

Devoir surveillé du Mardi 7 Janvier

La calculatrice est interdite. Durée : 1h45

Dans tout ce devoir, on suppose avoir importé sur Python les bibliothèques :

- `numpy` avec le raccourci `np` ;
- `numpy.random` avec le raccourci `rd` ;
- `numpy.linalg` avec le raccourci `al` ;
- `matplotlib.pyplot` avec le raccourci `plt`.

Exercice 1

Dans cet exercice, on propose trois méthodes algorithmiques pour le calcul de la racine carrée de 2.

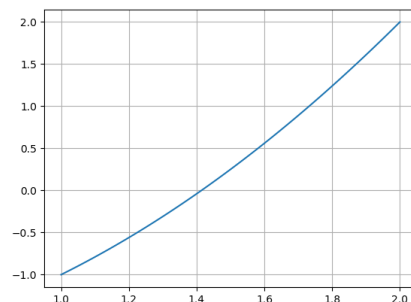
Méthode par balayage

On considère la fonction $f : x \in [1, 2] \mapsto x^2 - 2$.

1. Écris un script, incluant la définition de la fonction f par l'en-tête `def f(x)`, qui renvoie la courbe représentative de la fonction f sur l'intervalle $[1, 2]$.

Après exécution, ce script génère le graphe ci-contre.

On constate que f est strictement croissante sur $[1, 2]$, et s'annule en changeant de signe en $\sqrt{2}$.



2. On considère le script suivant :

```

1 | def balayage(p):
2 |     v = np.linspace(1,2,10**p+1)
3 |     i = 0
4 |     while f(v[i])<0 :
5 |         i=i+1
6 |     return v[i]
```

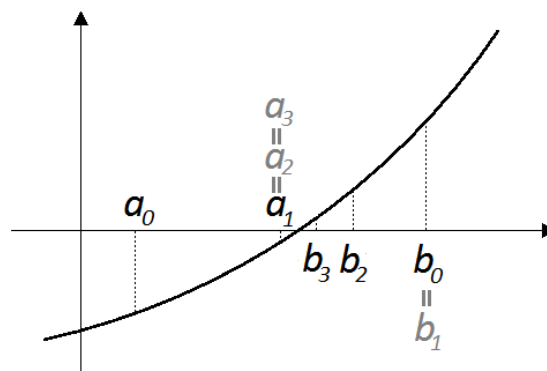
Que renvoie la commande `balayage` ? On justifiera sa réponse en expliquant le fonctionnement du script ci-dessus.

Méthode de dichotomie

Rappelons la méthode de dichotomie.

On construit une suite d'intervalles $[a_n, b_n]$ contenant $\sqrt{2}$. Pour cela, on définit trois suites $(a_n)_{n \in \mathbb{N}}$, $(b_n)_{n \in \mathbb{N}}$, $(m_n)_{n \in \mathbb{N}^*}$ par $a_0 = 1$, $b_0 = 2$ et pour tout $n \in \mathbb{N}$, $m_{n+1} = \frac{a_n + b_n}{2}$ et

- si $f(a_n)f(m_{n+1}) \leq 0$, alors $\begin{cases} a_{n+1} = a_n \\ b_{n+1} = m_{n+1} \end{cases}$,
- si $f(m_{n+1})f(b_n) \leq 0$, alors $\begin{cases} a_{n+1} = m_{n+1} \\ b_{n+1} = b_n \end{cases}$.



3. Compléter la fonction suivante, qui prend comme paramètre d'entrée un entier n , et qui renvoie les valeurs de a_n et b_n .

```

1 | def suites(n):
2 |     a = 1
3 |     b = 2
4 |     for ..... :
5 |         m = .....
6 |         if ..... :
7 |             b = m
8 |         else :
9 |             a = .....
10 |    return a, b

```

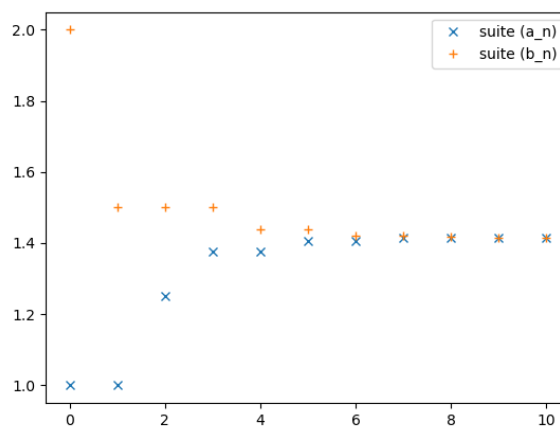
4. On souhaite étudier le comportement des suites (a_n) et (b_n) , ou tout du moins de leurs premiers termes.

Écrire pour cela un script renvoyant un graphique sur lequel sont représentés les points (n, a_n) et (n, b_n) pour $n \in \llbracket 0, 10 \rrbracket$.

On obtient le graphe ci-contre.

Les suites (a_n) et (b_n) semblent adjacentes, et converger vers la même limite ℓ . On admettra que $\ell = \sqrt{2}$, et que pour tout $n \in \mathbb{N}$:

$$a_n \leq \sqrt{2} \leq b_n \quad \text{et} \quad |\sqrt{2} - a_n| \leq |b_n - a_n|.$$



5. Écrire une fonction d'en-tête `def dichotomie(eps)`, qui prend comme paramètre d'entrée un réel $\varepsilon > 0$, et qui renvoie une valeur approchée de $\sqrt{2}$ à ε près.

Méthode de Newton

On considère la suite (x_n) définie par $x_0 = 1$ et pour tout $n \geq 0$:

$$x_{n+1} = \frac{1}{2} \left(x_n + \frac{2}{x_n} \right).$$

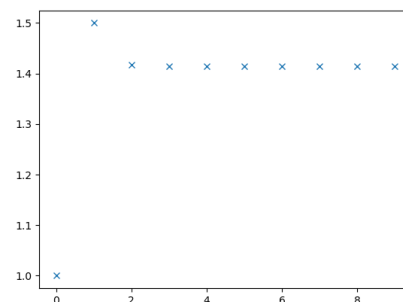
On définit la fonction $g : x \in [1, 2] \mapsto \frac{1}{2} \left(x + \frac{2}{x} \right)$.

6. Écrire une fonction d'en-tête `def suite(n)` qui prend comme paramètre d'entrée un entier n et calcule la valeur de x_n correspondante.
7. On souhaite étudier le comportement de la suite (x_n) , ou tout du moins de ces premiers termes.

Écrire pour cela un script renvoyant un graphique sur lequel se trouvent les points (n, x_n) pour $n \in \llbracket 0, 10 \rrbracket$.

On obtient le graphe ci-contre.

La suite (x_n) semble décroissante à partir du rang 1, et converger vers une limite finie ℓ appartenant à $[1, 2]$. Nous admettrons ce résultat sans démonstration dans la suite.



8. Déterminer la limite ℓ .

9. On admet que : $\forall n \in \mathbb{N}, |x_n - \sqrt{2}| \leq 3 \times \left(\frac{1}{2}\right)^{2^n}$.

Écrire une fonction d'en-tête `def newton(eps)`, qui prend comme paramètre d'entrée un réel $\varepsilon > 0$, et qui renvoie une valeur approchée de $\sqrt{2}$ à ε près.

Exercice 2

Soient a, b deux entiers strictement positifs. Une urne contient initialement a boules rouges et b boules blanches. On effectue une succession d'épreuves, chaque épreuve étant constituée des trois étapes suivantes :

- on pioche une boule au hasard dans l'urne,
- on replace la boule tirée dans l'urne,
- on rajoute dans l'urne une boule de la même couleur que celle qui vient d'être piochée.

Après n épreuves, l'urne contient donc $a + b + n$ boules. Pour tout $n \in \mathbb{N}^*$, on note X_n le nombre de boules rouges qui ont été **ajoutées** dans l'urne (par rapport à la composition initiale) à l'issue des n premières épreuves.

On souhaite simuler l'expérience grâce à **Python**.

1. Compléter la fonction **Python** suivante, qui simule le tirage d'une boule dans une urne contenant x boules rouges et y boules blanches et qui retourne la valeur 0 si la boule est rouge et 1 si elle est blanche.

```

1 | def tirage(x, y):
2 |     r = rd.random()
3 |     if ..... :
4 |         return(0)
5 |     else:
6 |         return(1)

```

2. Compléter la fonction **Python** suivante, qui simule n tirages successifs dans une urne contenant initialement a boules rouges et b boules blanches (selon le protocole décrit ci-dessus) et qui retourne la valeur de X_n :

```

1 | def experience(a, b, n):
2 |     x = a
3 |     y = b
4 |     for k in range(n):
5 |         r = tirage(x,y)
6 |         if r == 0 :
7 |             x = .....
8 |         else:
9 |             .....
10 |     return( ..... )

```

3. Compléter la fonction **Python** suivante qui fait appel m fois à la fonction précédente pour estimer la loi de X_n . Le paramètre de sortie sera un vecteur contenant les approximations de $P(X_n = 0)$, $P(X_n = 1), \dots, P(X_n = n)$.

```

1 | def simulation(a, b, n, m):
2 |     loi = np.zeros(n+1)
3 |     for k in range(m):
4 |         r = .....
5 |         loi[r] = .....
6 |     return( ..... )

```

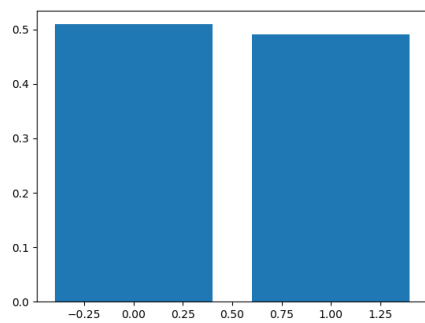
4. On s'intéresse ici au cas où $a = b = 1$. On utilise la fonction `simulation` avec des valeurs de n entre 1 et 5 et on affiche à chaque fois l'estimation de la loi de X_n sous forme d'un diagramme en bâton.

```

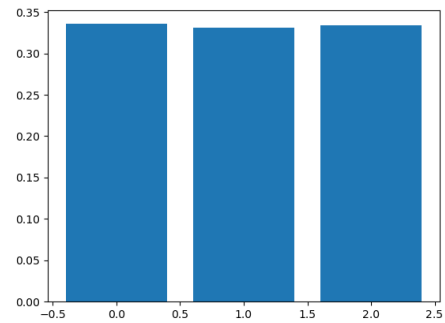
1 | n = int(input('Donner n : '))
2 | plt.bar(np.arange(0, n+1), simulation(1, 1, n, 100000))
3 | plt.show()

```

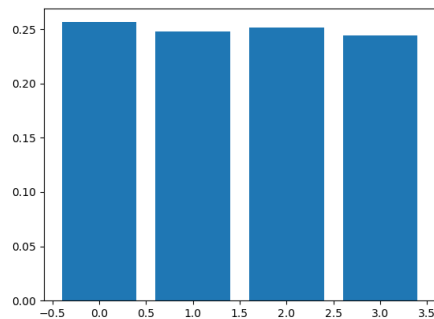
- Pour $n = 1$, on obtient :



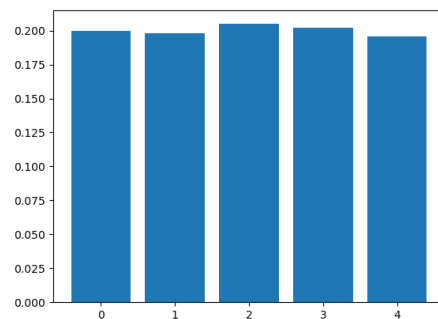
- Pour $n = 2$, on obtient :



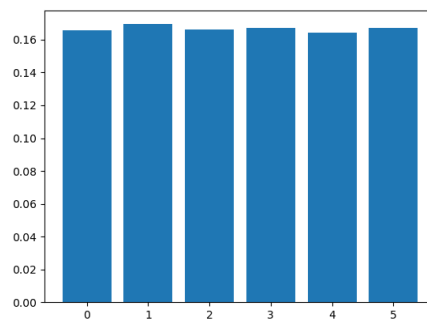
- Pour $n = 3$, on obtient :



- Pour $n = 4$, on obtient :



- Pour $n = 5$, on obtient :



À l'aide de ces résultats, conjecturer la loi de X_n . Justifier votre réponse.

Exercice 3

A l'abord d'une élection opposant deux candidats A et B , une société d'études statistiques réalise une série de sondages auprès de la population composées de n individus.

A chaque sondage, la société choisit un échantillon de taille N dans la population. Si i électeurs sur les N interrogés se sont déclarés en faveur du candidat A (et donc $N - i$ en faveur de B), on constate qu'après ce sondage, la population totale s'aligne parfaitement sur le choix de l'échantillon.

Désormais, une proportion $p = \frac{i}{N}$ de la population totale est en faveur de A (et $1 - p$ en faveur de B). On suppose que ce phénomène se répète à chaque nouveau sondage.

On suppose que, lors du premier sondage, n_0 électeurs sur les N de l'échantillon sont en faveur de A .

Nous admettrons que le rapport entre la taille de la population totale n et celle de l'échantillon N est assez grand pour qu'un sondage s'apparente à un tirage **avec** remise (une personne peut être interrogée plusieurs fois).

Pour tout entier k supérieur ou égale à 1, on note X_k la variable aléatoire égale au nombre d'électeurs se déclarant en faveur du candidat A lors du k -ième sondage suivant le sondage initial, et on pose X_0 la variable certaine égale à n_0 .

Ainsi définie, la suite $(X_k)_{k \in \mathbb{N}}$ est une chaîne de Markov homogène.

1. (a) Soit $i \in \llbracket 0, N \rrbracket$ et $k \in \mathbb{N}$.

Reconnaître la loi conditionnelle de X_{k+1} , sachant que $(X_k = i)$ est réalisé. Donner alors l'expression de $P_{(X_k=i)}(X_{k+1} = j)$ pour tout $j \in \llbracket 0, N \rrbracket$.

- (b) En déduire comment compléter la fonction suivante pour que les composantes de la liste L retournée en sortie simulent les valeurs de $X_1, X_2, \dots, X_k$, pour des valeurs k , N et n_0 entrées par l'utilisateur.

```

1 | def simulX(k, N, n0):
2 |     X = n0
3 |     L = []
4 |     for i in range(k):
5 |         X = .....
6 |         L.append(X)
7 |     return(L)

```

- (c) On exécute 4 fois la fonction `simulX` pour $k = 20$, $N = 10$ et différentes valeurs de n_0 :

- Pour $n_0 = 3$:

```

[6, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[3, 5, 8, 7, 5, 8, 9, 8, 6, 6, 6, 6, 5, 4, 7, 8, 9, 9, 10, 10]
[4, 4, 2, 2, 3, 5, 3, 5, 2, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[6, 8, 3, 2, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

```

- Pour $n_0 = 5$:

```

[8, 7, 6, 6, 8, 7, 8, 9, 8, 8, 9, 9, 8, 9, 9, 9, 10, 10, 10, 10]
[5, 5, 4, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[4, 3, 1, 2, 3, 6, 4, 5, 3, 1, 2, 4, 6, 5, 7, 5, 5, 3, 4, 7]
[6, 7, 9, 7, 8, 5, 6, 7, 5, 7, 8, 8, 7, 8, 8, 8, 10, 10, 10, 10]

```

- Pour $n_0 = 8$:

```

[9, 9, 8, 7, 9, 6, 6, 7, 7, 9, 8, 9, 10, 10, 10, 10, 10, 10, 10, 10]
[3, 5, 8, 7, 5, 8, 9, 8, 6, 6, 6, 6, 5, 4, 7, 8, 9, 9, 10, 10]
[6, 8, 3, 2, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[9, 9, 7, 8, 6, 6, 8, 8, 9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10]

```

Commenter les résultats obtenus.

2. (a) Rappeler la valeur de $\binom{j}{i}$ si $i > j$ et vérifier que si $i \leq j$, alors $\binom{j}{i} = \prod_{k=0}^{i-1} \frac{j-k}{i-k}$.
- (b) Dédire de la question précédente comment compléter le script de la fonction `C` suivante de telle sorte que l'appel de `C(i,j)` renvoie $\binom{j}{i}$:

```

1 | def C(i,j):
2 |     if i>j:
3 |         return 0
4 |     else:
5 |         aux = 1
6 |         for k in range(i):
7 |             aux = .....
8 |         return aux

```

- (c) En déduire des instructions permettant d'affecter à une variable `M` la matrice M de transition associée à la chaîne de Markov (X_k) .
- (d) Justifier que la chaîne n'admet pas un unique état stable.
3. On prend de nouveau $N = 10$ et on considère les instructions suivantes, exécutées pour les valeurs successives de n_0 égales à 3, 5 et 8 :

```

1 | n0 = int(input("Entrer la valeur de n0 : "))
2 | S = []
3 | for j in range(10000):
4 |     X = simulX(100, 10, n0)
5 |     S.append(X[-1])
6 | loiemp = [S.count(i)/10000 for i in range(11)]
7 | print(loiem)
8 | V0 = np.zeros(11)
9 | V0[n0] = 1
10 | V100 = np.dot(V0, al.matrix_power(M, 100))
11 | print(V100)

```

Entrer la valeur de n0 : 3

```
[0.7017, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.2983]
[0.7, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.3]
```

Entrer la valeur de n0 : 5

```
[0.5071, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.4929]
[0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5]
```

Entrer la valeur de n0 : 8

```
[0.1971, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.8028]
[0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.8]
```

- (a) Que contient la variable `S` (ligne L5) à l'issue des 10000 passages dans la boucle `for` ?
- (b) Que contient le vecteur `loiemp` (ligne L6) ?
- (c) Que contient le vecteur `V100` (ligne L10) ?
- (d) Quel constat les réponses renvoyées par `Python` quant à la simulation de la variable X_k proposée à la question 1.(c) permettent-elles de faire ? Que peut-on conjecturer quant à la loi limite de la suite (X_k) ?

4. On suppose maintenant que $N = 3$ et $n_0 = 1$. On note T la variable aléatoire égale au plus petit entier k tel que X_k prend la valeur 0 ou 3.

(a) Compléter les instructions pour que le vecteur T contienne 10000 simulations de T .

```

1 | N = 3
2 | n0 = 1
3 | T = []
4 | for i in range(10000):
5 |     t = 0
6 |     X = n0
7 |     while ..... :
8 |         t = t+1
9 |         X = .....
10 |    T.append(t)

```

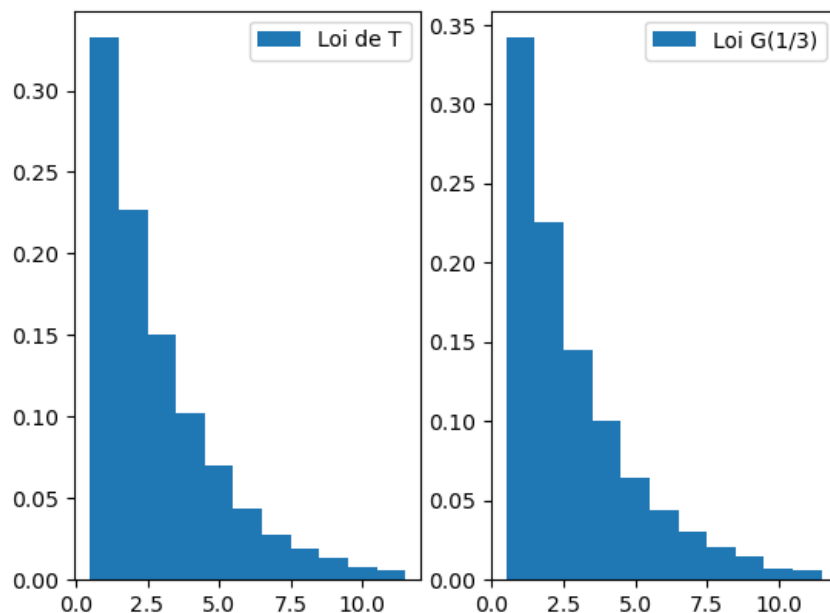
(b) On ajoute à ces instructions les suivantes :

```

1 | c = np.arange(0.5, 12.5)
2 | plt.subplot(1, 2, 1)
3 | plt.hist(T, c, density=True, label="Loi de T")
4 | plt.legend()
5 |
6 | plt.subplot(1, 2, 2)
7 | plt.hist(rd.geometric(1/3, 10000), c, density='True',
8 | label="Loi G(1/3)")
9 | plt.legend()
10 | plt.show()

```

On donne le résultat qu'elles permettent d'afficher :



Que fait ce programme ? Que représente chacun des deux graphiques ? Que peut-on conjecturer pour la loi de T ? Justifier.

Exercice 4

On dispose d'une base de données comportant deux tables **vehicule** et **annonce** décrites ci-dessous.

- La table **vehicule** recense des informations sur les modèles de véhicules en vente sur le marché. Elle est composée des attributs suivants.
 - **id.vehicule** (de type **INTEGER**) : un code permettant d'identifier de façon unique chaque référence de véhicule (marque et modèle).
 - **marque** (de type **TEXT**) : le nom du constructeur du véhicule.
 - **modele** (de type **TEXT**) : le modèle du véhicule, un constructeur proposant en général plusieurs modèles de véhicules à la vente.
 - **prix.neuf** (de type **INTEGER**) : prix de vente du véhicule neuf.
- La table **annonce** regroupe des informations sur un grand nombre d'annonces de véhicules d'occasion. Chaque enregistrement correspond à une annonce et possède les attributs suivants.
 - **id.annonce** (de type **INTEGER**) : un code permettant d'identifier chaque annonce de façon unique.
 - **id.vehicule** (de type **INTEGER**) : l'identifiant du modèle de véhicule vendu, qui correspond à l'identifiant utilisé dans la table **vehicule**.
 - **annee** (de type **INTEGER**) : année de première mise en circulation du véhicule.
 - **km** (de type **INTEGER**) : nombre de kilomètres parcourus par le véhicule au moment de la revente.
 - **prix.occasion** (de type **INTEGER**) : prix de vente du véhicule d'occasion.

1. En justifiant brièvement, identifier une clef primaire dans chacune des tables **vehicule** et **annonce**, ainsi qu'une clef étrangère dans la table **annonce**.
2. Donner le schéma relationnel associé aux tables **vehicule** et **annonce**.
3. Écrire une requête SQL permettant d'extraire les noms de tous les modèles de véhicules mis en vente par le constructeur **Dubreuil Motors**.
4. Expliquer le fonctionnement de la requête SQL suivante et préciser l'effet éventuel de cette requête sur chacune des tables **vehicule** et **annonce**.

```
UPDATE annonce
SET prix.occasion = prix.neuf
FROM vehicule
WHERE vehicule.id.vehicule = annonce.id.vehicule
      AND vehicule.prix.neuf < annonce.prix.occasion
```

5. A l'aide d'une jointure, écrire une requête SQL permettant d'obtenir, sur une même table, la liste de toutes les annonces de la table **annonce** avec les attributs suivants :
 - l'identifiant de l'annonce **id.annonce**.
 - le kilométrage **km**,
 - le prix de vente du véhicule neuf **prix.neuf**,
 - le prix de l'annonce d'occasion **prix.occasion**.