])$?

On a :
$$\mathbb{P}_{[X_0=i_0]\cap\dots\cap[X_n=i_n]}([X_{n+1} = i_{n+1}]) = \mathbb{P}_{[X_n=i_n]}([X_{n+1} = i_{n+1}])$$
Cette propriété est appelée `propriété de Markov`.
On la retient souvent par la phrase :
Le futur (la position X_{n+1} à l'instant $n+1$) ne dépend du passé (positions X_0, …, X_n) que par le présent (la position X_n à l'instant n).

Ces deux propriétés font de (X_n) une chaîne de Markov homogène.

Remarque

- Il est classique de faire l'étude d'une grandeur aléatoire qui varie dans le temps discret. Par exemple, on peut étudier :
  - × l'évolution du prix d'une action jour après jour,
  - × l'évolution du gain d'un joueur après chaque partie d'un jeu,
  - × le déplacement d'un mobile (d'une puce) sur les sommets d'un carré (ou sur un axe gradué), *(c'était le cas du sujet* `EDHEC 2017`*)*
  - × …

  Lorsque l'évolution se fait de sorte que l'état à un instant ne dépend que de l'état à l'instant précédent, on est dans le cadre de la chaîne de Markov.
- Même si le mot chaîne de Markov n'apparaît que dans le programme d'informatique (et donc pas dans celui de maths), il est fréquent de voir ce type d'étude aux concours. Comme on va le voir, cela permet de faire un sujet qui mêle les résultats des chapitres Probabilités (FPT, sce) et algèbre linéaire (étude d'une matrice et des ses puissances).

I.4. Matrice de transition et évolution de la loi de X_n

Dans la suite, on considère le vecteur U_n qui définit la loi de X_n :

$$U_n = \begin{pmatrix} \mathbb{P}([X_n = 1]) \\ \mathbb{P}([X_n = 2]) \\ \mathbb{P}([X_n = 3]) \end{pmatrix}$$

► Déterminer la probabilité $\mathbb{P}([X_{n+1} = 1])$ en fonction de $\mathbb{P}([X_n = 1])$, $\mathbb{P}([X_n = 2])$ et $\mathbb{P}([X_n = 3])$ et des cœfficients de la matrice A.

La famille $([X_n = 1], [X_n = 2], [X_n = 3])$ est un système complet d'événements.
Ainsi, d'après la formule des probabilités totales :

$$\begin{aligned}
\mathbb{P}([X_{n+1} = 1]) &= \sum_{k=1}^{3} \mathbb{P}([X_n = k] \cap [X_{n+1} = 1]) \\
&= \sum_{k=1}^{3} \mathbb{P}([X_n = k]) \times \mathbb{P}_{[X_n=k]}([X_{n+1} = 1]) \\
&= \mathbb{P}([X_n = 1]) \times \mathbb{P}_{[X_n=1]}([X_{n+1} = 1]) \\
&\quad + \mathbb{P}([X_n = 2]) \times \mathbb{P}_{[X_n=2]}([X_{n+1} = 1]) \\
&\quad + \mathbb{P}([X_n = 3]) \times \mathbb{P}_{[X_n=3]}([X_{n+1} = 1]) \\
&= a_{1,1} \times \mathbb{P}([X_n = 1]) + a_{1,2} \times \mathbb{P}([X_n = 2]) + a_{1,3} \times \mathbb{P}([X_n = 3])
\end{aligned}$$

► Déterminer de même $\mathbb{P}([X_{n+1} = 2])$ et $\mathbb{P}([X_{n+1} = 3])$.
En déduire que pour tout $n \in \mathbb{N}$, $U_{n+1} = A \times U_n$. Exprimer enfin U_n en fonction de A et de U_0.
Que signifie le choix $U_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$? Comment obtient-on U_n dans ce cas ?

En procédant de la même manière, on obtient :

- $\mathbb{P}([X_{n+1} = 2]) = a_{2,1} \times \mathbb{P}([X_n = 1]) + a_{2,2} \times \mathbb{P}([X_n = 2]) + a_{2,3} \times \mathbb{P}([X_n = 3])$
- $\mathbb{P}([X_{n+1} = 3]) = a_{3,1} \times \mathbb{P}([X_n = 1]) + a_{3,2} \times \mathbb{P}([X_n = 2]) + a_{3,3} \times \mathbb{P}([X_n = 3])$

On en déduit que : $\forall n \in \mathbb{N},\ U_{n+1} = A \times U_n$.
Par une récurrence immédiate, on obtient : $\forall n \in \mathbb{N},\ U_n = A^n \times U_0$.

Si $U_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$, c'est que $\mathbb{P}([X_0 = 1]) = 1$. Ce choix signifie donc que le hamster est initialement dans l'état 1. En appliquant la formule précédente, on obtient la valeur de U_n correspondant à ce choix initial.
D'après la formule, cette valeur de U_n est le contenu de la première colonne de A^n.

► Déterminer A^5, A^{10} et A^{20}. Que remarque-t-on ?

On remarque que la suite des puissances itérées (A^n) semble converger vers une matrice proche de :

$$\begin{pmatrix} 0.27 & 0.27 & 0.27 \\ 0.5 & 0.5 & 0.5 \\ 0.23 & 0.23 & 0.23 \end{pmatrix}$$

II. Simulation des trajectoires en Scilab

II.1. Évolution de l'état du hamster partant d'un état x_0 fixé

Dans ce qui suit, et sauf mention du contraire, on fait l'hypothèse que le hamster se trouve initialement dans l'état $x_0 = 2$. Autrement dit : $\mathbb{P}([X_0 = x_0]) = 1$.

▶ Que vaut U_0 dans ce cas ? Et U_1 ?

Dans ce cas, on a $U_0 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ et $U_1 = A \times U_0 = \begin{pmatrix} 0.3 \\ 0.4 \\ 0.3 \end{pmatrix}$

▶ Faire le lien entre U_1 et A. En **Scilab**, quel appel sur A permet de récupérer U_1 ?

U_1 est la deuxième colonne de A.
On récupère cette colonne à l'aide de l'appel : `A(:,2)`.

▶ Rappeler le support de X_1. Décrire le comportement que doit avoir une fonction simulant X_1.

$X_1(\Omega) = [\![1, 3]\!]$. Une fonction simulant X_1 doit renvoyer :

- 1 avec probabilité 0.3 ($= A(1, 2)$),
- 2 avec probabilité 0.4 ($= A(2, 2)$),
- 3 avec probabilité 0.3 ($= A(3, 2)$).

▶ Écrire, à l'aide de la fonction `rand`, une fonction `simuX1` (sans paramètre d'entrée et dont la sortie est stocké dans une variable `x1`) permettant de simuler la v.a.r. X_1 dans le cas où $x_0 = 2$.
On testera la fonction une dizaine de fois et on commentera le résultat obtenu.

```
function x1 = simuX1()
    r = rand()
    if r < 0.3 then
        x1 = 1
    elseif r < 0.7 then
        x1 = 2
    else // r < 1
        x1 = 3
    end
endfunction
```

- Lorsque l'on exécute une dizaine de fois la fonction `simuX1`, elle renvoie à peu près le même nombre de 1, de 2 et de 3.
- En vertu de la LfGN, il faudrait exécuter un grand nombre de fois cette fonction pour vérifier si elle simule correctement la v.a.r. X_1. Par exemple, si on exécute $N = 10000$ fois cette fonction, on obtiendra environ 3000 fois le nombre 1, 4000 fois le nombre 2 et 3000 fois le nombre 3.

▶ Modifier cette fonction afin de l'adapter au cas où x_0 est quelconque.
On prendra la valeur `x0` en paramètre ainsi que la matrice de transition `A`.

```
function x1 = simuX1(x0, A)
    r = rand()
    if r < A(1,x0) then
        x1 = 1
    elseif r < A(1,x0) + A(2,x0) then
        x1 = 2
    else // r < A(1,x0) + A(2,x0) + A(3,x0)
        x1 = 3
    end
endfunction
```

On peut faire le même commentaire que dans le cas de la fonction précédente. On gardera en tête le lien fort lien simulation informatique et LfGN.

II.2. Parenthèse : simulation d'une v.a.r. suivant une loi finie quelconque

Dans la section précédente, on a codé une fonction permettant de simuler la v.a.r. X_1. On cherche ici à écrire une fonction permettant de simuler une v.a.r. Y discrète finie quelconque.
On note $Y(\Omega) = \{y_1, \dots, y_n\}$ et pour tout $k \in [\![1, n]\!]$, $p_n = \mathbb{P}([Y = k])$.

▶ Que vaut $p_1 + \dots + p_n$?

La famille $([Y = k])_{k \in [\![1,n]\!]}$ est un système complet d'événements.
On en déduit que : $\sum_{k=1}^{n} \mathbb{P}([Y = k]) = 1$.

▶ Décrire le comportement que doit avoir une fonction simulant Y.

Une fonction simulant Y doit renvoyer :

- y_1 avec probabilité p_1,
- y_2 avec probabilité p_2,
- ...
- ...
- y_n avec probabilité p_n.

▶ Comment coder une telle fonction à l'aide de la fonction `rand` ?

Pour ce faire, on effectue un tirage `r = rand()` et on teste si `r` est dans l'intervalle :

- $[0, p_1]$ (cela se produit avec probabilité p_1),
- $]p_1, p_1 + p_2]$ (cela se produit avec probabilité p_2),
- $]p_1 + p_2, p_1 + p_2 + p_3]$ (cela se produit avec probabilité p_3),
- ...
- $]p_1 + \dots + p_{n-1}, p_1 + \dots + p_n]$ (cela se produit avec probabilité p_n).

On rappelle que l'appel `cumsum(P)` permet d'obtenir $[p_1, p_1 + p_2, \dots, p_1 + \dots + p_n]$.

▶ Coder la fonction `simuY` qui prend en paramètre une matrice ligne `Y` contenant les valeurs de $Y(\Omega)$ et une matrice ligne `P` contenant les probabilités associées.
Cette fonction doit mettre en œuvre la stratégie précédente.

```
function res = simuY(Y, P)
    r = rand()
    tabProbCumul = cumsum(P)
    k = 1
    while tabProbCumul(k) < r
        k = k + 1
    end
    res = Y(k)
endfunction
```

▶ Écrire une fonction `simuMarkovEtape` qui prend en paramètre l'état initial `x0` du hamster et la matrice `A` et renvoie un état possible du hamster (avec la bonne probabilité) au bout d'une heure.

```
function x = simuMarkovEtape(x0, A)
    x = simuY(1:3, A(:,x0))
endfunction
```

II.3. Évolution de l'état du hamster après n heures

▶ Écrire une fonction `simuMarkov(x0,A,n)` qui prend en paramètre l'état initial `x0` et la matrice de transition `A` et renvoie une simulation de la v.a.r. X_n.

Il s'agit simplement d'itérer la fonction précédente.

```
function x = simuMarkov(x0, A, n)
    x = x0
    for i = 1:n
        x = simuMarkovEtape(x, A)
    end
endfunction
```

II.4. Simulation d'une trajectoire

- Pour $\omega \in \Omega$, on appelle **trajectoire** de taille n la suite finie $(X_0(\omega), X_1(\omega), \ldots, X_n(\omega))$.
- Simuler une trajectoire de taille n c'est donc obtenir une suite $(x_0, x_1, \ldots, x_n)$ où x_i est le résultat de la simulation de X_i.

▶ Écrire une fonction `simuTrajectoire(x0, P, n)` qui simule une trajectoire de taille `n`.

```
function tab = simuTrajectoire(x0, P, n)
    tab = zeros(1, n+1)
    tab(1) = x0
    for i = 1:n
        tab(i+1) = simuMarkovEtape(tab(i), P)
    end
endfunction
```

Fonctionnalité Scilab

La simulation d'une trajectoire de taille `n` est implémentée dans **Scilab**.
Plus précisément, la commande `grand(n,'markov', P',x0)`

permet de simuler une trajectoire ayant les caracatéristiques suivantes :

- × la trajectoire est de taille `n`,
- × elle démarre de l'état initial `x0`.

(où `P` est la matrice de transition de la chaîne de Markov)
On notera que l'on considère la transposée de la matrice `P` dans cet appel.

Il faut faire attention lorsque l'on utilise l'appel `grand(n,'markov',P',x0)` :

- × la trajectoire obtenue ne contient pas `x0`,
- × les états sont nommés 1, 2, …, m où m est le nombre de lignes de la matrice carrée P.

▶ Par quel appel utilisant la fonction `grand` peut-on obtenir le même type de simulation que l'appel `simuTrajectoire(x0, P, n)` ?

```
[x0, grand(n,'markov',P',x0)]
```

II.5. Représentation de trajectoires

▶ Étant donné un état initial `x0` et une population de taille `N`, quel appel permet de représenter une trajectoire simulée de taille `n` donnée ? On utilisera la fonction `plot2d`.

```
plot2d((0:n), simuTrajectoire(x0, transitionP(N), n))
```

▶ Écrire un programme :

- × qui demande à l'utilisateur d'entrer la taille des trajectoires à simuler et stocke cette information dans une variable `tailleTraj`,
- × qui demande à l'utilisateur d'entrer l'état initial et le stocke dans une variable `x0`,
- × qui calcule une trajectoire de longueur `tailleTraj` partant de l'état initial `x0`,
- × qui représente cette trajectoire à l'aide de la commande `plot2d`.

```
1 tailleTraj = input("Entrez la taille souhaitée pour la trajectoire : ")
2 x0 = input("Entrez l''état initial de la chaîne de Markov : ")
3 P = transitionP(N)
4 plot2d((0:tailleTraj), simuTrajectoire(x0, P, tailleTraj))
```

▶ Comment modifier ce programme afin qu'il permette l'affichage de plusieurs trajectoires ?
On demandera initialement à l'utilisateur d'entrer le nombre de trajectoires souhaitées (à stocker dans une variable `nbTraj`).
(indication : on pourra effectuer une boucle)

```
1 N = input("Entrez la taille N de la population : ")
2 nbTraj = input("Entrez le nombre de trajectoires souhaitées : ")
3 tailleTraj = input("Entrez la taille souhaitée pour les trajectoires : ")
4 x0 = input("Entrez l''état initial de la chaîne de Markov : ")
5 P = transitionP(N)
6 for k = 1:nbTraj
7     plot2d((0:tailleTraj), simuTrajectoire(x0, P, tailleTraj), style=k)
8 end
```

III. Comportement asymptotique du hamster

On souhaite conjecturer le comportement de la loi de X_n lorsque n tend vers $+\infty$.
Pour ce faire, on compare :

× la valeur théorique de U_n (loi de X_n) pour n grand,

× une valeur approchée de U_n obtenue en recueillant les effectifs de chaque état à la suite de N simulations de trajectoires de taille n.

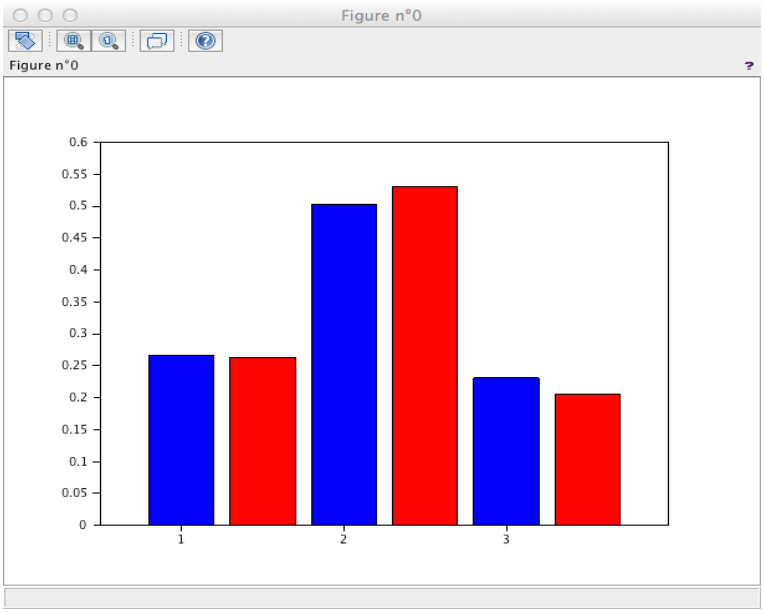
▶ Partant de l'état initial `x0`, par quel appel obtient-on U_n ?

> On stocke tout d'abord le résultat de A^n dans une matrice B : `B = A^n`
> et on sélectionne la colonne correspondant à l'état `x0` : `B(:,x0)`

▶ Compléter le programme suivant.

```
1  // Valeur des paramètres
2  N = input('Entrez le nombre N de simulations souhaitées : ')
3  n = input('Entrez la taille n de chaque trajectoire simulée : ')
4  x0 = input('Entrez l''état initial : ')
5
6  // Distribution théorique
7  B = A^n
8  P = B(:,x0)
9
10 // Valeurs observées
11 Obs = zeros(1, N)
12 for i = 1:N
13     Obs(i) = simuMarkov(x0, A, n)
14 end
15
16 // Calcul des effectifs observés
17 v = tabul(Obs)
18
19 clf()
20 // Diagramme des fréquences observées
21 bar(v(:, 1) + 0.5, v(:,2)/N, width = 0.4, 'red')
22 // Diagramme de la distribution théorique
23 bar(1:3, P, width = 0.4)
```

- Exécutez ce programme pour `N = 1000`, `n = 50`, et `x0 = 2`.
  On obtient le diagramme suivant :



- Quel diagramme obtient-on si l'on part intialement avec un autre état `x0` ?

> On obtient les mêmes diagrammes en partant des états initiaux 1 et 3.

- Que peut-on en conclure sur le comportement à terme du hamster ?

> À terme, le hamster mange avec une probabilité d'environ 0.27, dort avec une probabilité d'environ 0.5 et tourne dans sa roue avec une probabilité d'environ 0.23.

- Si l'on est capable de démontrer que la suite (U_n) converge vers une limite notée U_∞, quelle relation peut-on établir entre U_∞ et A ?

> Si l'on est capable de montrer que U_n converge (*i.e.* si tous ses cœfficients convergent) et si on note U_∞ sa limite, l'égalité $U_{n+1} = A \times U_n$ nous fournit, par passage à la limite :
>
> $$U_\infty = A \times U_\infty$$
>
> Une telle loi U_∞ est dite **stationnaire** (ou **invariante**).
>
> On peut démontrer les propriétés suivantes.
>
> - Une loi invariante est un vecteur propre de la matrice de transition A.
> - Une chaîne de Markov homogène sur un espace d'étas fini S admet au moins une loi invariante.
> - Avec une hypothèse supplémentaire (*la chaîne de Markov est irréductible*) on peut démontrer qu'une telle chaîne de Markov admet une unique loi invariante.

IV. Les chaînes de Markov aux concours (EDHEC 2017)

L'épreuve EDHEC 2017 portait sur le déplacement au cours du temps d'un mobile sur les 4 sommets d'un carré. Voici une retranscription de l'énoncé.

Les sommets d'un carré sont numérotés 1, 2, 3, et 4 de telle façon que les côtés relient le sommet 1 au sommet 2, le sommet 2 au sommet 3, le sommet 3 au sommet 4 et le sommet 4 au sommet 1.
Un mobile se déplace aléatoirement sur les sommets de ce carré selon le protocole suivant :

- *Au départ, c'est à dire à l'instant* 0*, le mobile est sur le sommet 1.*
- *Lorsque le mobile est à un instant donné sur un sommet, il se déplace à l'instant suivant sur l'un quelconque des trois autres sommets, et ceci de façon équiprobable.*

Pour tout $n \in \mathbb{N}$, on note X_n la variable aléatoire égale au numéro du sommet sur lequel se situe le mobile à l'instant n. D'après le premier des deux points précédents, on a donc $X_0 = 1$.

IV.1. Les résultats de l'énoncé

Voici les principaux résultats qu'il fallait démontrer :

1) $\forall n \in \mathbb{N},\ \mathbb{P}([X_{n+1} = 1]) = \dfrac{1}{3}\left(\mathbb{P}([X_n = 2]) + \mathbb{P}([X_n = 3]) + \mathbb{P}([X_n = 4])\right)$

puis : $\forall n \in \mathbb{N},\ \mathbb{P}([X_{n+1} = 1]) = -\dfrac{1}{3}\,\mathbb{P}([X_n = 1]) + \dfrac{1}{3}$

et enfin : $\forall n \geqslant 2,\ \mathbb{P}([X_n = 1]) = \dfrac{1}{4} + \dfrac{3}{4}\left(-\dfrac{1}{3}\right)^n$

2) $\forall n \in \mathbb{N},\ \mathbb{P}([X_{n+1} = 2]) = \dfrac{1}{3}\left(\mathbb{P}([X_n = 1]) + \mathbb{P}([X_n = 3]) + \mathbb{P}([X_n = 4])\right)$

3) $\forall n \in \mathbb{N},\ \mathbb{P}([X_{n+1} = 3]) = \dfrac{1}{3}\left(\mathbb{P}([X_n = 1]) + \mathbb{P}([X_n = 2]) + \mathbb{P}([X_n = 4])\right)$

4) $\forall n \in \mathbb{N},\ \mathbb{P}([X_{n+1} = 4]) = \dfrac{1}{3}\left(\mathbb{P}([X_n = 1]) + \mathbb{P}([X_n = 2]) + \mathbb{P}([X_n = 3])\right)$

On considérait alors, pour tout n de $\mathbb{N}$, la matrice-ligne $U_n \in \mathcal{M}_{1,4}(\mathbb{R})$ et la matrice A :

$$U_n = \begin{pmatrix}\mathbb{P}([X_n = 1]) & \mathbb{P}([X_n = 2]) & \mathbb{P}([X_n = 3]) & \mathbb{P}([X_n = 4])\end{pmatrix} \quad \text{et} \quad A = \frac{1}{3}\begin{pmatrix}0 & 1 & 1 & 1\\ 1 & 0 & 1 & 1\\ 1 & 1 & 0 & 1\\ 1 & 1 & 1 & 0\end{pmatrix}$$

(ce choix correspond à prendre la transposée du vecteur U_n choisi dans ce TP)

Cela permettait d'obtenir les résultats suivants.

5) $\forall n \in \mathbb{N},\ U_{n+1} = U_n\ A$

6) $\forall n \in \mathbb{N},\ U_n = U_0\ A^n$

► En déduire la première ligne de A^n.

> La première ligne de A^n est obtenue en prenant $U_0 = \begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}$.

► Expliquer comment choisir la position du mobile au départ pour trouver les trois autres lignes de la matrice A^n.

> Les lignes suivantes de A^n sont obtenues en prenant successivement :
> × $U_0 = \begin{pmatrix}0 & 1 & 0 & 0\end{pmatrix}$,
> × $U_0 = \begin{pmatrix}0 & 0 & 1 & 0\end{pmatrix}$,
> × $U_0 = \begin{pmatrix}0 & 0 & 0 & 1\end{pmatrix}$.

On écrivait alors A sous la forme : $A = -\frac{1}{3}\ I + \frac{1}{3}\ J$ avec les matrices I et J suivantes :

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{et} \quad J = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

et on déterminait $A^n = \left(-\frac{1}{3}\ I + \frac{1}{3}\ J\right)^n$ par la formule du binôme de Newton.

IV.2. La partie informatique

► Compléter le script **Scilab** suivant pour qu'il affiche les 100 premières positions autres que celle d'origine, du mobile dont le voyage est étudié dans ce problème, ainsi que le nombre n de fois où il est revenu sur le sommet numéroté 1 au cours de ses 100 premiers déplacements (on pourra utiliser la commande `sum`).

```
1 A = [---] /3
2 x = grand(100,'markov',A,1)
3 n = ---
4 disp(x)
5 disp(n)
```

```
1 A = [0, 1, 1, 1; 1, 0, 1, 1; 1, 1, 0, 1; 1, 1, 1, 0] /3
2 x = grand(100,'markov',A,1)
3 n = sum(x==1)
4 disp(x)
5 disp(n)
```

► Après avoir exécuté cinq fois ce script, les réponses concernant le nombre de fois où le mobile est revenu sur le sommet 1 sont : $n = 23$, $n = 28$, $n = 23$, $n = 25$, $n = 26$.
En quoi est-ce normal ?

À chaque fois que 100 déplacements sont effectués, on note environ 25 retours en position 1. Cela correspond à la formule obtenue précédemment :

$$\mathbb{P}([X_m = 1]) = \frac{1}{4} + \frac{3}{4}\ \left(-\frac{1}{3}\right)^m$$

avec $\left(-\frac{1}{3}\right)^m$ qui se rapproche rapidement de 0.