ECS1 : Initiation à Scilab

Alain GUICHET

7 septembre 2020

Table des matières

Chapitre 1

L'univers Scilab

1.1 L'environnement de travail Scilab

1.1.1 Démarrage

Au lancement du programme (ouverture d'une session), on découvre dans la fenêtre : un navigateur de fichiers, une console, un tableau récapitulatif des variables utilisées, un historique des commandes. On peut obtenir d'autres fenêtres comme de l'aide (touche F1).

1.1.2 Travailler en mode console

Dans la console, le logiciel fonctionne comme une calculatrice et attend la saisie d'une commande derrière le « prompt » noté `-->`. Saisir successivement :

- `1+1` (puis validation, bien sûr)
- `x=3 ; (x+2)^3`
- `ans/1.6`
- `floor(ans)`

La variable `ans` contient donc le résultat du dernier calcul effectué. On peut aussi constater que le tableau des variables s'est enrichi de cette variable (cas mémoire de la machine). On peut écrire une succession de calculs sur une même ligne d'écriture, séparés par un point-virgule : seul le résultat du dernier calcul sera affiché.

- `clear` (efface toutes les variables existantes)

1.1.3 Travailler sous l'éditeur SciNotes

Pour écrire des petits programmes ou **scripts**, les sauvegarder et les modifier à sa guise, on utilise l'éditeur de texte intégré `SciNotes` (Applications/SciNotes). La fenêtre peut être intégrée à la fenêtre principale (maintenir le clic gauche enfoncé sur le nom du programme ouvert et déplacer sur la fenêtre principale à l'endroit souhaité).

Il est conseillé de commenter ses lignes de programme : en fin de ligne, écrire // suivi du message. On remarquera que les commentaires sont écrits en italique et en couleur verte. De manière générale, tout « mot » écrit dans l'éditeur peut voir sa couleur modifiée selon sa fonction dans la ligne de code : on parle de **coloration syntaxique**. C'est plutôt un mauvais signe que la couleur d'un mot usuel ne change pas (erreur de frappe).

Après la fin de l'écriture d'un script, on lance son exécution (sauvegarde nécessaire, le logiciel n'exécute que le fichier enregistré). **Il faut toujours tester un script avec des données numériques qui permettent de vérifier par des « calculs à la main » les résultats affichés à l'écran**.

1.2 Éléments du langage Scilab

1.2.1 Affectation

La gestion et l'utilisation des lieux de stockage de l'information est une des composantes essentielles de la programmation : il faut toujours prendre le temps de bien réfléchir à l'organisation des structures de données puisque cela va conditionner les méthodes de programmation qui seront utilisées ensuite.

Définition 1.1 (Affectation) :
Un lieu de stockage porte un nom chargé de faire le lien entre l'adresse mémoire de la machine et notre programme.
`lieu_de_stockage = expression_a_calculer // peut utiliser des contenus stockés`
Le contenu d'un lieu de stockage ne change pas tant qu'on ne demande pas à la machine de le modifier.

Attention à ne pas confondre le lieu de stockage avec le contenu de ce lieu.

Par exemple, que contient `U` en fin de code : `U=3 ; U=U ; U=2*U ; U=U+1` ?

1.2.2 Fonctions mathématiques et constantes prédéfinies

Définition 1.2 (Opérations usuelles) :
- Les opérations usuelles sont : `+`, `-`, `*`, `/` et `^`.
- Les fonctions usuelles sont : `log`, `exp`, `cos`, `sin`, `tan`, `sqrt`, `floor`, `abs`.
- Les constantes connues de `Scilab` sont : `%e`, `%pi`, `%i`.

Remarques :
- Bien d'autres fonctions préexistent dans `Scilab` mais ce sont quasiment les seules qui sont exigibles.
- Si la couleur d'écriture d'une fonction usuelle ne change pas, c'est qu'elle est mal orthographiée.
- Pour modifier le nombres de chiffres affichés, on utilise l'instruction `format(n)` avec $n \in [\![2, 25]\!]$.

1.2.3 Clavier et écran

Définition 1.3 (Instuctions de saisie et d'affichage) :
- Pour saisir un nombre entier ou réel, on utilise l'instruction **input**, avec ou sans message d'information :
  - `nom_de_variable = input() // pas de message à l'écran de saisie`
  - `nom_de_variable = input("Mon message") // avec un message`
- Pour afficher le résultat d'un calcul, on utilise l'instruction **disp** :
  - `disp(mon_calcul) // affiche le résultat du calcul`
  - `disp(mon_calcul_1,...,mon_calcul_n) // affiche les résultats mais dans l'ordre inverse`

Tester : `n=input() ; disp(n+1)`. Tester aussi : `n=input("n=") ; disp(sqrt(n))`.

1.2.4 Propositions

Définition 1.4 (Écriture des propositions) :
Les propositions sont écrites à l'aide d'opérateurs de comparaison et opérateurs de conjonction (par ordre décroissant de priorité, prioritaires sur les opérateurs de comparaison) :

	Comparaison						Conjonction		
Maths	$=$	$\neq$	$<$	$>$	$\leqslant$	$\geqslant$	non	et	ou
Scilab	==	<>	<	>	<=	>=	~	&	|

1.2.5 Conditions

Définition 1.5 (Instruction conditionnelle) :
Le langage de programmation de `Scilab` est doté d'une **instruction conditionnelle** qui permet de réaliser une ou plusieurs instructions lorsqu'une certaine proposition, appelée **condition**, est vraie mais qui peut aussi réaliser, au besoin, d'autre(s) instruction(s) lorsque cette même proposition est fausse :

```
if condition then // 'then' est facultatif
    liste_d_instructions_si_condition_est_vraie
end
```

(ou en une seule ligne `if condition then instruction ; end`). Dans le cas de la gestion d'une alternative :

```
if condition then
    liste_d_instructions_si_condition_est_vraie
else
    liste_d_instructions_si_condition_est_fausse
end
```

ou encore, lorsqu'il y a au moins trois alternatives :

```
if condition then
    liste_d_instructions_si_condition_est_vraie
elseif condition2 then // autant de elseif que nécessaire
    liste_d_instructions_si_condition_est_fausse_et_condition2_est_vraie
else
    liste_d_instructions_si_condition_est_fausse_et_condition2_est_fausse
end
```

Remarques :
- `condition` est une proposition dont le programme évaluera la véracité. Lorsqu'elle s'écrit par conjonction de plusieurs propositions, il ne faut pas hésiter à ajouter suffisamment de niveaux de parenthèses.

— Pour faciliter la (re)lecture du code (notamment pour la recherche d'erreurs éventuelles), il est conseillé de respecter les alignements/décalages automatiques proposés par l'éditeur `SciNotes`.

1.2.6 Boucles `for...end`

Définition 1.6 (Instruction itérative définie) :
Cette première **instruction itérative** permet de répéter une séquence d'instructions autant de fois qu'il y a de valeurs dans une certaine liste :

```
for compteur=[liste_des_valeurs]
    instruction_1
    ...
end
```

Elle peut s'écrire sur une unique ligne : `for compteur=[liste_des_valeurs] instruction ; ... ; end`.

Remarques :

— L'éditeur indente automatiquement les instructions pour les mettre en évidence, est sensible à la casse (il y a de modification de la couleur des mots importants si on écrit `FOR` à la place de `for`).

— La liste `[liste_des_valeurs]` est une matrice, en général une matrice-ligne, de réels ou d'entiers (assez souvent en progression arithmétique mais il n'y a nulle obligation) séparés par une virgule.
Exemples : `for k=[1,3,9,27] disp(sqrt(k)) ; end` ou encore `for k=[1:10] disp(k^2) ; end`.

1.2.7 Boucles `while...end`

Définition 1.7 (Instruction itérative indéfinie) :
Cette seconde **instruction itérative** permet de répéter une séquence d'instructions tant qu'une certaine condition, appelée **test d'arrêt**, est vraie et qui s'arrête dès que cette condition est fausse :

```
while condition
    instruction_1
    ...
end
```

Elle peut s'écrire sur une unique ligne : `while condition then instruction1 ; ... ; end`.

Remarques :

— `condition` **doit pouvoir changer** au cours de l'exécution de la boucle (sinon le programme est sans fin).

— Une telle boucle nécessite *une initialisation de la variable servant au test d'arrêt* puisque le test est effectué avant la première répétition de la boucle (la boucle peut ne jamais se réaliser)

— Pour faciliter la lecture du code et la recherche d'erreurs éventuelles, il est plus que souhaitable de respecter l'indentation automatique proposée par l'éditeur SciNotes.

1.3 Exercices

Exercice 1.1 (Une proposition)
Écrire en langage `Scilab` : m ou n sont des entiers naturels pairs.

Exercice 1.2 (Permutation de 2 variables)
Écrire en langage `Scilab` des instructions qui permettent d'échanger les valeurs contenues dans des variables `a` et `b`.

Exercice 1.3 (Équation du second degré)
Écrire un programme qui saisit trois réels a, b et c puis détermine et affiche l'ensemble des solutions de l'équation $ax^2+bx+c=0$ dans $\mathbb{R}$. Il faudra tenir compte de toutes les situations possibles (même les cas $a=0$ ou encore $a=0$ et $b=0$).

Exercice 1.4 (Théorème de Pythagore)
Écrire un programme qui saisit trois réels a, b et c puis détermine si un triangle dont les longueurs de ses trois côtés sont les réels a, b, c est un triangle rectangle ou non (attention, les trois réels ne sont pas nécessairement rangés dans l'ordre croissant).

Exercice 1.5 (Tri décroissant sur 3 réels)
Écrire un programme qui saisit trois réels a, b et c puis les affiche tous les trois dans l'ordre décroissant.

Exercice 1.6 (Quelques boucles)
Calculer et afficher :

1. x^x pour $x \in \{10, 1, 0.1, 0.01, 0.001, 0.0001\}$.
2. $n \ln\left(1 + \frac{a}{n}\right)$ pour $n \in \left\{10^k \mid k \in [\![0, 9]\!]\right\}$ et un réel $a \geqslant 0$ demandé à l'utilisateur.
3. $\mathrm{e}^x \ln\left(1 + \mathrm{e}^{-x}\right)$ pour $x \in \left\{10^k \mid k \in [\![0, 15]\!]\right\}$.
4. Déterminer le plus petit entier naturel n tel que : $\left|\left(1 + \frac{2}{n}\right)^n - \left(1 + \frac{1}{n}\right)^{2n}\right| < 10^{-k}$ pour $k \in \{3, 4, 5, 6\}$.

Exercice 1.7 (Un produit et une somme)
À l'aide d'une boucle, calculer 50! puis $1^{50} + 2^{49} + 3^{48} + \cdots + 48^3 + 49^2 + 50^1$.

Exercice 1.8 (Jeu du "c'est plus / c'est moins")
Écrire un programme qui choisit, au hasard, un nombre entier compris entre 1 et 100 (`N=1+floor(100*rand())`) puis propose au plus 6 essais à l'utilisateur de deviner le nombre choisi puis répond à l'utilisateur après chacune de ses propositions de valeur par : `gagné`, `plus grand`, `plus petit`. Dans le cas où l'utilisateur n'a pas deviné le nombre à l'issue des 6 essais, l'ordinateur affiche la phrase : `Dommage, le nombre à deviner était ...` (on affiche ici la valeur exacte à deviner).

Chapitre 2

Sommes et produits

2.1 Stratégies de calculs d'une somme

Définition 2.1 :
Soit $(u_n)_{n\in\mathbb{N}}$ une suite de réels. Pour tout $n \in \mathbb{N}$, on pose : $S_n = \sum_{k=0}^{n} u_k$. Trois stratégies de calculs peuvent s'envisager, le choix va dépendre de la méthode de définition de la suite $(u_n)_{n\in\mathbb{N}}$:

— Opérations pointées et instruction `sum` : la suite $(u_n)_{n\in\mathbb{N}}$ est définie explicitement par des opérations simples :

$$S_n = \sum_{k=0}^{n} \frac{x^k}{k+1} \quad \rightsquigarrow \quad \texttt{K=[0:n] ; S=sum((x.\^{}K)./(K+1))}$$

— Boucle `for` et instruction `sum` : la suite $(u_n)_{n\in\mathbb{N}}$ est définie par récurrence (ou explicitement) et les termes sont stockés dans un tableau (on fera attention aux indices puisque les tableaux démarrent au rang 1 alors que les suites démarrent souvent au rang 0) :

$$\begin{cases} \forall k \in \mathbb{N},\ u_{k+1} = \sqrt{1-u_k} & u_0 = \frac{1}{4} \\ \forall n \in \mathbb{N},\ S_n = \sum_{k=0}^{n} u_k \end{cases} \quad \rightsquigarrow$$

```
u=zeros(1,n+1) ; u(1)=1/4
for k=1:n u(k+1)=sqrt(1-u(k)) ; end
S=sum(u)
```

— Boucle `for` uniquement : la suite $(u_n)_{n\in\mathbb{N}}$ est définie par récurrence (ou bien s'y ramène simplement tout en limitant le nombre de calculs) et on ne stocke rien dans un tableau : :

$$\begin{cases} \forall k \in \mathbb{N},\ u_{k+1} = \sqrt{1-u_k} & u_0 = \frac{1}{4} \\ \forall n \in \mathbb{N},\ S_n = \sum_{k=0}^{n} u_k \end{cases} \quad \rightsquigarrow$$

```
u=1/4 ; S=u
for k=1:n u=sqrt(1-u) ; S=S+u ; end
```

$$S_n = \sum_{k=0}^{n} \frac{x^k}{k!} \quad \rightsquigarrow$$

```
u=1 ; S=u
for k=1:n u=u*x/k ; S=S+u ; end
```

Remarques :

— Les stratégies sont les mêmes pour un calcul de produit.

— Les opérations pointées sont optimisées en terme de temps de calcul en comparaison d'une boucle : il faut les privilégier dès que possible lorsqu'on travaille avec beaucoup de données.

— Par contre, les tableaux de taille supérieure à trois millions (environ) de cellules provoquent une erreur.

2.2 Exercices

Exercice 2.1

1. Pour tout entier $n \geqslant 2$, on pose : $S_n = \sum_{k=2}^{n} \frac{1}{k\ln(k)}$.
   (a) Donner une valeur approchée de S_n pour $n \in \left\{10^3, 10^4, 10^5, 10^6, 3\times10^6, 10^7, 10^8\right\}$ (penser à des stratégies différentes selon les valeurs de n : opérations pointées ... ou pas).
   (b) La suite $\left(\sum_{k=2}^{n} \frac{1}{k\ln(k)}\right)_{n\geqslant 2}$ est-elle convergente ?
2. Mêmes questions pour la suite $\left(\ln(n) - \sum_{k=1}^{n} \frac{1}{k}\right)_{n\geqslant 1}$.

Exercice 2.2

Soit $a \in \mathbb{R}$. On pose : $u_0 = a$ et $u_{k+1} = \frac{2u_k+1}{u_k+1}$ pour tout $k \in \mathbb{N}$. Écrire un programme qui calcule $\sum_{k=0}^{n} u_k$ pour un entier n fourni par l'utilisateur. Cette somme converge-t-elle lorsque n tend vers $+\infty$?

Exercice 2.3

Soit $p \in [0.1, 0.9]$. On pose : $q = 1-p$, $u_2 = p^2$, $u_3 = qp^2$ et $u_{k+2} = q \times u_{k+1} + pq \times u_k$ pour tout $k \in \mathbb{N}$.

1. Écrire un programme qui calcule $\sum_{k=2}^{n} k u_k$ pour un entier $n \geqslant 2$ fourni par l'utilisateur.

2. On admet que la suite $\left(\sum_{k=0}^{n} k u_k\right)_{n\geqslant 2}$ converge et on note $f(p)$ sa limite. Représenter graphiquement la fonction f sur $[0.1, 0.9]$.

Exercice 2.4
Écrire un programme qui calcule la somme $\sum_{i=1}^{n}\sum_{j=i}^{n} \frac{1}{i^j 2^i}$ pour un entier n fixé.

Exercice 2.5 (Factorielle et coefficient du binôme)
1. Écrire un programme qui calcule $n!$ pour un entier n défini au préalable.
2. Écrire un programme qui calcule $\binom{n}{k}$ pour des entiers n et k définis au préalable (penser à gérer les cas $n<0$, $k>n$ et $k<0$).

Exercice 2.6
Écrire un programme qui calcule $1^n + 2^{n-1} + 3^{n-2} + \cdots + (n-2)^3 + (n-1)^2 + n^1$ pour un entier n défini au préalable.

Exercice 2.7
Calculer : $\prod_{k=1}^{n-1} \sqrt{2}^{n-k} \sin^2\left(\frac{k\pi}{2n}\right)$ lorsque $n \in \{2, 5, 10, 20, 50, 100\}$. À-t-on une suite croissante de la variable n ?

Exercice 2.8
Soit p une entier tel que $p \geqslant 2$. Soit $(a_0, a_1, \ldots, a_{p-1}) \in \mathbb{R}^p$. Soit E_p l'ensemble des suites réelles $(u_n)_{n\in\mathbb{N}}$ telles que :

$$(u_0, \ldots, u_{p-1}) \in \mathbb{R}^p \qquad \text{et} \qquad \forall n \in \mathbb{N},\ u_{n+p} = a_{p-1}u_{n+p-1} + \cdots + a_1 u_{n+1} + a_0 u_n$$

1. *Programmation.*
   (a) Dans cette seule question, on suppose que $p = 3$ ainsi que :
   $$(a_0, a_1, a_2) = \left(\frac{1}{3}, -1, \frac{1}{2}\right) \qquad \text{et} \qquad (u_0, u_1, u_2) = (1, -1, 0)$$
   Calculer la valeur de u_n lorsque $n \in \left\{10^k \mid k \in [\![1, 6]\!]\right\}$.
   (b) Dans cette seule question, on suppose que $p = 100$ ainsi que :
   $$\forall k \in [\![0, p-1]\!], \qquad a_k = (-1)^k k \qquad \text{et} \qquad u_k = \frac{1}{p-k}$$
   Calculer la valeur de u_n lorsque $n \in \left\{10^k \mid k \in [\![1, 6]\!]\right\}$.
2. *Mathématiques.* Pour tout entier $n \geqslant 0$, on note X_n la matrice colonne de $\mathcal{M}_{p,1}(\mathbb{R})$ dont les composantes sont $(u_n, u_{n+1}, \ldots, u_{n+p-1})$.
   (a) Montrer qu'il existe une matrice M de $\mathcal{M}_p(\mathbb{R})$, indépendante de l'entier n, telle que : $\forall n \in \mathbb{N},\ X_{n+1} = MX_n$. Cette matrice M est appelée ***matrice compagnon*** de la suite $(u_n)_{n\in\mathbb{N}}$.
   (b) En déduire une expression de X_n en fonction de M, X_0 et n.
3. *Programmation.* Déduire du résultat qui précède un nouvel algorithme pour obtenir les résultats de la première question.

Chapitre 3

Structure de données : les matrices

3.1 Matrices

Définition 3.1 (Matrice) :
Une **matrice** est un tableau de nombre à double entrée (lignes/colonnes) : le marqueur de début/fin est le *crochet*, les matrices sont saisies par ligne séparées d'un *point-virgule*, les éléments d'une ligne sont séparées par une *virgule*.

Exemple :

`A=[1,20,300;4,50,600]` déclare le tableau

1	20	300
4	50	600

sous le nom `A`.

Définition 3.2 (Taille d'une matrice) :
- `length(A)` : nombre total de coefficients de la matrice A.
- `[n,p]=size(A)` : la variable n (resp. p) contient alors le nombre de lignes (resp. colonnes) de la matrice A.

Exemple :
Avec la matrice `A` définie ci-avant, `disp(length(A))` donne `6` et `disp(size(A))` donne `[2,3]`.

Définition 3.3 (Matrices pré-construites) :
- `zeros(n,p)` : matrice nulle à n lignes et p colonnes.
- `ones(n,p)` : matrice remplie de 1 sur n lignes et p colonnes.
- `eye(n,p)` : matrice unité avec des 1 en diagonale et 0 ailleurs (fonctionne aussi avec $n \neq p$).
- `rand(n,p)` : matrice aléatoire à n lignes et p colonnes, chaque coefficient est choisi au hasard dans $[0,1[$.
- `linspace(a,b,n+1)` : matrice-ligne de premier terme a, de dernier b et contenant $n+1$ termes en progression arithmétique (donc de raison $\frac{b-a}{n}$).
- `[a:r:b]` : matrice-ligne de premier terme a, en progression arithmétique de raison r et dont le dernier terme est soit exactement b soit le plus grand terme possible inférieur à b. Si le nombre r est manquant (`[a:b]`) alors la raison vaut 1 par défaut.

3.2 Opérations constructivistes sur les matrices

Définition 3.4 (Concaténation) :
- `A=[B,C]` : les colonnes de A sont les colonnes de B suivies de celles de C pourvu que B et C aient le même nombre de lignes,
- `A=[B;C]` : les lignes de A sont les lignes de B suivies de celles de C pourvu que B et C aient le même nombre de colonnes.

Définition 3.5 (Extraction de sous-matrices) :
- `A(i,:)` : ligne i de A,
- `A(:,j)` : colonne j de A,
- `A(n1:n2,p1:p2)` : sous-matrice de A constituée des lignes n_1 à n_2 et des colonnes p_1 à p_2 de A.
- `A(L,C)` : plus généralement, A étant une matrice à n lignes et p colonnes, L une matrice-ligne dont les coefficients sont des entiers de $[\![1,n]\!]$, C matrice-ligne dont les coefficients sont des entiers de $[\![1,p]\!]$, désigne la matrice comportant `length(L)` lignes et `length(C)` colonnes et pour coefficients les réels $A_{i,j}$ pour tout $i \in L$ et tout $j \in C$.

Définition 3.6 (Extraction de coefficients) :
- `A(i,j)` : coefficient de la ligne i et colonne j de A,
- `A(k)` : $k$ème coefficient de A en les comptant de haut en bas et gauche à droite et .
  Si A est une matrice à n lignes et p colonnes, si $i \in [\![1,n]\!]$ et si $j \in [\![1,p]\!]$, on : `A(i,j)=A((j-1)*n+i)`.

Définition 3.7 (Recherche de rangs) :
- `L=find(A>0)` : fournit une matrice-ligne contenant la liste des indices k tels que $A(k) > 0$.
- `L=find(A==0)` : fournit une matrice-ligne contenant la liste des indices k tels que $A(k) = 0$.
- `[L,C]=find(A>=0)` : fournit deux matrices-lignes contenant la liste L des numéros de ligne i et la liste C des numéros de colonnes j telles que $A(i,j) > 0$.

Exemples :
- `A=[1,2,3;4,5,6] ; length([1,2,3;4,5,6])`
  `B=size(A)`
- `n=7 ; C=[1:n;2:n+1;3:n+2] ; C([2,3],[1,3])`
- `D=[1:2:9;2:2:10] ; D([3,1],[1,3])`
- `E=[linspace(0,10,6);linspace(1,11,6)] ; find(E==4)`

3.3 Opérations mathématiques sur les matrices

Définition 3.8 (Opérations algébriques) :
- Transposée d'une matrice A : `A'` (pour permuter les lignes en colonnes et réciproquement).
- Opérations d'espace vectoriel : somme de deux matrices de même taille `A+B` et produit par un réel `5*A`.
- Produit mathématique de deux matrices (cf cours de maths) : `A*B`, `A^k`, `inv(A)` (inverse : $A * A^{-1} = I$).

Définition 3.9 (Opérations terme à terme) :
- Opérations « pointées » : `A.*B`, `A./B`, `4./A`, `A.^3`,... *Les opérations d'espace vectoriel sont naturellement pointées.*
- Fonctions numériques opérant terme à terme : `log(A)`, `abs(A)`, `cos(A)`, ...

Exemple :

Si `A=`
1	-2	3
-4	5	-6

alors `A.^3=`
1	-8	27
-64	125	-216

et `abs(A)=`
1	2	3
4	5	6

.

Si `A=`
1	2
2	1

alors `A^2=`
5	4
4	5

et `A.^2=`
1	4
4	1

sont des matrices différentes.

Définition 3.10 (Fonctions matricielles) :
- `sum(A)` (resp. `prod(A)`, `mean(A)`) : détermine la somme (resp. le produit, la moyenne) de tous les éléments de A.
- `max(A)` (resp. `min(A)`) : détermine le plus grand (resp. petit) de tous les éléments de A.

Remarque :
Toutes ces opérations sont applicables sur des sous-matrices comme sur une matrice `A` de taille $n \times p$:

$$\begin{aligned}
\texttt{sum(A(1,:))} &\rightarrow \sum_{j=1}^{p} a_{1,j} \\
\texttt{prod(A([1,3],:))} &\rightarrow \left(\prod_{j=1}^{p} a_{1,j}\right) \times \left(\prod_{j=1}^{p} a_{3,j}\right) \\
\texttt{max(A([1:3],[2,4]))} &\rightarrow \max\{a_{1,2}, a_{1,4}, a_{2,2}, a_{2,4}, a_{3,2}, a_{3,4}\}
\end{aligned}$$

3.4 Exercices

Exercice 3.1 (Constructions de matrices)

1. Construire la matrice à n lignes et n colonnes (n fourni par l'utilisateur) : $\begin{pmatrix} 1 & 1 & \dots & 1 & 1 \\ 1 & 0 & \dots & 0 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & 0 & \dots & 0 & 1 \\ 1 & 1 & \dots & 1 & 1 \end{pmatrix}$.
2. (a) Construire la matrice $A = \begin{pmatrix} 1 & 2 & \dots & n & n+1 & \dots & 2n-1 & 2n \\ 2 & 4 & \dots & 2n & 2n+2 & \dots & 4n-2 & 4n \\ 3 & 6 & \dots & 3n & 3n+3 & \dots & 6n-3 & 6n \end{pmatrix}$.

   (b) Transformer la matrice A en la matrice $B = \begin{pmatrix} n+1 & \dots & 2n-1 & 2n & 1 & 2 & \dots & n \\ 2n+2 & \dots & 4n-2 & 4n & 2 & 4 & \dots & 2n \\ 3n+3 & \dots & 6n-3 & 6n & 3 & 6 & \dots & 3n \end{pmatrix}$.

Exercice 3.2 (Autres constructions)
Soit r un réel non nul, a un réel quelconque et m un entier naturel non nul. On considère la suite arithmétique $(u_n)_{n\in\mathbb{N}}$ de raison r (non nulle) et de terme de rang m égal au réel a. Soit A la matrice à trois lignes telle que sa première ligne comporte les termes de rang $0, 1, 2, \ldots, 2m$ de la suite (u_n), la deuxième ligne comporte les mêmes termes dans l'ordre contraire, la troisième ligne est égale à la somme des deux premières. Construire la matrice A de deux façons différentes (avec et sans `linspace`).

Exercice 3.3 (Autres constructions)
Soient a, b deux réels tels que $a < b$ et n un entier naturel non nul. Construire une liste de réels stockés dans une matrice ligne `x` découpant l'intervalle $[a, b]$ en n intervalles de même largeur (avec et sans `linspace`).

Exercice 3.4
Avec des opérations matricielles, calculer pour $n = 50$:

$$n! \qquad \text{et} \qquad 1^n + 2^{n-1} + 3^{n-2} + \cdots + (n-2)^3 + (n-1)^2 + n^1$$

Exercice 3.5 (Des limites)

1. On pose : $\forall n \in \mathbb{N}^*,\ u_n = \dfrac{n!}{\left(\frac{n}{\mathrm{e}}\right)^n \sqrt{2\pi n}}$.
   (a) Construire une matrice ligne `U` contenant les valeurs u_n pour $n \in \left\{10^k \mid k \in [\![0, 6]\!]\right\}$.
   (b) Estimer la valeur de la limite : $\lim\limits_{n\to+\infty} u_n$. Donner une valeur approchée de $n!$ lorsque n est grand.
2. On pose : $\forall (k, n) \in (\mathbb{N}^*)^2,\ u_{k,n} = \dfrac{n^{k+1}}{1^k + 2^k + 3^k + \cdots + n^k}$.
   (a) Construire une matrice U dont l'indice de ligne est k et l'indice de colonne est n contenant les valeurs de $u_{k,n}$ pour $k \in [\![1, 10]\!]$ et $n \in \left\{10^i \mid i \in [\![0, 6]\!]\right\}$.
   (b) L'entier k étant fixé, estimer la valeur de la limite : $\lim\limits_{n\to+\infty} u_{k,n}$. Donner une valeur approchée de $1^k + 2^k + 3^k + \cdots + n^k$ lorsque n est grand.

Exercice 3.6 (Différentes moyennes)
Calculer les moyennes arithmétique, géométrique, harmonique et quadratique des entiers de 1 à n :

$$A = \frac{1}{n}\sum_{k=1}^{n} k \qquad G = \exp\left(\frac{1}{n}\sum_{k=1}^{n} \ln(k)\right) \qquad H = \frac{1}{\frac{1}{n}\sum\limits_{k=1}^{n} \frac{1}{k}} \qquad Q = \sqrt{\frac{1}{n}\sum_{k=1}^{n} k^2}$$

L'ordre des ces quatre nombres dépend-il de n ?

Exercice 3.7

1. Soit $n \in \mathbb{N}$ tel que $n \geqslant 2$. Construire la matrice A de taille $n \times n$ suivante : $A = \begin{pmatrix} \frac{1}{1} & \frac{1}{2} & \cdots & \frac{1}{n-1} & \frac{1}{n} \\ \frac{2}{1} & \frac{2}{2} & \cdots & \frac{2}{n-1} & \frac{2}{n} \\ \vdots & \vdots & & \vdots & \vdots \\ \frac{n-1}{1} & \frac{n-1}{2} & \cdots & \frac{n-1}{n-1} & \frac{n-1}{n} \\ \frac{n}{1} & \frac{n}{2} & \cdots & \frac{n}{n-1} & \frac{n}{n} \end{pmatrix}$.
2. Vérifier que le produit des coefficients de A vaut 1.
3. Calculer la somme S_n des coefficients de l'antidiagonale de A. Proposer une valeur pour $\lim\limits_{n\to+\infty} \dfrac{S_n}{n\ln(n)}$.

Exercice 3.8 (Tables d'opérations)

1. Afficher la table de multiplication des entiers de 0 à 10.
2. Même question mais on affiche, à chaque calcul, le reste de la multiplication dans la division par 6.

Exercice 3.9 (Colonne de somme maximale)

1. La matrice A étant de taille 3×3 et ses coefficients choisis au hasard parmi les réels de $[0, 1[$, afficher le numéro de la colonne de A dont la somme est la plus grande.
2. Même question avec une matrice de taille $n \times n$ où $n \in \mathbb{N}^*$ est choisi par l'utilisateur.

Exercice 3.10 (Matrices stochastiques en ligne)

1. Construire une matrice A de taille 3×3 dont les coefficient sont choisis au hasard parmi les réels de $[0, 1[$ de sorte que la somme de chaque ligne soit égale à 1.

2. Même question avec une matrice de taille $n \times n$ où $n \in \mathbb{N}^*$ est choisi par l'utilisateur.

Exercice 3.11 (Minimax)

1. La matrice A étant de taille 3×3 et ses coefficients choisis au hasard parmi les réels de $[0, 1[$:
   (a) construire la matrice ligne B dont le coefficient numéro j est le plus grand nombre de la colonne j de A puis stocker dans la variable `minimaxC` le plus petit nombre de la matrice B,
   (b) construire la matrice colonne C dont le coefficient numéro i est le plus grand nombre de la ligne i de A puis stocker dans la variable `minimaxL` le plus petit nombre de la matrice C,
   (c) comparer ces deux nombres.
2. Même question avec une matrice de taille $n \times n$ où $n \in \mathbb{N}^*$ est choisi par l'utilisateur.

Exercice 3.12 (Fréquences)

1. Écrire un script choisissant une matrice aléatoire A à 1000 lignes et 1000 colonnes puis construisant une matrice-ligne F dont les termes sont les fréquences d'apparition des termes de A inférieurs respectivement à $0.1, 0.2, \ldots, 0.9$.
2. On élève chaque coefficient de A au carré, la matrice finale s'appelant toujours A.
   (a) Reprendre la question qui précède.
   (b) [*] Quelle est, approximativement, la fréquence des coefficients de A dont le premier chiffre non nul est 1 ?
   (c) [*] Construire la matrice ligne G dont le coefficient k est, approximativement, la fréquence des coefficients de A dont le premier chiffre non nul est k ?

Chapitre 4

Tracé de courbes de fonctions dans le plan

4.1 Tracé à partir d'une liste de points

Définition 4.1 :
Soit `x` et `y` deux matrices-lignes (ou colonnes) de même taille $(1,n)$ (ou $(n,1)$). Soit $A_1,\dots,A_n$ les points de coordonnées respectives $(x(1),y(1)),\dots,(x(n),y(n))$. L'instruction `plot2d(x,y)` trace les segments $[A_1,A_2],\dots,[A_{n-1},A_n]$.

Exemple :
Tester successivement dans la console :

- `x=[-1:1:2] ; y=x.^2 ; plot2d(x,y)`
- `x=[-1:0.5:2] ; y=x.^2 ; plot2d(x,y)`
- `x=[-1:0.1:2] ; y=x.^2 ; plot2d(x,y)`
- `x=[-1:0.01:2] ; y=x.^2 ; plot2d(x,y)`

Remarques :

- Par défaut, le logiciel adapte automatiquement la fenêtre de visualisation selon les coordonnées des différents points, les graduations sont placées en bas du graphique pour les abscisses et à gauche pour les ordonnées.
- Si on tourne la molette de la souris, on obtient un agrandissement/réduction.
- On peut ajouter des options à cette instruction pour modifier l'aspect du graphique mais le plus simple est de modifier ces paramètres dans la fenêtre de la figure (Édition / Propriétés de la figure ou Propriétés des axes).
- Le tracé d'une nouvelle courbe n'efface pas le tracé de l'ancienne si la fenêtre graphique n'est pas fermée entre les deux tracés : c'est pratique si on a un tracé non continu, on sépare en plusieurs composantes.

Définition 4.2 :
Pour effectuer plusieurs tracés côte à côte (en lignes ou en colonnes), on utilise l'instruction `subplot(n,p,k)` pour indiquer, avant d'effectuer le tracé, sa position dans la fenêtre graphique : le prochain graphique sera positionné dans la fenêtre courante devant contenir $n\times p$ graphiques répartis sur n lignes et p colonnes en position (i,j) telle que $k=p(i-1)+j$.

Exemple :

```
x=[0:0.01:1] ; y=2*x-1 ; z=x.^2 ; subplot(1,2,1) ; plot2d(x,y1) ; subplot(1,2,2) ; plot2d(x,y2)
```

4.2 À partir d'une fonction

Définition 4.3 :
Soit `x` une matrice-ligne (ou matrice-colonne) et `f` une fonction définie à l'aide de l'instruction `function`. Pour tracer la courbe de la fonction `f` sur la plage d'abscisses définies par la matrice `x`, deux méthodes sont possibles :

- ou bien on utilise l'instruction `fplot2d(x,f)` (qui possède les mêmes options que `plot2d`) et effectue le tracé directement,
- ou bien on utilise de manière combinée les deux instructions `y=feval(x,f)`, qui produit une matrice de même taille que `x` contenant les images par `f` des éléments de `x`, suivie de `plot2d(x,y)`.

4.3 Exercices

Exercice 4.1
Tracer successivement la courbe représentative des fonctions :

$$f\colon x\mapsto\begin{cases}-2 & \text{si } x<-1\\ 2x & \text{si } x\in[-1,1]\\ 2 & \text{si } x>1\end{cases}\quad\text{sur } [-3,3]\qquad g\colon x\mapsto\frac{1}{x}\quad\text{sur } [-2,0[\cup]0,2]\qquad h\colon x\mapsto\frac{\ln(1+\mathrm{e}^{-x})}{\mathrm{e}^{-x}}\quad\text{sur } [-3,3]$$

Exercice 4.2
Tracer successivement la courbe représentative des fonctions qui suivent sur des intervalles adaptés. Admettent-elle un prolongement par continuité ?

$$f\colon x \mapsto x^{-\frac{1}{1+\ln(x)}} \qquad g\colon x \mapsto \frac{1}{x}\ln\left(\frac{1+x}{1-x}\right)$$

Exercice 4.3
Tracer, sur un même graphique, la courbe représentative des fonctions qui suivent sur l'intervalle $[0, 6\pi]$:

$$f\colon x \mapsto x\sin(x) \qquad g\colon x \mapsto \max\{f(t) \mid t \in [0,x]\}$$

Exercice 4.4
Tracer, sur un même graphique, les courbes des fonctions :

$$x \mapsto -\mathrm{e}^{-x} \qquad x \mapsto \mathrm{e}^{-x} \qquad x \mapsto \sin(x)\mathrm{e}^{-x}$$

sur l'intervalle $[0, 6\pi]$, les deux premières courbes étant en bleu, la troisième en rouge.

Exercice 4.5
Pour chaque fonction, tracer, dans une même fenêtre et les unes dessous les autres, les courbes représentatives des fonctions f, f' et f'' :

$$f_1\colon x \mapsto x^2\mathrm{e}^{-x} \qquad f_2(x) = \frac{1}{\sqrt{1-x^2}} \qquad f_3(x) = \tan(x) \qquad f_4(x) = x\ln(x)$$

On rappelle que : $f'(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$ mais on pourra utiliser, à part sur les bord de l'intervalle de représentation, l'égalité $f'(x) = \lim_{h\to 0}\frac{f(x+h)-f(x-h)}{2h}$ sous la forme de l'approximation suivante : $f'(x) \approx \frac{f(x+h)-f(x-h)}{2h}$ pour un réel $h > 0$ « assez petit ».

Exercice 4.6
Sur un seul et même graphique, représenter sur l'intervalle $[-10, 10]$, les courbes des fonctions :

$$f_{\sigma,m}\colon x \mapsto \frac{1}{\sigma\sqrt{2\pi}}\mathrm{e}^{-\frac{(x-m)^2}{2\sigma^2}} \qquad \text{avec} \qquad \sigma \in \left\{\frac{1}{2}, 1, 2\right\} \text{ et } m \in \{-3, 0, 3\}$$

Exercice 4.7
Pour chaque entier $n \in \{1, 2, 5, 10, 20, 50\}$, tracer la courbe de la fonction exponentielle ainsi que celle de la fonction $f_n\colon x \mapsto \sum_{k=0}^{n}\frac{x^k}{k!}$. Tous ces graphiques devront être dans la même fenêtre et répartis sur deux lignes.

Exercice 4.8
Tracer la fonction de répartition des lois de probabilité : $\mathcal{B}\left(4, \frac{1}{2}\right)$, $\mathcal{B}\left(4, \frac{1}{3}\right)$, $\mathcal{U}(\llbracket 1, 4\rrbracket)$ et $\mathcal{U}(\llbracket -2, 2\rrbracket)$.

Exercice 4.9
Représenter sur un même graphique les fonctions $f_n\colon x \mapsto \begin{cases} 0 & \text{si } x < 0 \\ x^n & \text{si } 0 \leqslant x \leqslant 1 \\ 1 & \text{si } x > 1 \end{cases}$ sur l'intervalle $[-1, 2]$ pour $n \in \{1, 2, 5, 10, 20, 50\}$. Pour tout réel x, on pose : $f(x) = \lim_{n\to+\infty} f_n(x)$. Représenter f sur le graphique précédent.

Exercice 4.10
Représenter sur des graphiques différents (sur 2 lignes dans une même fenêtre) les fonction $f_a\colon x \mapsto \begin{cases} 0 & \text{si } x \leqslant 0 \\ x^{a-1}\mathrm{e}^{-x} & \text{si } x > 0 \end{cases}$ sur l'intervalle $[-1, 5]$ pour $a \in \left\{\frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2}, 3\right\}$.

Chapitre 5

Notion de fonction informatique

5.1 Définir une nouvelle fonction

Définition 5.1 :

— Toute nouvelle fonction doit être déclarée avant de démarrer la partie principale d'un programme (mais on peut la placer après la définition de différents paramètres).

— La déclaration s'effectue de la manière suivante :

```
function y=f(t) // le nom de la fonction est « f » ici
    ... // éventuellement des calculs intermédiaires avec for/while/if
    y=... // il est nécessaire d'affecter la variable locale y
endfunction
```

— On utilise par la suite cette fonction de manière très classique : `disp(f(3))` ou encore : `x=4 ; u=f(x)`.

Remarques :

— La fonction n'est pas lue ni utilisée tant que le programme n'y fait pas appel.

— Les variables `y` et `t` sont dites **locales** (à la fonction), elles n'existent et n'ont une valeur que durant l'utilisation de la fonction `f` et disparaissent ensuite.

— L'instruction `z=f(2)` (dans la partie principale du programme) envoie la valeur 2 dans la variable locale `t` (cette variable est créée à ce moment là, de même que la variable locale `y`), la fonction s'exécute donc avec cette valeur de `t` et un résultat est stocké dans la variable locale `y`, lorsque l'exécution arrive à `endfunction`, la valeur stockée dans `y` est transférée au programme et stocké dans la variable **globale** `z` (une variable globale est utilisable dans tout le programme, y compris à l'intérieur de fonctions).

— L'instruction `t=2 ; y=f(t)` fonctionne de manière identique à la différence que les variables `t` et `y` ne sont pas les mêmes que celles de l'instruction `function y=f(t)` (d'ailleurs, elles ne sont pas écrites avec la même couleur dans l'éditeur).

— Il est possible de mettre plusieurs variables en paramètre à la fonction.

5.2 Évaluation d'une fonction à un paramètre sur une liste de valeurs

Définition 5.2 :

Soit `x` un tableau de réels (matrices à n lignes et p colonnes). Pour évaluer une fonction `f` sur chacune des valeurs du tableau `x`, deux méthodes sont possibles :

— Si la définition de la fonction `f` n'est pas « trop sophistiquée », on peut écrire : `y=f(x)`.

— Dans tous les cas, on peut utiliser l'instruction `feval` (évaluation de fonction) : `y=feval(x,f)`.

Remarques :

— Dans les deux cas, le tableau `y` a le même format que le tableau `x` :

y(1,1)	...	y(1,p)	=	f(x(1,1))	...	f(x(1,p))
⋮		⋮		⋮		⋮
y(n,1)	...	y(n,p)		f(x(n,1))	...	f(x(n,p))

— L'utilisation de `feval` est préférable pour éviter les mauvaises surprises.

— Dans les deux cas, cela correspond à l'écriture avec des boucles (et initialisation de `y` auparavant) :

```
y=zeros(x)
for i=1:n
    for j=1:p
        y(i,j)=f(x(i,j))
    end
end
```

5.3 Exercices

Exercice 5.1
On donne le code `Scilab` suivant :

```
n=3 ; m=4
function z=f(x,y)
    t=x/n+y/m ; u=x^m+y^n ; z=x*log(abs(u))-y*exp(t)
endfunction
z=f(1,2)
```

1. Peut-on afficher les valeurs de `t` et `u` à l'issue de ce code ? Si oui, quelles sont leur valeur respective ?
2. Préciser les variables qui sont locales (à la fonction) et celles qui sont globales (au programme).

Exercice 5.2
On donne le code `Scilab` suivant :

```
x=3 ; y=-2
function z=f(x,y)
    z=x*log((y+1)^2)
endfunction
z=f(4,%e-1)
```

1. Préciser les variables qui sont locales (à la fonction) et celles qui sont globales (au programme).
2. Quelles sont les valeurs respectives des variables `x`, `y` et `z` à l'issue de ce script ?

Exercice 5.3
Définir la fonction $f\colon x \mapsto \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{x^2}{2}}$. Calculer $f(\ln(8))$.

Exercice 5.4
Définir la fonction $h\colon t \mapsto \begin{cases} \mathrm{e}^{-t} & \text{si } t \leqslant 0 \\ 1-t & \text{si } 0 < t \leqslant 1 \\ \ln(t) & \text{si } t > 1 \end{cases}$ et afficher $h(t)$ pour 101 valeurs équiréparties du réel $t \in [-2,3]$.

Exercice 5.5
Écrire un script qui comporte une fonction et qui, après avoir saisi un réel x_0, calcule et affiche les réels

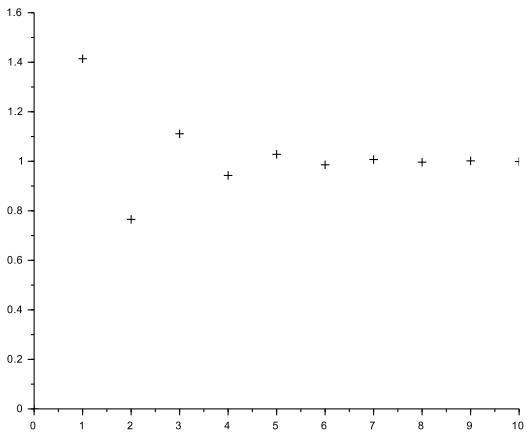
$$f(x_0),\ f(x_0+0.1),\ f(x_0+0.2),\ \ldots, f(x_0+0.9),\ f(x_0+1)$$

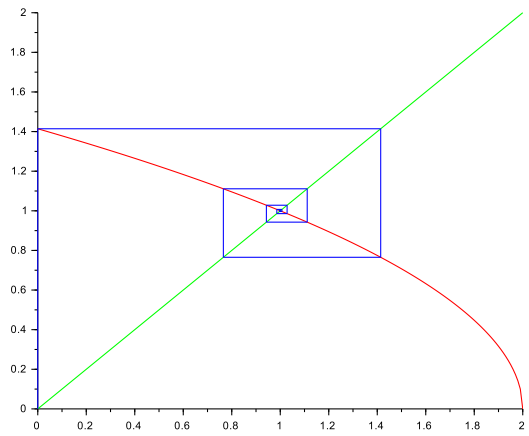
pour, successivement, les fonctions : $f\colon \mathbb{R}\to\mathbb{R},\ x \mapsto \sqrt{\frac{x+\mathrm{e}^x}{2-\cos\left(\frac{1}{x}\right)}}$ puis $f\colon \mathbb{R}\to\mathbb{R},\ x \mapsto \ln\left(\frac{|x+\lfloor x\rfloor|}{x+1-\lfloor x\rfloor}\right)$.

Exercice 5.6
Définir la fonction $g\colon t \mapsto \frac{\sin(t)+\cos(t)}{2\sin(t)\sin(3t)}$. Calculer $g\left(\frac{\pi}{8}\right)$. Pour quelle(s) valeur(s) de x la fonction g est-elle minimale puis maximale sur $\left[\frac{5\pi}{12}, \frac{7\pi}{12}\right]$? On donnera les résultats au 10^{-4} près.

Exercice 5.7
1. Représenter la fonction $f\colon x \mapsto x\sin(x)$ sur $[0, 6\pi]$: on définira les tableaux `X` $= \left\{\frac{k}{10000}6\pi \mid k \in [\![0, 10000]\!]\right\}$ (pour les abscisses) et `F` (pour les images).
2. À l'aide du tableau `F`, déterminer et représenter la fonction $h\colon x \mapsto \max_{[0,x]} f$ sur ces mêmes réels (tableau `H`). Que peut-on dire de la fonction h en observant les éléments de `H` ?

Exercice 5.8 (Suite récurrente)
On pose : $u_0 = 0$ et $u_{n+1} = \sqrt{2-u_n}$ pour tout $n \in \mathbb{N}$. Écrire un script permettant d'obtenir le graphique ci-dessous :





Chapitre 6

Simulations de variables discrètes finies

6.1 Simuler « le hasard »

Définition 6.1 :

— `rand()` est une fonction : elle fournit un résultat choisit « au hasard » dans l'intervalle $[0,1[$. Ce résultat doit être stocké ou affiché ou utilisé d'une façon quelconque.

— Si `m` et `n` sont deux entiers, `rand(m,n)` fournit une matrice à m lignes et n colonnes dont chacun élément est choisi « au hasard » dans $[0,1[$.

— Si `x` est une matrice, `rand(x)` fournit une matrice ayant le même nombre de lignes et le même nombre de colonnes que `x` et dont chaque élément est choisi de manière équiprobable (donc « au hasard ») dans $[0,1[$.

— L'instruction `rand("seed",getdate("s"))` permet « d'initialiser » le générateur de nombres aléatoires.

Exemple :

Une urne contient n boules numérotées de 1 à n. On tire, au hasard, successivement et avec remise deux boules et on note X l'écart absolu entre les deux numéros obtenus. Simuler la variable aléatoire X :

```
n=10 ; T=1+floor(n*rand(1,2)) ; X=T(1)-T(2)
```

Théorème 6.1 (Simulation de variables aléatoire suivant les lois discrètes finies usuelles) :

— Loi uniforme $\mathcal{U}(\llbracket a,b \rrbracket)$: `X=a+floor((b-a+1)*rand())`

— Loi binomiale $\mathcal{B}(n,p)$: `X=0 ; for k=1:n ; if rand()<p ; X=X+1 ; end ; end`

6.2 Représenter une approximation de la loi de probabilité

Définition 6.2 (Représentations graphiques de plusieurs répétitions) :

— L'instruction `bar(x,y)` représente le diagramme en rectangles des éléments de la matrice-ligne `y` (en ordonnée) en fonction des éléments de la matrice-ligne `x` (en abscisse). Les matrices `x` et `y` doivent avoir le même nombre de colonnes. Comme pour les représentations graphiques de fonctions, il est possible de représenter côte à côte plusieurs rectangles pour une même abscisse : pour ce faire, `x` doit être une matrice-colonne et `y` une matrice ayant le même nombre de lignes que `x`.

— L'instruction `histplot(x,y)` représente l'histogramme des fréquences d'apparition des éléments de la matrice `y` dans les intervalles définis par les éléments de la matrice x (rangés dans l'*ordre strictement croissant*) :

$$[x_1,x_2]\,,\ ]x_2,x_3]\,,\ \ldots,\ ]x_{n-1},x_n]$$

Pour plus de lisibilité du graphique, il est parfois de choisir pour valeurs du tableau `y` les centres des intervalles.

Exemple :

Simuler $N=10^5$ fois l'expérience qui précède puis représenter l'histogramme des fréquences observées :

```
n=10 ; N=10^5 ; T=1+floor(n*rand(N,2)) ; X=T(:,1)-T(:,2) ; histplot([-0.5:n-0.5],X)
```

6.3 Déterminer la loi et représenter la fonction de répartition

Définition 6.3 :

— Soit `X` une matrice-ligne. L'instruction `M=tabul(X,"i")` donne une matrice à 2 colonnes : la colonne 1 contient la liste de toutes les valeurs deux à deux distinctes contenues dans le tableau `X` rangées dans l'ordre croissant (`"i"` pour *increasing*, pour l'ordre décroissant c'est `"d"` pour *decreasing*) ; la colonne 2 contient le nombre d'apparitions, dans le tableau `X`,de la valeur correspondante dans la première ligne de `M` (instruction utilisée par `histplot`).
Exemple : `X=[1,2,2,2,1,1,2,1,1] ; M=tabul(X,"i")` donne `M=[1,5;2,4]`.

— Soit `X` une matrice-ligne. L'instruction `S=cumsum(X)` donne une matrice de même format que `X` de sorte que le terme de rang i de `S` est la somme des termes de rang 1 à i de `X`.
Exemple : `S=cumsum([1,2,3,4,5])` donne `S=[1,3,6,10,15]`.

Théorème 6.2 (Fonction de répartition) :
Un tableau `X` contient N réalisations d'une variable aléatoire X.

— On obtient les valeurs du support $X(\Omega)$ de X dans la colonne `x` et les valeurs remarquables de la fonction de répartition F_X (fréquences observées) dans la colonne `f` qui suivent :
```
XF=tabul(X,"i") ; x=XF(:,1) ; f=cumsum(XF(:,2))/N
```

— Une représentation graphique de la fonction de répartition est alors donnée par la suite d'instructions :
```
n=length(x) ; plot2d([x(1)-1,x(1)],[0,0]) ; plot2d([x(n),x(n)+1],[1,1])
for i=1:n-1 ; plot2d([x(i),x(i+1)],[f(i),f(i)]) ; end
```

6.4 Estimer l'espérance et la variance

Théorème 6.3 (Estimation de l'espérance et de la variance) :
Quand on possède N réalisations d'une variable aléatoire X stockée dans un tableau X, on obtient une estimation de l'espérance et de la variance de X (d'autant meilleure que N est grand) par :
```
E=mean(X) ; E2=mean(X.^2) ; V=E2-E^2 // ou bien E=sum(X)/N ; E2=sum(X.^2)/N
```
On peut aussi utiliser une boucle mais pas de tableau.

6.5 Exercices

Pour chaque exercice, à partir du protocole expérimental décrit et de la variable aléatoire X indiquée :

1. écrire une suite d'instructions réalisant la variable aléatoire X (de préférence sous la forme d'une fonction),
2. tracer un graphique des fréquences observées des valeurs prises par X,
3. calculer et afficher la loi de probabilité de la variable aléatoire X, représenter la fonction de répartition de X,
4. calculer une estimation de $\mathbb{E}(X)$ et de $\mathbb{V}(X)$.

Exercice 6.1
On lance un dé non truqué à 6 faces numérotées de 1 à 6. Lorsque le dé donne le résultat $k \in [\![1, 6]\!]$, on lance alors une pièce non truquée k fois et on appelle X le nombre de côtés *pile* obtenus.

Exercice 6.2
On considère une succession de n lancers d'une pièce donnant *pile* ou *face* de manière équiprobable. On note X le nombre de fois que l'on obtient 2 tirages consécutifs identiques.

Exercice 6.3
On considère une succession de n lancers d'une pièce donnant *pile* ou *face* de sorte le côté *pile* apparait avec la probabilité p. On note X le rang pour lequel on obtient pour la première fois *pile* après deux *face* consécutifs ou bien deux *face* après *pile* (X prend la valeur 0 si aucune de ces deux situations ne se produit au cours des n tirages).

Exercice 6.4
Une urne contient $n-1$ boules numérotées de 2 à n ($n \geqslant 3$). On effectue y un premier tirage : on retire toutes les boules portant un numéro strictement supérieur à ce premier numéro, on replace la boule tirée dans l'urne et on ajoute une boule portant le numéro 1. On effectue un second tirage et on note X le numéro de cette seconde boule tirée.

Exercice 6.5
Une urne contient r boules rouges et b boules bleues. On tire successivement et sans remise n boules dans cette urne ($n \in [\![1, r+b]\!]$). On note X le nombre de rouges obtenues.

Exercice 6.6
On tire n fois consécutivement et avec remise entre les tirages l'une des boules numérotées de 1 à s d'une urne. On note X le nombre de faces qui ne sont jamais apparues au cours de ces n lancers. On étudiera aussi ce qui se passe lorsque n tend vers $+\infty$.

Exercice 6.7
Une urne A contient initialement 100 boules blanches. Une urne B contient initialement 100 boules noires. On effectue n tirages (avec $n \leqslant 100$) de la façon suivante : à chaque tirage, on choisit au hasard l'une des urnes, on y pioche au hasard une boule, on place la boule tirée dans l'autre urne. On note X le nombre de boules blanches dans l'urne A à l'issue des n tirages. On étudiera aussi ce qui se passe lorsque n tend vers $+\infty$.

Exercice 6.8
Une urne contient initialement n boules numérotées de 1 à n. On effectue un certain nombre de tirages dans l'urne de la façon suivante :

— si la boule tirée porte le numéro 1, on s'arrête,

— si la boule tirée porte un numéro $k > 1$ alors on ne remet pas cette boule dans l'urne et on retire de l'urne toutes les boules portant un numéro supérieur à k. On note X le nombre de tirages effectués lorsque les tirages cessent.

Chapitre 7

Suites récurrentes

7.1 Une suite récurrente

Exercice 7.1

Soit la suite définie par : $u_0 = 2$ et $\forall n \in \mathbb{N},\ u_{n+1} = \frac{3u_n - 1}{u_n + 1}$.

1. À l'aide d'un tableau, écrire une fonction d'en-tête `function x=u(n)` permettant de calculer u_n pour un entier n fourni en paramètre.
2. Sans utiliser de tableau, écrire une fonction d'en-tête `function x=v(n)` permettant de calculer u_n pour un entier n fourni en paramètre.

Exercice 7.2

Soit la suite définie par : $u_0 = 2$ et $\forall n \in \mathbb{N},\ u_{n+1} = (n+1)u_n^2 - n(u_n + 2)$.

1. À l'aide d'un tableau, écrire une fonction d'en-tête `function x=u(n)` permettant de calculer u_n pour un entier n fourni en paramètre.
2. Sans utiliser de tableau, écrire une fonction d'en-tête `function x=v(n)` permettant de calculer u_n pour un entier n fourni en paramètre.

Exercice 7.3 (Suite de Syracuse)

On considère la suite (u_n) définie par :

$$u_0 \in \mathbb{N}^* \qquad \text{et} \qquad \forall n \in \mathbb{N},\ u_{n+1} = \begin{cases} 3u_n + 1 & \text{si } u_n \text{ est impair} \\ \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \end{cases}$$

1. Écrire un programme qui saisit le réel u_0 et l'entier n puis calcule et affiche le réel u_n.
2. Modifier le programme pour qu'il affiche ou bien le plus petit entier $k \in [\![1, n]\!]$ tel que $u_k = 1$ s'il existe, ou bien la mention « pas de terme égal à 1 ! ».

Remarque : Une conjecture (non démontrée à ce jour) affirme que : $\forall u_0 \in \mathbb{N}^*,\ \exists k \in \mathbb{N}^*\ /\ u_k = 1$ (et alors on a : $u_{k+1} = 4$, $u_{k+2} = 2$, $u_{k+3} = 1$, $u_{k+4} = 4$, ...)

7.2 Suites récurrentes d'ordre 2

Exercice 7.4

Soit $(a, b) \in \mathbb{R}^2$. On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par : $u_0 = a$, $u_1 = b$ et $u_{n+2} = u_{n+1} + u_n$ pour tout $n \in \mathbb{N}$.

1. À l'aide d'un tableau, écrire une fonction d'en-tête `function x=u(n)` permettant de calculer u_n pour un entier n fourni en paramètre.
2. Sans utiliser de tableau, écrire une fonction d'en-tête `function x=v(n)` permettant de calculer u_n pour un entier n fourni en paramètre.

7.3 Suites adjacentes

Exercice 7.5

Soit la suite définie par : $u_0 \in]0, 1[$ et $\forall n \in \mathbb{N},\ u_{n+1} = \sqrt{1 - u_n}$.

1. *Mathématiques.* Montrer que les suites $(u_{2n})_{n\in\mathbb{N}}$et $(u_{2n+1})_{n\in\mathbb{N}}$ sont adjacentes. En déduire que la suite $(u_n)_{n\in\mathbb{N}}$ converge.
2. *Informatique.* Soit $\varepsilon > 0$ et $u_0 \in]0, 1[$ donnés. Écrire un script qui calcule et affiche une valeur approchée de la limite ℓ de cette suite à ε près.

7.4 Suites récurrentes imbriquées

Exercice 7.6 (Moyenne arithmético-géométrique)

Soit $(a,b) \in [0,+\infty[^2$. On définit les suites $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ par : $\begin{cases} u_0 &= a \\ v_0 &= b \end{cases}$ et $\begin{cases} u_{k+1} &= \frac{u_k+v_k}{2} \\ v_{k+1} &= \sqrt{u_k v_k} \end{cases}$ pour tout $k \in \mathbb{N}$.

1. *Mathématiques.* Vérifier que : $\forall k \in \mathbb{N},\ u_{k+1} - v_{k+1} = \frac{1}{2}\left(\sqrt{u_k} - \sqrt{v_k}\right)^2$. En déduire que les deux suites ainsi définies sont adjacentes.
2. *Informatique.*
   (a) Écrire une fonction d'en-tête `U=moy_ari_geo(a,b,n)` recevant en paramètres les réels a et b et l'entier n et retournant la matrice $U_n = \begin{pmatrix} u_n & v_n \end{pmatrix} \in \mathcal{M}_{1,2}(\mathbb{R})$.
   (b) Soit $\varepsilon = 10^{-k}$ avec $k \in \mathbb{N}^*$. Pour tout réel $a \in [0,+\infty[$, on pose : $f(a) = \ell(a,1)$. Représenter la fonction f sur l'intervalle $[0,5]$ en utilisant pour valeur de $f(a)$ le premier terme u_n tel que $|u_n - v_n| \leqslant \varepsilon$.

7.5 Exercices

Exercice 7.7

1. Soit $a \in \mathbb{R}$. Calculer u_n lorsque : $u_0 = a$ et $u_{k+1} = (1-a)u_k + a$ pour tout $k \in \mathbb{N}$.
2. Calculer u_{10000} pour tout $a \in \left\{ \frac{i}{10} \mid i \in [\![-10,30]\!] \right\}$. On donnera le résultat sous la forme d'une matrice colonne.

Exercice 7.8

Soit $a \in \mathbb{R}$. On pose : $u_0 = a$ et $u_{k+1} = \frac{2u_k+1}{u_k+1}$ pour tout $k \in \mathbb{N}$. Calculer u_n pour un entier n fourni par l'utilisateur. La suite (u_n) converge-t-elle ?

Exercice 7.9 (Convergence d'une suite)

Soit a un réel strictement positif. On définit une suite (u_n) par : $u_0 > 0$ et, pour tout $\forall n \in \mathbb{N},\ u_{n+1} = \frac{1}{2}\left(u_n + \frac{a}{u_n}\right)$.

1. Vérifier que la seule limite possible est $\sqrt{a}$.
2. Pour $a = 999999$ et $u_0 = a$, déterminer le plus petit entier n tel que $|u_n - a| \leqslant 10^{-15}$.

Exercice 7.10

Soit A la matrice carrée de taille n dont la diagonale ne comporte que des 2 et dont tous les autres coefficients sont égaux à 1. On pose : $B_0 = A$ et $B_{k+1} = \frac{1}{2}(B_k + B_k^{-1}A)$ pour tout entier k. Calculer B_{10} et B_{10}^2. Que constate-t-on ?

Chapitre 8

Calcul approché d'intégrales

8.1 Méthodes des rectangles

Théorème 8.1 (Sommes de Riemann, rappel) :
Soit f une fonction continue sur un segment $[a,b]$. On sait alors :

$$\int_a^b f(x)\mathrm{d}x = \lim_{n\to+\infty} \frac{b-a}{n}\sum_{k=1}^{n} f\left(a+k\frac{b-a}{n}\right)$$

Compléter le programme suivant afin qu'il calcule une valeur approchée de $\int_a^b \ln(1+x)\mathrm{d}x$ lorsque l'utilisateur fournit les données $a=0$, $b=1$, et successivement $n\in\{10,100,1000\}$.

```
function y=f(x)
    y=log(1+x)
endfunction
disp("Méthode des rectangles pour le calcul d'"une intégrale")
a=input("Borne inférieure de l'"intégrale : ")
b=input("Borne supérieure de l'"intégrale : ")
n=input("Nombre d'"intervalles : ")
X=linspace(.....................)
Y=feval(.....................)
S=.....................
disp(S)
```

Vérifier les résultats en calculant la valeur exacte de cette intégrale.

8.2 Exercices

Exercice 8.1

Pour tout entier naturel $n\geqslant 1$, on définit les fonctions f_n et F_n par :

$$\forall x\in[0,+\infty[,\quad f_n(t)=\frac{1}{(n-1)!}t^{n-1}\mathrm{e}^{-t}\quad F_n(x)=\int_0^x f_n(t)\mathrm{d}t$$

1. Écrire en `Scilab` une fonction d'en-tête `function y=f(t)` retournant la valeur de $f_n(t)$ pour une valeur du réel t fournie en paramètre (l'entier n étant fourni préalablement à cette fonction). *On pourra utiliser la fonction* `factorial(.)`.
2. Écrire une autre fonction d'en-tête `function I=integrale(a,b)` retournant une valeur approchée de l'intégrale $\int_a^b f_n(t)\mathrm{d}t$ par la méthode des rectangles avec un nombre $m=100$ de rectangles (l'entier n étant fourni préalablement à cette fonction).
3. On suppose l'entier n fixé dans $\mathbb{N}^*$.
   (a) Pour $(a,b)\in\mathbb{R}^2$ tel que $a<b$, exprimer $F_n(b)-F_n(a)$ à l'aide d'une intégrale.
   (b) Compléter alors le code suivant (utilisant la fonction `integrale`) pour qu'il calcule et stocke dans un tableau `F` les valeurs de $F_n(x)$ pour tout $x\in\left\{\frac{6}{100}k \mid k\in[\![0,100]\!]\right\}$:
   ```
   x=linspace(..........)
   F=zeros(..........)
   for k=..........
       ..........
   end
   ```
4. En utilisant les trois questions qui précèdent, représenter f_4 et F_4 sur l'intervalle $[0,6]$ sur deux graphiques côte à côte.
5. Conjecturer la valeur de $\lim_{x\to+\infty} F_n(x)$ pour tout entier $n\geqslant 1$.

Exercice 8.2
On pose :

$$\forall x \in \mathbb{R},\ \Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{t^2}{2}}\,\mathrm{d}t$$

et on admet que :

$$\Phi(0) = \frac{1}{2} \qquad \text{et} \qquad \forall x \in \mathbb{R},\ \Phi(-x) = 1 - \Phi(x)$$

1. (a) Vérifier que $\Phi(x) = \frac{1}{2} + \int_0^x \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{t^2}{2}}\,\mathrm{d}t$ pour tout réel $x > 0$.
   (b) En déduire un programme qui calcule une valeur approchée de $\Phi(x)$ pour un réel $x > 0$ fourni par l'utilisateur avec des rectangles de largeur 10^{-k}, l'entier k étant aussi fourni par l'utilisateur.
2. (a) Modifier le programme pour qu'il calcule $\Phi(x)$ pour tout réel x, toujours avec des rectangles de largeur 10^{-k}.
   (b) Modifier le programme pour tracer l'allure de la courbe de Φ sur l'intervalle $[-5, 5]$.

Exercice 8.3 (Méthode des trapèzes ou interpolation affine)
On modifie la méthode des rectangles (ou des sommes de Riemann) par la méthode dite des trapèzes.

1. *Mathématiques* : On note $a_k = a + k\frac{b-a}{n}$ pour tout $k \in [\![0, n]\!]$. Au lieu de calculer l'aire du rectangle de largeur $\frac{b-a}{n}$ et de hauteur $f(a_k)$ pour tout $k \in [\![1, n]\!]$ (rectangle avec le bord droit), on calcule l'aire du trapèze délimité par les points $(a_{k-1}, 0)$, $(a_{k-1}, f(a_{k-1}))$, $(a_k, f(a_k))$ et $(a_k, 0)$ pour chaque entier $k \in [\![1, n]\!]$.
   (a) Préciser l'aire de chaque trapèze et montrer que la somme des aires des n trapèzes est :

$$S_n = \frac{b-a}{n}\left[\frac{f(a)}{2} + \sum_{k=1}^{n-1} f(a_k) + \frac{f(b)}{2}\right]$$

   (b) En déduire que :

$$S_n \xrightarrow[x\to+\infty]{} \int_a^b f(x)\mathrm{d}x$$

2. *Informatique* : Écrire un programme qui calcule la valeur approchée de $\int_a^b \ln(1+x)\mathrm{d}x$ par cette méthode ; a, b et n étant fournis par l'utilisateur. Pour un même entier n, comparer les résultats fournis par la méthode des rectangles et la méthode des trapèzes.

Exercice 8.4 (Interpolation de degré 2)
On remplace l'approximation affine de f de la méthode des trapèzes par une approximation parabolique. Autrement dit, sur chaque intervalle $[a_{k-1}, a_k]$ (toujours avec $a_k - a_{k-1} = \frac{b-a}{n}$), on va remplacer la fonction f par le polynôme $P_k = \alpha_k X^2 + \beta_k X + \gamma_k$ de sorte que, en notant $a_{k-\frac{1}{2}}$ le milieu de $[a_{k-1}, a_k]$ (c'est-à-dire que $a_{k-\frac{1}{2}} = \frac{a_{k-1}+a_k}{2}$), on a :

$$\begin{cases} P_k(a_{k-1}) & = f(a_{k-1}) \\ P_k(a_{k-\frac{1}{2}}) & = f(a_{k-\frac{1}{2}}) \\ P_k(a_k) & = f(a_k) \end{cases}$$

et ainsi :

$$I_n(f) = \sum_{k=1}^{n} \int_{a_{k-1}}^{a_k} P_k(t)\mathrm{d}t \approx \int_a^b f(t)\mathrm{d}t$$

1. Vérifier que la somme $I_n(f)$ approchant l'intégrale vaut :

$$\begin{aligned} I_n(f) &= \sum_{k=1}^{n}\left[\alpha_k\frac{a_k^3 - a_{k-1}^3}{3} + \beta_k\frac{a_k^2 - a_{k-1}^2}{2} + \gamma_k\frac{a_k^1 - a_{k-1}^1}{1}\right] \\ &= \frac{b-a}{n}\sum_{k=1}^{n}\left[\alpha_k\frac{a_k^2 + a_{k-1}a_k + a_{k-1}^2}{3} + \beta_k\frac{a_k + a_{k-1}}{2} + \gamma_k\right] \end{aligned}$$

2. Montrer que :

$$\forall k \in [\![1, n]\!],\ \begin{pmatrix}\alpha_k \\ \beta_k \\ \gamma_k\end{pmatrix} = \begin{pmatrix} a_{k-1}^2 & a_{k-1} & 1 \\ a_{k-\frac{1}{2}}^2 & a_{k-\frac{1}{2}} & 1 \\ a_k^2 & a_k & 1\end{pmatrix}^{-1}\begin{pmatrix} f(a_{k-1}) \\ f(a_{k-\frac{1}{2}}) \\ f(a_k)\end{pmatrix}$$

3. À l'aide des instructions matricielles, écrire un programme qui calcule la valeur approchée de $\int_a^b \ln(1+x)\mathrm{d}x$ par cette méthode, les réels a, b et l'entier n étant fournis par l'utilisateur.

Remarque : En utilisant la base de Lagrange pour la détermination de P_k, on pourrait montrer que

$$\int_{a_{k-1}}^{a_k} P_k(t)dt = \frac{b-a}{n} \times \frac{f\left(a_{k-1}\right) + 4f\left(a_{k-\frac{1}{2}}\right) + f\left(a_k\right)}{6}$$

Chapitre 9

Matrices et résolution de systèmes

9.1 Instructions

Définition 9.1 (Matrices pré-construites) :
- Matrice nulle de $\mathcal{M}_{n,p}(\mathbb{R})$: `zeros(n,p)`.
- Matrice de $\mathcal{M}_{n,p}(\mathbb{R})$ dont tous les coefficients sont égaux à k : `k*ones(n,p)`.
- Matrice unité de $\mathcal{M}_{n,p}(\mathbb{R})$: `eye(n,p)`.
- Matrice diagonale $\text{diag}(a_1,\dots,a_n)$: `diag(a)` ou `diag([a,b,c])`.
- Matrice aléatoire de $\mathcal{M}_{n,p}(\mathbb{R})$ à coefficients dans $[0,1[$: `rand(n,p)`.

Définition 9.2 (Opérations algébriques) :
- Transposée d'une matrice A : `A'` (pour permuter les lignes en colonnes et réciproquement).
- Opérations d'espace vectoriel : somme de deux matrices de même taille `A+B` et produit par un réel `5*A`.
- Produit mathématique de deux matrices : `A*B`, `A^k`, `inv(A)` (inverse : $A * A^{-1} = I$).

Définition 9.3 (Résolution de systèmes) :
- Noyau d'une matrice (solutions de $AX = 0$) : `kernel(A)` (les colonnes de la matrice donnée en réponse forment une base orthonormale du noyau).
- Résolution d'un système linéaire $AX + B = 0$: `[X0,KA]=linsolve(A,B)` (X_0 est une solution particulière de l'équation et K_A est le noyau de A donc les solutions du système sont de la forme $X_0 + H$ avec $H \in K_A$).

9.2 Exercices

Exercice 9.1

Soit $A = \begin{pmatrix} a & b & b \\ b & a & b \\ b & b & a \end{pmatrix}$ où $(a,b) \in \mathbb{R}^2$. Soit $P = \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & \frac{-2}{\sqrt{6}} \end{pmatrix}$.

1. Tester l'inversibilité de la matrice P.
2. Écrire un programme qui calcule $\sum_{k=0}^{n} \frac{1}{k!} A^k$ pour les réels $a = \frac{1}{3}$ et $b = -\frac{1}{2}$.
3. Déterminer le plus petit entier naturel n tel que la valeur absolue du plus grand coefficient de la matrice

$$P \begin{pmatrix} \mathrm{e}^{a+2b} & 0 & 0 \\ 0 & \mathrm{e}^{a-b} & 0 \\ 0 & 0 & \mathrm{e}^{a-b} \end{pmatrix} {}^tP - \sum_{k=0}^{n} \frac{1}{k!} A^k$$

soit inférieur à $\varepsilon = 10^{-12}$ pour les valeurs précédentes des réels a et b.

Exercice 9.2

1. (a) Écrire un script *Scilab* qui affiche deux solutions distinctes dans $\mathbb{R}^3$ du système : $\begin{cases} x+y+z=1 \\ x-y-3z=2 \end{cases}$.

   (b) Déterminer la solution de ce système qui vérifie la contrainte : $y = 8$.

2. (a) Écrire un script *Scilab* qui affiche l'ensemble des solutions dans $\mathbb{R}^4$ du système : $\begin{cases} 2x-y+3z-4t=0 \\ -x+2y-4z+3t=1 \end{cases}$.

   (b) Déterminer la solution de ce système qui vérifie la contrainte : $y = -6$ et $z = 9$.

Exercice 9.3

1. En choisissant au hasard ses coefficients, construire une matrice $A \in \mathcal{M}_4(\mathbb{R})$ telle que tous ses coefficients soient strictement positifs et, sur chaque colonne, la somme de ses coefficients soit égale à 1.
2. Vérifier qu'il existe une unique colonne $X \in \mathcal{M}_{4,1}(\mathbb{R})$ telle que : $AX = X$ et la somme de ses coefficients est égale à 1. La matrice X admet-elle un coefficient strictement négatif ?

Exercice 9.4

Soit $A \in \mathcal{M}_{n,p}(\mathbb{R})$ et $B \in \mathcal{M}_{n,1}(\mathbb{R})$. On souhaite résoudre le système linéaire $AX = B$ dans $\mathcal{M}_{p,1}(\mathbb{R})$.

1. (a) Écrire un script qui résout le système en utilisant la fonction `inv` lorsque cela est possible et `linsolve` sinon.

   (b) Application à la résolution de :

$$\begin{cases} x+y+z &= 1 \\ x-y+z &= 2 \end{cases} \qquad \begin{cases} x+y+z &= 1 \\ x+2y+3z &= 2 \\ x+4y+9z &= 3 \end{cases} \qquad \begin{cases} -2x+y+z &= 1 \\ x-2y+z &= 2 \\ x+y-2z &= 3 \end{cases} \qquad \begin{cases} x+y &= 1 \\ 2x-y &= 3 \\ 3x+2y &= 0 \end{cases}$$

2. On admet le résultat suivant (programme d'algèbre ECS2) : *si $A \in \mathcal{M}_{n,p}(\mathbb{R})$ est de rang p (donc $n \geqslant p$), si $B \in \mathcal{M}_{n,1}(\mathbb{R})$ et si le système $AX = B$ n'admet pas de solution dans $\mathbb{R}^p$ alors la matrice ${}^tAA$ est inversible et le système ${}^tAAX = {}^tAB$ admet une unique solution X_0 appelée meilleure approximation du système $AX = B$ au sens des moindres carrés : $\|AX_0 - B\| < \|AX - B\|$ pour tout $X \neq X_0$ élément de $\mathcal{M}_{p,1}(\mathbb{R})$.*
   Modifier le script précédent pour donner cette meilleure approximation le cas échéant. Tester.

Exercice 9.5 (Pivot de Gauss)

1. Expliquer pourquoi le script suivant détermine une matrice réduite de la matrice A initiale :

```
A=[1,1,1 ; 2,3,4 ; 5,6,7] ; disp(A)
[n,p]=size(A) ; i=0 ; j=0
while j<p
    i=i+1 ; pivot=0
    while (pivot==0)&(j<p) // recherche du plus "grand" pivot non nul pour la ligne i
        j=j+1
        for k=i+1:n
            if abs(A(i,j))<abs(A(k,j)) then C=A(i,:) ; A(i,:)=A(k,:) ; A(k,:)=C ; end
        end
        pivot=A(i,j)
    end
    if pivot<>0 then // réduction avec l'aide de la ligne i
        for k=i+1:n do A(k,:)=A(k,:)-A(k,i)/pivot*A(i,:) ; end
    end
end
disp(A)
```

2. Déterminer le rang de la matrice proposée puis son noyau. Comparer avec le résultat de la réduction obtenue.

Chapitre 10

Espaces vectoriels

10.1 Rang d'une famille de vecteurs

Définition 10.1 :
- `r=rank(A)` : la variable r contient le rang de la matrice A (ou encore le rang de la famille de vecteurs constituant les colonne de dans la matrice A).
- `[n,p]=size(A)` : la variable n (resp. p) contient alors le nombre de lignes (resp. colonnes) de la matrice A.

Exercice 10.1

On choisit une matrice de $\mathcal{M}_5(\mathbb{R})$ dont tous les coefficients sont tirés au hasard dans $[\![-10, 10]\!]$. Compléter le programme qui suit afin qu'il affiche le nombre de matrices A à choisir pour obtenir, pour la première fois, une matrice non inversible.

```
n=5 ; k=0 ; A=eye(n,n) // initialisation avec une matrice inversible
...
...
...
disp(A) ; disp(k)
```

Remarque :

On constatera qu'une matrice tirée au hasard selon ce protocole est très souvent inversible.

10.2 Coordonnées d'un vecteur dans une base

Exercice 10.2

Dans $\mathbb{R}^5$, on considère la famille de vecteurs $(\vec{x}_i)_{1\leqslant i\leqslant 5}$ ainsi que le vecteur $\vec{x}$ représentés par les matrices-colonnes respectives :

$$X_1 = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} \quad X_2 = \begin{pmatrix} 0 \\ -1 \\ 0 \\ -1 \\ 0 \end{pmatrix} \quad X_3 = \begin{pmatrix} -1 \\ 0 \\ -1 \\ -1 \\ 1 \end{pmatrix} \quad X_4 = \begin{pmatrix} 0 \\ -1 \\ -1 \\ -1 \\ 1 \end{pmatrix} \quad X_5 = \begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad X = \begin{pmatrix} 1 \\ -3 \\ 4 \\ 6 \\ 0 \end{pmatrix}$$

1. À l'aide de *Scilab*, montrer que les vecteurs $(\vec{x}_i)_{1\leqslant i\leqslant 5}$ forment une base de $\mathbb{R}^5$.
2. À l'aide de *Scilab*, déterminer les coordonnées de $\vec{x}$ dans cette base.

10.3 Théorème de la base incomplète

Exercice 10.3

Soit $A \in \mathcal{M}_{n,p}(\mathbb{R})$ avec $n > p$ de sorte que $\text{rg}(A) = p$. Les colonnes de A forment donc une famille libre p vecteurs de l'espace vectoriel $\mathbb{R}^n$ (ou $\mathcal{M}_{n,1}(\mathbb{R})$). En procédant à des tirages au sort uniformes dans $[0, 1]$, compléter (directement dans la matrice A) cette famille libre en une base de $\mathbb{R}^n$. On testera le programme sur la matrice A définie dans le début de programme suivant :

```
A=[1,2,3,-4,-5,-6;1,0,-1,0,1,0]' // matrice initiale
...
...
...
disp(A) // matrice finale de taille 6x6 et de rang 6
```

10.4 Extraction d'une famille libre

Exercice 10.4

Soit $A \in \mathcal{M}_{n,p}(\mathbb{R})$ avec $n < p$ de sorte que $\text{rg}(A) = r \leqslant n$. On souhaite extraire des colonnes de A une famille libre de r vecteurs de $\mathbb{R}^n$. Pour cela, deux méthodes sont possibles :

— partir de rien et ajouter successivement les colonnes de A, de gauche à droite, lorsque elles ne sont pas liées aux précédentes (on ajoute une colonne lorsque le rang augmente)
$\rightsquigarrow$ rien // colonne 1 si non nulle // colonne 2 si non colinéaire à colonne 1 // ...

— partir de tout en enlever, de droite à gauche, les colonnes de A qui sont liées à celles qui restent (on enlève une colonne qui ne change pas le rang initial) :
$\rightsquigarrow$ tout // colonne p en moins si liée aux autres // colonne $p-1$ en moins si liée aux autres // ...

1. En utilisant l'une ou l'autre (ou les deux) des méthodes ci-dessus, écrire un programme déterminant une matrice B de rang r dont les colonnes sont des colonnes de A. On testera le programme sur la matrice :

$$A = \begin{pmatrix} 1 & 0 & -1 & 0 & -4 & 0 & 0 & 3 & 0 & 3 \\ 1 & -1 & 0 & 3 & -5 & -1 & -2 & 3 & -1 & 3 \\ -1 & 0 & -1 & -2 & 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 & -1 & -2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 2 & -2 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

2. Une fois ce programme fonctionnel, on le modifiera afin que, de plus, pour chaque colonne éliminée, il donne une relation liant cette colonne aux colonnes conservées.

10.5 Somme de sous-espaces vectoriels

Exercice 10.5

Soit F et G deux sous-espaces vectoriels de $\mathbb{R}^n$ dont on connaît, pour chacun, une base stockée dans les matrices respectives `F` et `G` (les colonnes de ces matrices constituent les vecteurs de ces bases). Écrire un programme qui construit une matrice `FplusG` dont les colonnes forment une base de $F+G$ puis qui indique si la somme est directe ou non. Tester ce programme avec les sous-espaces vectoriels :

$$F = \text{Vect}\left(\begin{pmatrix}1\\1\\1\end{pmatrix}, \begin{pmatrix}1\\2\\3\end{pmatrix}\right) \qquad G = \left(\begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}1\\-2\\-3\end{pmatrix}\right)$$

Exercice 10.6

Soit F et G deux sous-espaces vectoriels de $\mathbb{R}^n$ dont on connaît, pour chacun, une base stockée dans les matrices respectives `F` et `G` (les colonnes de ces matrices constituent les vecteurs de ces bases). Écrire un programme qui construit des matrices `F1`, `FinterG` et `G1` dont les colonnes forment une base respectivement des sous-espaces F_1, $F \cap G$ et G_1 de sorte que l'on a les décompositions $F = F_1 \oplus (F \cap G)$ et $G = (F \cap G) \oplus G_1$. Indiquer ensuite si la somme $F+G$ est directe ou non. Tester ce programme avec les sous-espaces vectoriels :

$$F = \text{Vect}\left(\begin{pmatrix}1\\1\\1\end{pmatrix}, \begin{pmatrix}1\\2\\3\end{pmatrix}\right) \qquad G = \left(\begin{pmatrix}1\\0\\1\end{pmatrix}, \begin{pmatrix}1\\-2\\-3\end{pmatrix}\right)$$

$$F = \text{Vect}\left(\begin{pmatrix}1\\1\\1\\0\\-1\end{pmatrix}, \begin{pmatrix}0\\-1\\-1\\1\\-1\end{pmatrix}, \begin{pmatrix}-1\\-1\\1\\-1\\0\end{pmatrix}, \begin{pmatrix}1\\-1\\-1\\1\\0\end{pmatrix}\right) \qquad G = \text{Vect}\left(\begin{pmatrix}1\\0\\-1\\0\\-1\end{pmatrix}, \begin{pmatrix}0\\-1\\1\\0\\1\end{pmatrix}, \begin{pmatrix}-1\\-1\\1\\1\\1\end{pmatrix}, \begin{pmatrix}0\\1\\1\\1\\-1\end{pmatrix}\right)$$

10.6 Supplémentaire d'un sous-espace vectoriel

Exercice 10.7

1. *Partie mathématique.* Soit F un sous-espace vectoriel de $\mathbb{R}^n$. On note G l'ensemble des vecteurs $\vec{x}$ dont les coordonnées $(x_1, \dots, x_n)$ dans la base canonique vérifie la relation $x_1y_1 + \cdots + x_ny_n$ pour tout vecteur $\vec{y}$ de F.
   (a) Montrer que G est un sous-espace vectoriel de F et que : $\mathbb{R}^n = F \oplus G$.
   (b) Si $F = \text{Vect}(\vec{u}_1, \dots, \vec{u}_p)$, montrer que $\vec{x} \in G$ si et seulement si les relations précédentes sont vraies uniquement pour les vecteurs $(\vec{u}_1, \dots, \vec{u}_p)$.
2. *Partie Scilab.* Déterminer une base de G lorsque : $F = \text{Vect}\left(\begin{pmatrix}-1\\1\\1\\1\\-1\\-1\end{pmatrix}, \begin{pmatrix}-1\\-1\\1\\-1\\0\\-1\end{pmatrix}, \begin{pmatrix}-1\\-1\\0\\0\\-1\\0\end{pmatrix}\right)$

Chapitre 11

Développements limités

11.1 Au voisinage d'un point à différents ordres

Exercice 11.1

1. Construire, sur un même graphique et sur l'intervalle $\left[-\frac{1}{2}, \frac{1}{2}\right]$, les représentations graphiques des fonctions :

$$f : x \mapsto \ln(1+x) \qquad g_n : x \mapsto \sum_{k=0}^{n} \frac{f^{(k)}(0)}{k!} x^k \text{ pour } n \in [\![0, 4]\!] \cup \{20\}$$

2. Procéder de même pour les autres fonctions associées aux développements limités usuels au voisinage de 0 :

$$f : x \mapsto \frac{1}{1+x} \qquad f : x \mapsto \sqrt{1+x} \qquad f : x \mapsto \mathrm{e}^x \qquad f : x \mapsto \cos(x) \qquad f : x \mapsto \sin(x)$$

11.2 Élargissement de l'intervalle au domaine de définition

Exercice 11.2

1. Reprendre l'exercice du paragraphe 11.1 et élargir l'intervalle à :
   (a) $[-0.99, 0.99]$ puis $[-0.99, 1.99]$ pour les fonction $x \mapsto \ln(1+x)$, $x \mapsto \frac{1}{1+x}$ et $x \mapsto \sqrt{1+x}$,
   (b) $[-1, 1]$ puis $[-2, 2]$ puis $[-10, 10]$ pour les fonctions $x \mapsto \mathrm{e}^x$, $x \mapsto \cos(x)$ et $x \mapsto \sin(x)$.
2. Expliquer les différences constatées entre les deux types de fonctions.
   On pourra aussi faire varier les valeurs de l'entier n représentant l'ordre pour effectuer les observations.

11.3 Au voisinage de l'infini à différents ordres

Exercice 11.3

1. Déterminer le développement limité à l'ordre 3 au voisinage de $+\infty$ de la fonction $f : x \mapsto \left(1 + \frac{1}{x}\right)^x$.
2. Pour $n \in [\![0, 3]\!]$, on note g_n la fonction donnant la partie régulière du développement limité de f à l'ordre n. Construire, sur un même graphique et sur l'intervalle $[1, 10]$, les représentations graphiques des fonctions f et g_n pour tout $n \in [\![0, 3]\!]$.

Chapitre 12

Séries

12.1 Rappel sur les calculs de sommes

Définition 12.1 :
Soit $(u_n)_{n\geqslant 0}$ une suite réelle. On considère la série $\sum_{n\geqslant 0} u_n$ à travers sa suite $(S_n)_{n\geqslant 0}$ des sommes partielles définie par $S_n = \sum_{k=0}^{n} u_k$ pour tout entier $n \geqslant 0$. On suppose que les réels $u_0, u_1, \ldots, u_m$ sont stockés dans une matrice ligne `u` (cellules numérotées de 1 à $m+1$).

- Soit $n \in [\![0, m]\!]$. Pour calculer le seul terme S_n, deux méthodes :
  - boucle : `S=0 ; for k=1:n+1 S=S+u(k) ; end`
  - instruction particulière : `S=sum(u(1:n+1))`
- Pour calculer tous les termes $S_0, S_1, \ldots, S_m$ de la suite des sommes partielles :, deux méthodes :
  - boucle : `S=zeros(u) ; S(1)=u(1) ; for k=2:m+1 S(k)=S(k-1)+u(k) ; end`
  - instruction particulière : `S=cumsum(u)`
  - représentation graphique : `N=[0:m] ; plot2d3(N,S,5) // trace des segments verticaux`

Si les termes de la suite (u_n) ne sont pas stocké, on les calcule au fur et à mesure dans la boucle for juste avant le calcul de `S` ou bien on les calculs tous auparavant pour les avoir dans un tableau et se ramener à ce qui précède.

Exercice 12.1

1. Calculer et afficher $\sum_{k=0}^{50} \frac{1}{k+1}$. Représenter graphiquement la suite $\left(\sum_{k=0}^{n} \frac{1}{k+1}\right)_{0\leqslant n\leqslant 50}$
2. Calculer, afficher et représenter graphiquement les 50 premiers termes de la suite des sommes partielles de la série $\sum_{n\geqslant 1} \frac{(-1)^{n-1}}{n}$.

12.2 Représentations de séries dépendant d'un paramètre

Exercice 12.2 (Séries de Riemann, fonction zêta de Riemann)

Pour tout $x \in]1, +\infty[$, on pose :

$$\zeta(x) = \sum_{n=1}^{+\infty} \frac{1}{n^x} \qquad (\textit{fonction zêta de Riemann})$$

1. *Mathématiques.* Justifier que la fonction ζ est bien définie puis montrer qu'elle est décroissante sur $]1, +\infty[$.
2. *Programmation.* En utilisant des sommes partielles de rang 1000 comme approximation de $\zeta(x)$:
   (a) représenter graphiquement la fonction ζ sur l'intervalle $[1.01; 5]$,
   (b) déterminer une approximation de la solution x_0 de l'équation $\zeta(x) = x$ sur l'intervalle $[1.01; 5]$.

Exercice 12.3 (Autre fonction somme d'une série)

Pour tout $x \in]0, +\infty[$, on pose :

$$f(x) = \sum_{n=0}^{+\infty} \frac{n + \frac{1}{x} + x}{n^3 + \frac{1}{\sqrt{x}}}$$

1. *Mathématiques.* Montrer que f est définie sur $]0, +\infty[$.
2. *Programmation.* En utilisant des sommes partielles de rang 1000 comme approximation de $f(x)$:
   (a) représenter graphiquement la fonction f sur l'intervalle $[0.01; 5]$,
   (b) déterminer une approximation du réel x_0 en lequel la fonction f admet son minimum global sur l'intervalle $[0.01; 5]$.

12.3 Convergence d'une série à signes alternés

Exercice 12.4 (Convergence de la série régulièrement alternée)
Soit $(a_n)_{n\in\mathbb{N}}$ une suite réelle décroissante et de limite nulle. Pour tout $n \in \mathbb{N}$, on pose : $S_n = \sum_{k=0}^{n} (-1)^k a_k$.

1. *Mathématiques.*
   (a) Montrer que les suites $(S_{2n})_{n\in\mathbb{N}}$ et $(S_{2n+1})_{n\in\mathbb{N}}$ sont adjacentes. Quelle est la nature de la série $\sum_{n\geqslant 0} (-1)^n a_n$?
   (b) Soit ℓ la somme de cette série. Montrer que, pour tout $\varepsilon > 0$, si $a_{2n+1} \leqslant \varepsilon$ alors $|S_{2n+1} - \ell| \leqslant \varepsilon$.
2. *Programmation.* Dans cette question, on suppose que : $\forall n \in \mathbb{N},\ a_n = \frac{1}{n+1}$.
   (a) Programmer la détermination d'une valeur approchée de la limite de la suite $(S_n)_{n\in\mathbb{N}}$ à 10^{-p} près où $p \in \mathbb{N}^*$.
   (b) Modifier le programme pour qu'il affiche la première valeur de l'entier n qui réalise cette condition lorsque $p \in [\![1, 6]\!]$. Commenter ce résultat.
3. *Programmation.* Représenter la fonction $f : x \mapsto \sum_{n=1}^{+\infty} \frac{(-1)^{n-1}}{n^x}$ sur l'intervalle $[0.1; 5]$.

12.4 Changement de l'ordre des termes d'une série semi-convergente

Exercice 12.5 (Cas de la série régulièrement alternée)
Pour tout $n \in \mathbb{N}$, on pose : $u_n = \frac{(-1)^n}{n+1}$. Soit $L \in \mathbb{R}$. On pose : $L_0 = u_0$, $p_0 = 0$, $q_0 = 0$ et pour tout entier $n \in \mathbb{N}$:

$$(p_{n+1}, q_{n+1}) = \begin{cases} (p_n + 1, q_n) & \text{si } L_n < L \\ (p_n, q_n + 1) & \text{si } L_n \geqslant L \end{cases} \qquad L_{n+1} = \begin{cases} L_n + u_{2p_{n+1}} & \text{si } L_n < L \\ L_n + u_{2q_{n+1}-1} & \text{si } L_n \geqslant L \end{cases}$$

1. (a) *Programmation.* Écrire un programme *Scilab* qui calcule le terme de rang n de la suite $(L_n)_{n\in\mathbb{N}}$. On affichera, à chaque rang $k \in [\![0, n]\!]$, les valeurs de p_k, q_k et L_k. La suite $(L_n)_{n\in\mathbb{N}}$ converge-t-elle lorsque $L = 3$? lorsque $L = -4$?
   (b) *Mathématiques.* Dans le cas où $L > 1$, écrire L_n en fonction de n, de $p_0, \ldots, p_n$, de $q_0, \ldots, q_n$ et des réels u_k. Vérifier que la suite $(L_n)_{n\in\mathbb{N}}$ recouvre bien tous les termes de la suite $(u_n)_{n\in\mathbb{N}}$.
2. Modifier le programme pour que la suite $(L_n)_{n\in\mathbb{N}}$ diverge vers $+\infty$ tout en recouvrant tous les termes de la suite $(u_n)_{n\in\mathbb{N}}$.

Chapitre 13

Simulation de variables discrètes infinies

13.1 Lois usuelles

Théorème 13.1 (Simulation de variables aléatoire suivant les lois discrètes usuelles) :

— Loi géométrique $\mathcal{G}(p)$: `X=1 ; while rand()>p ; X=X+1 ; end`

— Loi de Poisson $\mathcal{P}(\lambda)$: comme il n'y a pas de modèle expérimental, une possibilité consiste à utiliser l'approximation $\mathcal{P}(\lambda) \approx \mathcal{B}\left(n, \frac{\lambda}{n}\right)$ pour n « assez grand »
`X=0 ; for k=1:n if rand()<lambda/n then X=X+1 ; end ; end`
Une autre possibilité consiste à utiliser la méthode « d'inversion » décrite dans l'exercice 13.5

13.2 Exercices

Exercice 13.1 (Un temps d'attente)
Soit n un entier naturel non nul. On considère n urnes $U_1, \dots, U_n$, l'urne numéro k comportant k boules blanches et $n+1-k$ boules noires. On choisit une urne au hasard puis on tire successivement et avec remise une boule dans cette urne jusqu'à obtenir pour la première fois une boule blanche. On note X_n le nombre de tirages effectués.

1. Écrire une fonction qui reçoit un entier n en paramètre et qui renvoie une réalisation de la variable aléatoire X_n. On écrira son en-tête sous la forme : `function Xn=premiere_blanche(n)`.
2. Compléter ce script afin de représenter la loi de probabilité de X_n.

Exercice 13.2 (Un temps d'attente)
Une urne contient b boules blanches et une boule rouge.

1. Première situation.
On effectue des tirages successifs d'une boule avec remise de la boule tirée entre deux tirages accompagnée d'une boule rouge. Le jeu s'arrête dès que l'on tire pour la première fois une boule rouge. On note alors X le numéro du dernier tirage effectué (si toutefois le jeu s'arrête).
Écrire une fonction qui reçoit un entier b en paramètre et qui renvoie une réalisation de la variable aléatoire X.
2. Seconde situation.
Entre chaque tirage, toujours en remettant la boule tirée, on ajoute une boule verte (au lieu d'une rouge) et les tirages cessent dès que la boule tirée n'est pas blanche. On note X le numéro du dernier tirage effectué (si toutefois le jeu s'arrête) et on pose :

$$Y = \begin{cases} 2X & \text{si on termine avec la boule rouge} \\ 2X-1 & \text{si on termine avec une boule verte} \end{cases}$$

   (a) Écrire une fonction qui reçoit un entier b en paramètre et qui renvoie une réalisation de la variable aléatoire Y puis compléter ce script afin de représenter la loi de probabilité de Y.
   (b) Compléter ce script afin qu'il calcule et affiche une estimation de la probabilité p que les tirages s'arrêtent sur l'obtention de la boule rouge.

Exercice 13.3 (Un temps d'attente)
Soit n un entier naturel non nul. On considère une urne contenant n boules numérotées de 1 à n. On tire successivement et avec remise entre les tirages une boule jusqu'à obtenir pour la première fois tous les numéros possibles. On note X_n le nombre de tirages effectués.

1. Écrire une fonction qui reçoit un entier n en paramètre et qui renvoie une réalisation de la variable aléatoire X_n. *Indication* : On pourra utiliser un tableau `U` à une ligne et n colonnes dont la cellule `U(i)` (pour $i \in [\![1, n]\!]$) vaut 0 tant que la boule numéro i n'a pas été tirée et vaut 1 sinon.
2. Compléter ce script afin de représenter la loi de probabilité de X_n.

Exercice 13.4 (Longueur de séries consécutives pile/face)
On lance indéfiniment une pièce dont la probabilité d'apparition du côté *pile* est p (avec $p \in]0,1[$) et celle du côté *face* est $q = 1-p$. Pour $k \in \mathbb{N}^*$, on note L_k la longueur de la k-ème série de résultats consécutifs identiques. Par exemples :

$$\omega = (P,P,P,F,F,F,F,P,F,F,F,P,\dots) \quad \text{donne} \quad L_1(\omega)=3 \quad L_2(\omega)=4 \quad L_3(\omega)=1 \quad L_4(\omega)=3$$

$$\omega = (F,P,P,F,P,F,F,F,F,F,F,P,\dots) \quad \text{donne} \quad L_1(\omega)=1 \quad L_2(\omega)=2 \quad L_3(\omega)=1 \quad L_4(\omega)=1 \quad L_5(\omega)=6$$

1. Écrire une fonction qui reçoit un entier r et un réel p en paramètres et qui renvoie une réalisation des variables aléatoires $L_1, L_2, \dots, L_r$.
2. Compléter ce script pour calculer une estimation des espérances $\mathbb{E}(L_1), \mathbb{E}(L_2), \dots, \mathbb{E}(L_r)$ lorsque $r = 4$ et $p = 0,3$.

Exercice 13.5 (Simulation d'une loi discrète quelconque par "inversion de la répartition")

1. *Mathématiques.* Soit $(x_n)_{n\in\mathbb{N}}$ une suite réelle strictement croissante et non majorée. Soit $(p_n)_{n\in\mathbb{N}}$ une suite de réels strictement positifs tels que la série $\sum_{n\in\mathbb{N}} p_n$ converge et sa somme vaut 1.
   (a) Justifier qu'il existe une variable aléatoire discrète X telle que $\forall n \in \mathbb{N}$, $\mathbb{P}(X = x_n) = p_n$.
   (b) Soit $t \in [0,1[$. Montrer que l'ensemble $N_t = \{n \in \mathbb{N} \mid p_0 + \cdots + p_n \geqslant t\}$ admet un minimum $n(t)$ dans $\mathbb{N}$. En déduire que $\mathbb{P}\left(X \leqslant x_{n(t)-1}\right) < t \leqslant \mathbb{P}\left(X \leqslant x_{n(t)}\right)$.
   (c) Soit $U \hookrightarrow \mathcal{U}([0,1[)$. Montrer que la variable aléatoire $x_{n(U)}$ a même loi que X.
2. *Informatique. Pour simuler la variable aléatoire X, on va donc simuler la variable aléatoire $x_{n(U)}$.*
   (a) Quelle variable aléatoire est simulée par la fonction `Scilab` suivante ? Justifier.

```
function n=mavariable(t)
    n=0 ; p=exp(-lambda) ; F=p
    while F<t
        n=n+1 ; p=p*lambda/n ; F=F+p
    end
endfunction
```

   Compléter ce script afin qu'il trace l'histogramme des fréquences observées des résultats lors de $N = 10^5$ simulations de cette variable aléatoire.
   (b) Simuler la variable aléatoire X telle que $\mathbb{P}\left(X = 2^{n-1}\right) = \frac{1}{n(n+1)}$ pour tout $n \in \mathbb{N}^*$.

Chapitre 14

Lois de probabilité usuelles

14.1 Fonction `grand`

Définition 14.1 :
La fonction `grand` permet de simuler des lois de probabilités usuelles, aussi bien celles au programme de première année que celles au programme de seconde année. Ainsi :

- `Y=grand(m,n,"nom_loi",paramètre(s))` permet de créer une matrice Y de taille (m,n) dont chacun des éléments est une réalisation d'une variable aléatoire dont la loi de probabilité et ses paramètres sont ceux indiqués,
- ou bien `Y=grand(X,"nom_loi",paramètre(s))` avec X matrice dont la taille déjà connue permet de créer une matrice Y de même taille que X et possédant les caractéristiques décrites ci-avant.
- Les lois discrètes usuelles sont :
  - loi uniforme $\mathcal{U}(\llbracket a,b \rrbracket)$: `grand(nb_lig,nb_col,"uin",a,b)`,
  - loi binomiale $\mathcal{B}(N,p)$: `grand(nb_lig,nb_col,"bin",N,p)`,
  - loi géométrique $\mathcal{G}(p)$: `grand(nb_lig,nb_col,"geom",p)`,
  - loi de Poisson $\mathcal{P}(\lambda)$: `grand(nb_lig,nb_col,"poi",lambda)`.
- Les lois à densité usuelles sont :
  - loi uniforme $\mathcal{U}([a,b])$: `grand(nb_lig,nb_col,"unf",a,b)`,
  - loi exponentielle $\mathcal{E}(\lambda)$: `grand(nb_lig,nb_col,"exp",1/lambda)`,
  - loi normale $\mathcal{N}(m,\sigma^2)$: `grand(nb_lig,nb_col,"nor",m,sigma)`.

Remarque :
L'instruction `grand(nb_lig,nb_col,"uin",a,b)` remplace donc l'instruction `floor((b-a)*rand(nb_lig,nb_col)+a)`. Elle semble même « plus aléatoire ».

14.2 Exercices

Exercice 14.1
Soit X_1, X_2, X_3, X_4 quatre variables aléatoires indépendantes de même loi $\mathcal{U}([0,1])$ définies sur un même espace probabilisé $(\Omega, \mathcal{T}, \mathbb{P})$. Pour tout $\omega \in \Omega$, on pose :

$$A(\omega) = \begin{pmatrix} X_1(\omega) & X_2(\omega) \\ X_3(\omega) & X_4(\omega) \end{pmatrix} \in \mathcal{M}_2(\mathbb{R}) \qquad D(\omega) = \det(A(\omega)) = X_1(\omega)X_4(\omega) - X_2(\omega)X_3(\omega) \in \mathbb{R}$$

On admet que D est une variable aléatoire définie sur ce même espace probabilisé. En réalisant 10^6 fois cette expérience aléatoire, tracer l'allure de la fonction de répartition de la variable aléatoire D et estimer son espérance et sa variance.

Exercice 14.2 (Temps d'attente du k^e pile)
1. Écrire un programme qui simule l'expérience aléatoire dont le résultat est le temps d'attente du k^e *pile* lors de lancers successifs d'une pièce truquée de sorte que le côté *pile* apparaît avec la probabilité p (l'entier k et le réel p étant fournis par l'utilisateur).
2. On réalise 10000 fois cette expérience. Calculer une valeur approchée du nombre moyen de lancers réalisé à chaque expérience.

Exercice 14.3 (Test de la "qualité" des inégalités de Markov et de Bienaymé-Tchebychev)
Soit $X \hookrightarrow \mathcal{P}(\lambda)$ où λ est un réel strictement positif donné. Soit a un réel strictement positif.
1. Que fait l'instruction `Scilab` suivante : `X=grand(1,100000,"poi",3)` ?
2. Écrire une instruction calculant une estimation de $\mathbb{P}(X \geqslant a)$ en utilisant les données du tableau `X` qui précède pour tout $a \in \llbracket 1, 10 \rrbracket$. Comparer ces estimations de $\mathbb{P}(X \geqslant a)$ avec $\frac{\mathbb{E}(X)}{a}$. (majorant dans l'inégalité de Markov).
3. Évaluer la « qualité » de l'inégalité de Bienaymé-Tchebychev.

Exercice 14.4
On considère une pièce truquée de sorte que le côté *pile* apparaît avec la probabilité p. On effectue des lancers jusqu'à obtenir une première fois le côté *pile*. X étant le nombre de lancers réalisés, on effectue ensuite X nouveaux lancers et on compte le nombre Y de *pile* obtenus au cours de cette nouvelle séquence de lancers.

1. Écrire un programme qui simule la variable aléatoire Y (le réel p étant fourni par l'utilisateur).
2. Calculer une valeur approchée de l'espérance de Y.

Exercice 14.5

Soit λ un réel strictement positif. Soit n un entier naturel strictement supérieur à λ.

1. Construire une matrice-ligne à n colonnes ne comportant que des 0 ou des 1 choisis « au hasard » de sorte que chaque coefficient vaut 1 avec la probabilité $\frac{\lambda}{n}$.
   Justifier que la somme des coefficients de cette matrice suit la loi $\mathcal{B}\left(n, \frac{\lambda}{n}\right)$.
2. Construire une matrice A à 1000 lignes et n colonnes sur le même principe. Construire la matrice-colonne B dont le coefficient de la ligne i est égal au nombre de 1 dans la ligne i de la matrice A.
3. Représenter l'histogramme des valeurs de la matrice B. Tester pour $\lambda = 5$ et $n \in \{20; 50; 100\}$. Qu'a-t-on représenté lorsque n est grand ?

Exercice 14.6 (Convergence en loi)

Soit λ un réel strictement positif. Dans un espace probabilisé $(\Omega, \mathcal{T}, \mathbb{P})$, on considère une variable aléatoire X de loi $\mathcal{P}(\lambda)$ et, pour tout entier $n \geqslant \lambda$, une variable aléatoire X_n de loi $\mathcal{B}\left(n, \frac{\lambda}{n}\right)$.

1. On considère le script suivant que l'on testera avec $\lambda = 1$, $\lambda = 2$ et $\lambda = 3$:

```
lambda=input("Donner un réel lambda >0 : ")
N=10000 ; xmax=floor(4*lambda) ; X=grand(N,1,"poi",lambda)
for k=1:6
    n=(floor(lambda+1))*3^(k-1) ; Xn=grand(N,1,"bin",n,lambda/n)
    subplot(2,3,k) ; histplot([-0.5:xmax+0.5],Xn,2) ; histplot([-0.5:xmax+0.5],X,5)
end
```

2. Interpréter ce script. Quel théorème du cours de mathématiques illustre-t-il ?

Exercice 14.7 (Une expérience aléatoire)

À l'entrée d'un restaurant, n personnes confient leur chapeau respectif (tous deux à deux distincts) à la réception. À la sortie, le réceptionniste, très joueur, décide de les rendre au hasard. Pour modéliser cette expérience aléatoire et donner la liste des personnes qui ont obtenu leur propre chapeau, on peut procéder de la façon suivante :

— On crée un tableau E à une seule ligne contenant initialement les entiers de 1 à n (les numéros des chapeaux dans l'ordre de leur dépôt à la réception) et qui contiendra, au fur et à mesure, les numéros (toujours dans l'ordre croissant) des chapeaux non encore distribués.

— On tire au sort successivement et sans remise tous les numéros.

— Au fur et à mesure des tirages, on complète un tableau S à une seule ligne qui contiendra les numéros des chapeaux dans l'ordre de leurs tirages. À la fin de chaque itération, la concaténation des tableaux E et S doit donner tous les numéros.

— Les tirages étant tous effectués, le chapeau i est à son propriétaire si et seulement si $S(i) - i = 0$.

1. Tester ainsi le script qui suit :

```
n=input("Donner n : ")
E=[1:n] // initialisation : liste des chapeaux non distribués
S=[] // initialisation : liste ordonnée des chapeaux distribués
for i=1:n // mise en place des répétitions
    k=grand(1,1,"uin",1,length(E)) // choix d'une case de E
    S=[S,E(k)] // on ajoute le numéro de chapeau à la liste S
    E=[E(1:k-1),E(k+1:length(E))] // on retire de E le numéro tiré
end
L=find(S-[1:n]==0) // liste des personnes ayant leur propre chapeau
disp(S) ; disp(L) // affichage de l'ordre des tirages puis de la liste
```

$$\texttt{E} = \begin{bmatrix}1 & 2 & 3 & 4 & 5\end{bmatrix} \qquad \texttt{S} \nexists [$$

$$k = 3 \hookrightarrow \mathcal{U}([\![1,5]\!]) \qquad \texttt{E(k)} = 3 \qquad \texttt{E} = \begin{bmatrix}1 & 2 & 4 & 5\end{bmatrix} \qquad \texttt{S} = \begin{bmatrix}3\end{bmatrix}$$

$$k = 3 \hookrightarrow \mathcal{U}([\![1,4]\!]) \qquad \texttt{E(k)} = 4 \qquad \texttt{E} = \begin{bmatrix}1 & 2 & 5\end{bmatrix} \qquad \texttt{S} = \begin{bmatrix}3 & 4\end{bmatrix}$$

$$k = 1 \hookrightarrow \mathcal{U}([\![1,3]\!]) \qquad \texttt{E(k)} = 1 \qquad \texttt{E} = \begin{bmatrix}2 & 5\end{bmatrix} \qquad \texttt{S} = \begin{bmatrix}3 & 4 & 1\end{bmatrix}$$

$$k = 1 \hookrightarrow \mathcal{U}([\![1,2]\!]) \qquad \texttt{E(k)} = 2 \qquad \texttt{E} = \begin{bmatrix}5\end{bmatrix} \qquad \texttt{S} = \begin{bmatrix}3 & 4 & 1 & 2\end{bmatrix}$$

$$k = 1 \hookrightarrow \mathcal{U}([\![1,1]\!]) \qquad \texttt{E(k)} = 5 \qquad \texttt{E} \nexists [\qquad \texttt{S} = \begin{bmatrix}3 & 4 & 1 & 2 & 5\end{bmatrix}$$

$$\texttt{S} - \begin{bmatrix}1 & 2 & 3 & 4 & 5\end{bmatrix} = \begin{bmatrix}2 & 2 & -2 & -2 & 0\end{bmatrix}$$

2. Soit X la variable aléatoire égale au nombre de personnes ayant leur propre chapeau à l'issue de cette expérience aléatoire. À l'aide du script qui précède :
   (a) Construire, pour un entier n donné, l'histogramme des fréquences observées des valeurs prises par X au cours de $N = 10^4$ répétitions de l'expérience.
   (b) Comparer cet histogramme avec celui de la loi de Poisson de paramètre 1 (pour différentes valeurs n).
   (c) Estimer le nombre moyen de personnes qui obtiennent leur propre chapeau pour un entier n donné.

Exercice 14.8

1. On lance deux dés cubiques non truqués dont les faces sont numérotées de 1 à 6.
   (a) Afficher la loi approchée de la variable aléatoire égale à la somme des numéros affichés par les deux dés et tracer l'histogramme correspondant.
   (b) Déterminer et afficher la somme la plus fréquente. Était-ce prévisible ?
2. Même question avec k dés ($k \geqslant 2$) ayant chacun p faces numérotées de 1 à p.

Chapitre 15

Résolution d'équations $f(x) = 0$

15.1 Méthode de dichotomie

Exercice 15.1 (Résolution d'équation par dichotomie)
On veut résoudre l'équation $f(x) = 0$ par la méthode dite de **dichotomie**. On en rappelle le principe :

— Soit $]a, b[$ un intervalle contenant une unique solution x_0 de l'équation $f(x) = 0$ pour f continue sur $[a, b]$ (donc $f(a)f(b) < 0$).

— Alors la solution x_0 est dans l'intervalle $\left]a, \frac{a+b}{2}\right]$ ou bien dans l'intervalle $\left]\frac{a+b}{2}, b\right[$ selon que $f(a)f(\frac{a+b}{2}) \leqslant 0$ ou bien que $f(a)f(\frac{a+b}{2}) > 0$. On définit donc, par récurrence, les suites $(a_n)_{n\geqslant 0}$ et $(b_n)_{n\geqslant 0}$ par : $a_0 = a$, $b_0 = b$ et pour tout entier naturel n :

$$a_{n+1} = \begin{cases} a_n & \text{si } f(a_n)\, f\left(\frac{a_n+b_n}{2}\right) \leqslant 0 \\ \frac{a_n+b_n}{2} & \text{si } f(a_n)\, f\left(\frac{a_n+b_n}{2}\right) > 0 \end{cases} \quad b_{n+1} = \begin{cases} \frac{a_n+b_n}{2} & \text{si } f(a_n)\, f\left(\frac{a_n+b_n}{2}\right) \leqslant 0 \\ b_n & \text{si } f(a_n)\, f\left(\frac{a_n+b_n}{2}\right) > 0 \end{cases}$$

1. Ces deux suites sont adjacentes (pourquoi ?), convergent vers x_0 (pourquoi ?) et a_n est une approximation de x_0 à $(b_n - a_n)$ près par défaut (pourquoi ?).
2. Calcule une valeur approchée à 10^{-k} près par défaut de la solution (existe ? unique ?) de l'équation $x + \ln(x) = 0$ dans l'intervalle $[a, b]$, a, b, k étant fournis par l'utilisateur. Afficher aussi le nombre d'itérations réalisées.
3. Résoudre l'équation $\tan(x) = x$ dans l'intervalle $\left]\frac{\pi}{2}, \frac{3\pi}{2}\right[$.
4. Résoudre dans $\mathbb{R}$ l'équation $\cos(x) = x$.

15.2 Algorithme de Newton

Exercice 15.2 (Résolution d'équation par l'algorithme de Newton)
On veut résoudre l'équation $f(x) = 0$ par la méthode dite de **l'algorithme de Newton**.

— Soit I un intervalle contenant une unique solution x_0 de l'équation $f(x) = 0$ pour f dérivable et de dérivée non nulle sur I et $a \in I$ (donc $f'(a) \neq 0$).

— On définit par récurrence la suite $(a_n)_{n\geqslant 0}$ par : $a_0 = a$ et, pour tout entier naturel n, $a_{n+1} = a_n - \frac{f(a_n)}{f'(a_n)} = g(a_n)$ pourvu que $g(I) \subset I$.

1. Montrer que, pour tout entier n, a_{n+1} est l'abscisse du point d'intersection de l'axe des abscisses avec la tangente à la courbe de f au point d'abscisse a_n. On admet que, dans des conditions adéquates, la suite $(a_n)_{n\geqslant 0}$ converge et que sa limite est solution de l'équation $f(x) = 0$.
2. Calculer une valeur approchée à 10^{-k} près de la solution de l'équation $x^2 - r = 0$ $(r > 0)$ dans l'intervalle $]0, +\infty[$, r, a, k étant fournis par l'utilisateur. Afficher aussi le nombre d'itérations réalisées.
   On admet que a_n est une valeur approchée de cette solution à 10^{-k} près lorsque $|a_{n+1} - a_n| \leqslant 10^{-k}$.
3. Résoudre l'équation $x + \ln(x) = 0$ dans l'intervalle $]0, +\infty[$. Afficher aussi le nombre d'itérations réalisées. Comparer avec la méthode de la dichotomie.

Chapitre 16

Dérivation

Chapitre 17

Loi binomiale

17.1 Simulation d'une loi binomiale

Exercice 17.1

1. Compléter la fonction suivante afin qu'elle simule la réalisation de n tirages indépendant succès/échec (p est la probabilité d'un succès et $1-p$ celle d'un échec) et retourne le nombre X de succès obtenus :
   ```
   function X=binome(n,p)
   ...
   endfunction
   ```
2. Pour $n = 10$ et $p = \frac{1}{3}$, écrire une boucle qui réalise $m = 100000$ fois cette expérience et stocke les résultats dans un tableau `X`.
   Remarque : On peut obtenir ce tableau sans utiliser de boucle grâce à l'instruction `feval` :
   `X=feval(n*ones(1,m),[p],binome)`

17.2 Représentation des fréquences observées des différents résultats

Exercice 17.2

Représenter l'histogramme des fréquences obtenues à partir du tableau X de l'exercice qui précède.

17.3 Fréquences cumulées croissantes

Exercice 17.3

À partir du tableau `X` du premier exercice, construire un tableau `LX` à 3 colonnes tel que la première colonne contient la listes des valeurs deux à deux distinctes contenues dans le tableau `X`, la seconde contient les fréquences d'apparition de ces valeurs parmi tous les éléments de `X` et, enfin, la troisième contient les fréquences cumulées croissantes correspondantes.

17.4 Représentation de la fonction de répartition

Exercice 17.4

On souhaite représenter une approximation de la fonction de répartition d'une loi binomiale à partir des données contenues dans le tableau `LX` de l'exercice qui précède. Compléter le code qui suit afin qu'il trace la fonction en escalier (sans « trait vertical ») représentant la fonction de répartition sur l'intervalle $[-1, n+1]$ (toujours avec $n = 10$ et $p = \frac{1}{3}$) :

```
x=[-1;LX(:,1);n+1] ; y=[0;LX(:,3)]
...
```

Remarque :

On aurait pu tracer directement la répartition en calculant les probabilités exactes sans effectuer de tirages au sort.

17.5 Exercice supplémentaire

Exercice 17.5

En utilisant l'instruction `subplot` (voir chapitre précédent), représenter dans une même fenêtre l'histogramme et la répartition l'un sous l'autre pour un même couple (n, p) pour $n \in \{4, 10, 40, 100, 1000\}$(sur la hauteur de la fenêtre) et $p \in \left\{\frac{1}{10}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{9}{10}\right\}$ (sur la largeur de la fenêtre).

Chapitre 18

Coefficients binomiaux

18.1 Exercices

Exercice 18.1

En utilisant la relation $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$, calculer $\binom{n}{k}$ et vérifier que $\binom{20}{10} = 184756$.

Exercice 18.2

1. À l'aide de la formule d'addition de Pascal, construire le triangle de Pascal dans une matrice de taille n, l'entier n étant fourni par l'utilisateur.
2. À l'aide de la formule d'addition de Pascal généralisée rappelée ci-dessous, construire le triangle de Pascal dans une matrice de taille n, l'entier n étant fourni par l'utilisateur. Pour $0 \leqslant k \leqslant h$:

$$\binom{h}{k} = \sum_{i=k}^{h} \binom{i}{k-1}$$