
UNIVERSITÉ FÉLIX HOUPHOUËT-BOIGNY
ABIDJAN-COCODY

UFR Mathématiques et Informatique

Cours de logiciels Scientifiques (Latex, Scilab et R)

MATHS LICENCE 2

Dr DOSSO Mouhamadou

E-mail : mouhamadou.dosso@univ-fhb.ci
mouhamadoudoss@yahoo.fr

Table des matières

1	Logiciel de traitement de texte scientifique : LATEX 2_ε	3
1.1	Overview	3
1.1.1	Édition d'un document	3
1.1.2	Compilation et visualisation d'un document tex	3
1.1.3	Structuration du document	4
1.1.4	Structure de documents types	5
1.2	Formules Mathématiques	6
1.2.1	Les environnements	6
1.2.2	Symboles mathématiques	7
1.2.3	constructions mathématiques	8
1.3	Images	10
1.3.1	Dessins à inclure	10
1.3.2	Dessins à inclure	10
1.4	Bibliographie, Index	11
1.4.1	Approche manuelle	11
1.4.2	Les commandes LATEX	12
1.5	Exercices de Travaux Pratiques	12
2	Logiciel Scilab	19
2.1	Fonctionnement général	19
2.2	Type de données	19
2.2.1	Principales opérations sur les matrices.	21
2.2.2	Autres objets mathématiques	21
2.3	Programmation	22
2.3.1	Ediction de scripts	22
2.3.2	Fonctions ou macros ("function" or "macros'...")	23
2.3.3	Boucles	24
2.3.4	Boucles (while)	25
2.3.5	Tests	25
2.3.6	Les opérations logiques dans le tests	26
2.3.7	Utilisation de fonctions Scilab	26
2.4	Entrées/Sortie sous Scilab	27

2.4.1	Interaction avec l'utilisateur	27
2.4.2	Sauvegarde de résultats en binaire	28
2.5	Sorties graphiques	29
2.5.1	Les tracés en 2 dimensions	29
2.5.2	Graphiques dans l'espace	30
2.6	Exercices de Travaux Pratiques	31
3	Logiciel R	35
3.1	Interface d'utilisation sous windows	35
3.1.1	Fonction des menus	35
3.1.2	D'autres fenêtres	35
3.2	Les objets : Vecteurs, Matrices, Matrices à plus de deux dimensions, liste et structures de données	36
3.2.1	Objets	36
3.2.2	Les vecteurs	36
3.2.3	Les matrices	36
3.2.4	Les matrices à plus de deux dimensions	37
3.2.5	Les listes	38
3.2.6	Les structures de données	38
3.3	Quelques fonctions usuelles	39
3.3.1	Quelques fonctions de statistique exploratoire	40
3.4	Construction d'une nouvelle fonction	40
3.4.1	Quelques éléments de programmation	41
3.5	Les entrées/Sorties et gestion des objets créés	42
3.5.1	Les entrées/Sorties	42
3.5.2	gestion des objets créés	42
3.6	Graphisme avec R	42
3.6.1	Les principales commandes graphiques	43
3.6.2	Les options	44
3.7	Exercices de Travaux Pratiques sous R	44

Introduction

Nous présentons, dans ce cours, trois logiciels scientifiques : **Logiciel Latex**, **logiciel Scilab** et **logiciel R**. Ces trois logiciels sont libres et il y a de nombreuses versions et de la documentation sur l'internet.

Le premier, **logiciel Latex**, est un logiciel de traitement de textes scientifiques. Il permet aux mathématiciens, aux Physiciens, aux chimistes, etc ..., de rédiger leurs rapports scientifiques sans trop d'effort pour la représentation des formules mathématiques. En général, la représentation des symboles scientifiques sous Word n'est pas toujours simple et elle n'est pas toujours jolie à voir. Pour permettre aux scientifiques de pouvoir rédiger leurs textes (rapports de recherche, articles, thèses, devoirs, interrogations, livres, etc...), il a été mis à leur disposition le **logiciel Latex**. C'est un logiciel dont la compréhension est facile. L'environnement du Latex est facile. Dans le premier chapitre de ce cours, nous expliquons l'essentiel à connaître pour démarrer avec Latex.

Les deux autres logiciels sont des logiciels de calcul scientifique. Ils sont utilisés par les mathématiciens appliqués (tels que les Numériciens, les statisticiens, etc...), les physiciens, les chimistes, etc ..., dans leur simulation numérique. Ainsi, le **logiciel Scilab** (Scientific Laboratory) est semblable au logiciel Matlab (Matrix laboratory) qui n'est pas libre. Il permet l'analyse et la résolution des problèmes mathématiques en utilisant, en générale, des tableaux. Quant au **logiciel R**, il est plus adaptée à l'analyse et à la résolution des problèmes probabilistes et Statistiques.

Dans le chapitre 2, on donne des notions de base du **logiciel Scilab** et dans le chapitre 3, l'essentielle des commandes à connaître pour une simulation sous le **logiciel R** y est énuméré.

LOGICIEL DE TRAITEMENT DE TEXTE SCIENTIFIQUE : LATEX 2_ε

1.1 Overview

1.1.1 Édition d'un document

On édite un document Latex comme dans les logiciels de programmation. Un exemple d'édition de document en Latex est ci-dessous :

```
\documentclass[french,12pt]{article}

\usepackage[T1]{fontenc}
\usepackage{babel}

\begin{document}
bonjour, Monde!
\end{document}
```

Le texte du document doit-être écrire entre `\begin{document}` et `\end{document}`. Puis, on le sauve sous un nom terminant par `.tex`.

Exemple 1 `premier.tex`

1.1.2 Compilation et visualisation d'un document tex

Pour la compilation (avec WinEdit), on clique sur le menu **LATEX** ou le menu en forme la tête de chat de couleur verte situé dans la barre de menu en haut de l'éditeur WinEdit. On peut également faire la compilation à l'aide du clavier en tapant **Shift+Ctrl+L** pour le menu **LATEX** et **Shift+Ctrl+X** pour le menu en forme de tête de chat.

N.B.: 1 En cliquant sur **LATEX**, on a une simple compilation qui vérifie les éventuelles erreurs ; mais il ne permet pas la visualisation du document. Cependant, avec le menu en forme de tête de chat de couleur verte, on a en plus de la compilation et la vérification, la visualisation du document s'il n'y a pas d'erreur.

Au cour de la compilation d'un fichier LATEX, plusieurs fichiers seront créés. Pour un fichier qui porte le nom premier.tex, on la création de :

premier.log : C'est la transcription détaillée de tout ce qui s'est passé à la compilation. En particulier des éventuelles erreurs.

premier.aux : C'est un fichier auxiliaire (parmi plusieurs autre pour certains documents plus complexes) très précieux en LATEX.

premier.dvi : C'est le résultat de la compilation. Celui que tu visualiseras.

Pour visualiser votre document Latex, après compilation sans visualisation, il vous suffit de clique sur le menu **en forme de loupe juste en bas du menu LATEX**. Votre document aura l'extention .dvi (Exemple : **premier.dvi**) Vous pouvez également visualiser votre document en fichier **PDF** ou en fichier **PS** ou **EPS** en cliquant sur les autres icônes qui suivent les icônes en forme de chat et LATEX.

1.1.3 Structuration du document

Avec LATEX 2_ε, des nouvelles notions apparaissent comme la notion de classe, d'option, de package...

- **Les classes** : Il s'agit des termes **report**, **article**, **book** et **letter** qui définissent les types de document que nous voulons avoir.
- **Les options** : Les options sont placées entre crochets juste après le nom de la commande ; Exemple :
 - ★ `\documentclass[12pt]{report}` permet de charger la classe de document **report** en 12 points.
 - ★ `\documentclass[12pt]{report}[2012/09/03]` permet de demander à LATEX 2_ε d'utiliser le style **report** avec l'option **12pt** dans une version datant d'après le 03 septembre 2012. On aura un message lors de compilation si une version plus ancienne est trouvée.
- **Les packages** : Ce sont les macros de fichiers que nous souhaitons utiliser ; comme exemple : pour **epsfig.sty** ou **fancyhdr.sty**, on place `\usepackage{epsfig}` et `\usepackage{fancyhdr}` entre le `\documentclass` et le `\begin{document}`.

N.B.: 2 Pour éviter la répétition de la commande `\usepackage`, il est possible de donner une liste de packages à charger, en séparant leurs noms par des virgules.

Exemple : `\usepackage[german,english]{babel}`

1.1.4 Structure de documents types

En-tête du document

```
\documentclass[french,twoside,openright]{report}
\usepackage[T1]{fontenc}
\usepackage{babel,indentfirst}
\begin{document}
```

- **fontenc** : permettre la écriture convenable de toute langue ayant des caractères accentués.
- **babel** : se charge de faire des traductions utiles (exemple "chapter" en "chapitre")
- **indentfirst** : pour indenter le premier paragraphe suivant un titre
- **french** : Pour pouvoir passer à tous les packages qui en auront besoin. C'est-à-dire les packages qui sont susceptible de produire du texte.
- **twoside** : demande à ce que l'on travaille en recto-verso.
- **openright** : indique que les chapitres doivent commencer sur des pages impaires (de droite)

Le titre

```
\title{rapport bidon}
\author{Benjamin \textsc{Bayart}}
\and \textsc{M \^eme}
\and Moi \textsc{Aussi}
\and Personne \textsc{d'Aussi}
\date{Le \today}
\maketitle
```

Le debut

```
\tableofcontents
\chapter*{Introduction}
\addcontentsline{toc}{chapter}{Introduction}
\markboth{\uppercase{Introduction}}
{\uppercase{Introduction}}
```

```

\ part{Étude préliminaire}
\ chapter{on a commencé}
\ section{Le commencement}
\ subsection{Une sous section}
\ paragraph{Un paragraphe}

```

La fin du rapport

```

\ chapter*{Conclusion}

\ end{document}

```

1.2 Formules Mathématiques

Pour écrire une formule (symbole) ou une équation mathématique, nous avons besoin d'un environnement mathématique.

1.2.1 Les environnements

Il y a 6 méthodes pour passer en mode maths :

- L'environnement **displaymath**,
- L'environnement **math**,
- $\left[\dots \right]$, équivalent à **displaymath**,
- $\left(\dots \right)$, équivalent à **math**,
- $\$ \$ \dots \$ \$$, équivalent à **displaymath**,
- $\$ \dots \$$, équivalent à **math**.

N.B.: 3

- "**displaymath**" est prévu pour faire apparaître une équation seule centrée sur une ligne ;
- "**math**" est prévu pour mettre une petite équation dans le texte.

1.2.2 Symboles mathématiques

Package latexsym

• Lettres Grecques :

<code>\alpha</code>	α	<code>\beta</code>	β	<code>\gamma</code>	γ	<code>\delta</code>	δ
<code>\epsilon</code>	ϵ	<code>\varepsilon</code>	ε	<code>\zeta</code>	ζ	<code>\eta</code>	η
<code>\theta</code>	θ	<code>\vartheta</code>	ϑ	<code>\iota</code>	ι	<code>\kappa</code>	κ
<code>\lambda</code>	λ	<code>\mu</code>	μ	<code>\nu</code>	ν	<code>\xi</code>	ξ
<code>\pi</code>	π	<code>\varphi</code>	φ	<code>\rho</code>	ρ	<code>\sigma</code>	σ
<code>\varsigma</code>	ς	<code>\tau</code>	τ	<code>\upsilon</code>	υ	<code>\phi</code>	ϕ
<code>\varphi</code>	φ	<code>\chi</code>	χ	<code>\psi</code>	ψ	<code>\omega</code>	ω
<code>\Gamma</code>	Γ	<code>\Delta</code>	Δ	<code>\Theta</code>	Θ	<code>\Lambda</code>	Λ
<code>\Xi</code>	Ξ	<code>\Pi</code>	Π	<code>\Sigma</code>	Σ	<code>\Upsilon</code>	Υ
<code>\Phi</code>	Φ	<code>\Psi</code>	Ψ	<code>\Omega</code>	Ω		

• Opérateurs binaires :

<code>\pm</code>	\pm	<code>\mp</code>	\mp	<code>\times</code>	\times	<code>\div</code>	\div
<code>*</code>	$*$	<code>\star</code>	\star	<code>\circ</code>	\circ	<code>\bullet</code>	\bullet
<code>\cdot</code>	\cdot	<code>\cap</code>	\cap	<code>\cup</code>	\cup	<code>\uplus</code>	\uplus
<code>\oplus</code>	\oplus	<code>\ominus</code>	\ominus	<code>\otimes</code>	\otimes	<code>\oslash</code>	\oslash
<code>\birc</code>	\birc	<code>\dagger</code>	\dagger	<code>\ddagger</code>	\ddagger	<code>\amalg</code>	\amalg
<code>\sqcap</code>	\sqcap	<code>\sqcup</code>	\sqcup	<code>\bigtriangleup</code>	\bigtriangleup		
<code>\vee</code>	\vee	<code>\wedge</code>	\wedge	<code>\bigtriangledown</code>	\bigtriangledown		
<code>\setminus</code>	\setminus	<code>\wr</code>	\wr	<code>\triangleleft</code>	\triangleleft		
<code>\diamond</code>	\diamond	<code>\odot</code>	\odot	<code>\triangleright</code>	\triangleright		

• Symboles de relation

<code>\leq</code>	\leq	<code>\leq</code>	\leq	<code>\geq</code>	\geq		
<code>\equiv</code>	\equiv	<code>\models</code>	\models	<code>\prec</code>	\prec	<code>\succ</code>	\succ
<code>\sim</code>	\sim	<code>\perp</code>	\perp	<code>\preceq</code>	\preceq	<code>\succeq</code>	\succeq
<code>\simeq</code>	\simeq	<code>\mid</code>	\mid	<code>\ll</code>	\ll	<code>\gg</code>	\gg
<code>\subset</code>	\subset	<code>\subset</code>	\subset	<code>\subseteq</code>	\subseteq	<code>\supseteq</code>	\supseteq
<code>\sqsubset</code>	\sqsubset	<code>\sqsupset</code>	\sqsupset	<code>\sqsubseteq</code>	\sqsubseteq	<code>\sqsupseteq</code>	\sqsupseteq
<code>\parallel</code>	\parallel	<code>\approx</code>	\approx	<code>\bowtie</code>	\bowtie	<code>\Join</code>	\Join
<code>\neq</code>	\neq	<code>\doteq</code>	\doteq	<code>\cong</code>	\cong	<code>\approx</code>	\approx
<code>\in</code>	\in	<code>\ni</code>	\ni	<code>\smile</code>	\smile	<code>\frown</code>	\frown
<code>\vdash</code>	\vdash	<code>\dashv</code>	\dashv			<code>\asymp</code>	\asymp

• Frèches

<code>\leftrightharrow</code>	\leftrightarrow	<code>\leftharpoondown</code>	\swarrow	<code>\gamma</code>	γ
<code>\longleftarrow</code>	\longleftarrow	<code>\hookleftarrow</code>	\hookleftarrow	<code>\zeta</code>	ζ
<code>\Leftrightarrow</code>	\Leftrightarrow	<code>\leftarrow</code>	\leftarrow	<code>\iota</code>	ι
<code>\Longleftarrow</code>	\Longleftarrow	<code>\uparrow</code>	\uparrow	<code>\nu</code>	ν
<code>\longrightarrow</code>	\longrightarrow	<code>\Leftarrow</code>	\Leftarrow	<code>\rho</code>	ρ
<code>\Longrightarrow</code>	\Longrightarrow	<code>\Uparrow</code>	\Uparrow	<code>\upsilon</code>	υ
<code>\Uparrow</code>	\Uparrow	<code>\mapsto</code>	\mapsto	<code>\psi</code>	ψ
<code>\Downarrow</code>	\Downarrow	<code>\nearrow</code>	\nearrow	<code>\Theta</code>	Θ
<code>\leftharpoonup</code>	\leftharpoonup	<code>\nwarrow</code>	\nwarrow	<code>\Sigma</code>	Σ
		<code>\updownarrow</code>	\updownarrow	<code>\Omega</code>	Ω

• Constructions mathématiques

<code>\widetilde{abc}</code>	\widetilde{abc}	<code>\widehat{abc}</code>	\widehat{abc}
<code>\overleftarrow{abc}</code>	\overleftarrow{abc}	<code>\overrightarrow{abc}</code>	\overrightarrow{abc}
<code>\overline{abc}</code>	\overline{abc}	<code>\underline{abc}</code>	\underline{abc}
<code>\overbrace{abc}</code>	\overbrace{abc}	<code>\underbrace{abc}</code>	\underbrace{abc}
<code>\sqrt{abc}</code>	\sqrt{abc}	<code>\sqrt[n]{abc}</code>	$\sqrt[n]{abc}$
<code>f'</code>	f'	<code>\frac{abc}{xyz}</code>	$\frac{abc}{xyz}$

1.2.3 constructions mathématiques

Quelques exemples de constructions mathématiques

Voyons des exemples de constructions mathématiques.

- Sommes

Pour écrire

$$\sum_{i=0}^n u_i$$

on tape : `\[\sum_{i=0}^n u_i]`

D'autre part, si on souhaite taper $\sum_{i=0}^n u_i$ en plein dans le texte, on fait appel à la commande

`\displaystyle` en tapant :

$$\left(\displaystyle \sum_{i=0}^n u_i \right)$$

Une autre commande est donnée par `\textstyle` : équation dans le texte $\sum_{i=0}^n u_i$

- Opérations et fonctions

Cas «Particulier» : la limite

$$\lim_{n \rightarrow +\infty} u_n = \ell$$

s'obtient en tapant

$$\lim_{n \rightarrow +\infty} u_n = \ell$$

- Fraction, racines et accolades

$$\sum_{n=0}^{+\infty} \frac{x^n}{n!} = e^x = \sqrt{e^{2x}}$$

$$\sum_{i=0}^n u_i = \underbrace{n_0 + u_1 + \cdots + u_n}_{n+1 \text{ termes}}$$

s'obtient à partir de

$$\begin{aligned} & \left[\sum_{n=0}^{+\infty} \frac{x^n}{n!} \right] \\ &= e^x = \sqrt{e^{2x}} \\ & \left[\sum_{i=0}^n u_i = \underbrace{n_0 + u_1 + \cdots + u_n}_{n+1 \text{ termes}} \right] \end{aligned}$$

- On aime avoir de grandes parenthèses en Mathématiques. On procède de la manière suivante :

Délimiteurs

Pour avoir

$$\left(\sum_{i=0}^n u_i \right)$$

on tape

$$\left(\sum_{i=0}^n u_i \right)$$

N.B. : Le délimiteur vide est utilisé pour les systèmes d'équations (une accolade à gauche et rien à droite), comme dans l'exemple suivant :

$$\left\{ \text{system} \right.$$

- Les matrices

Il est très facile d'écrire une matrice.

Exemple :

```

\ [A = \left(
\ begin{array}{ccccc}
1 & 2 & 3 & \cdots & n \\
2 & 3 & 4 & \cdots & n+1 \\
3 & 4 & 5 & \cdots & n+2 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
n & n+1 & n+2 & \cdots & 2n+1 \\
\end{array}
\right)^{2}]

```

produira :

$$A = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ 2 & 3 & 4 & \cdots & n+1 \\ 3 & 4 & 5 & \cdots & n+2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ n & n+1 & n+2 & \cdots & 2n+1 \end{pmatrix}^2$$

Le 'c' dans l'argument du 'array' indique que l'on souhaite une colonne centrée. On peut préférer indiquer 'l' pour une colonne gauche ou 'r' pour une colonne droite.

1.3 Images

1.3.1 Dessins à inclure

Pour inclure un fichier produit par certains logiciels comme **Xfig** et **gnuplot**, on utilisera les commandes prévues par LATEX en standard :

Inclusion de code LATEX

```

\ begin{figure}
\ input nom_du_fichier
\ caption{Titre de la figure}
\ end{figure}

```

1.3.2 Dessins à inclure

Inclure du PostScript (graphics)

Pour l'inclusion des graphics, on prendra soin d'inclure le package comme cela

```
\usepackage[dvips]{graphics}
```

Puis la commande

```
\includegraphics{fichier.ps}
```

viendra placer à l'endroit courant le dessin contenu dans le fichier placé en paramètre.

1.4 Bibliographie, Index

1.4.1 Approche manuelle

Elle repose sur la commande `\bibitem` et l'environnement `thebibliography`
Code Latex

Exemple 2

```
\begin{thebibliography}{99}
\ bibitem{thesedos} M. Dosso,
\ textit{Sur quelques algorithmes d'analyse de stabilité forte
de matrices symplectiques}, PHD Thesis (September 2006).
\ bibitem{dos-sad} M. Dosso, M. Sadkane,
\ textit{On the strongly stable of symplectic matrices,
Num. Lin. Alg. Appl. ( first published online :13 Dec. 2011 )}.
\end{thebibliography}
```

donnera :

Bibliographie

- [1]M. Dosso, *Sur quelques algorithmes d'analyse de stabilité forte de matrices symplectiques, PHD Thesis (September 2006).*
- [2] M. Dosso, M. Sadkane, *On the strongly stable of symplectic matrices, Num. Lin. Alg. Appl. (first published online :13 Dec. 2011).*

En générale, on utilise la bibliographie pour les références. Pour ce faire, dans le cadre de ce exemple, on a les commandes suivantes

```
\ cite{thesedos}
\ cite{dos-sad}
\ cite{thesedos, dos-sad}
```

D'autre part, on peut mettre les trois premières lettres et du nom de l'auteur et l'année de parution comme référence, on procède comme suite :

Code Latex

```
\begin{thebibliography}{WWW99}
\bibitem[DOS06]{thesedos} M. Dosso,
\textit{Sur quelques algorithmes d'analyse de stabilité forte
de matrices symplectiques}, PHD Thesis (September 2006).
\bibitem[DOSS-SAD11]{dos-sad} M. Dosso, M. Sadkane,
\textit{On the strongly stable of symplectic matrices},
Num. Lin. Alg. Appl. ( first published online :13 Dec. 2011 ).
\end{thebibliography}
```

1.4.2 Les commandes LATEX

Retenons la commandes suivante permettant de faire une citation :

```
\cite[commentaire]{clef1,clef1,...}
```

faire référence au(x) livre(s) ayant comme clef d'accès **clef1,clef2,...** avec le commentaire donné, par exemple, `\cite[page 30]{texbook}` donnera [58, page 30].

Puis cette autre commande qui indique les fichiers de la base de données bibliographies à exploiter :

```
\bibliography{fichier1,fichier2,...}
```

indique que les données utilisées sont dans les fichiers **fichier1.bib**, **fichier2.bib**,...

1.5 Exercices de Travaux Pratiques

Exercice 1 (diverses classes)

1. Créer des documents de classes respectivement *book* et *report* en introduisant des chapitres et sections.
2. Observer, avec *book*, les entêtes de pages.
3. Obtenir des titres courants. Pour cela, on pourra utiliser l'un des packages *fancyhdr* ou *titlesec* dont on parcourra la documentation (on pourra également s'aider de la FAD francophone)

Exercice 2 (*alignement-Tableaux.tex*)

Sachant qu'on peut remplacer c, l, r par $p\{ncm\}$ pour créer une colonne de n cm de large, obtenez un tableau ayant l'allure suivant :

$l(left)$	aligné à gauche
$r(right)$	aligné à droite
$c(center)$	centré
$p\{ncm\}$ (justifié)	justifie le texte dans la colonne de largeur fixée à n cm

Exercice 3

1. Créer un document contenant, sur deux pages différentes, deux tableaux, chacun dans un environnement `table`, de façon à ce qu'ils aient un titre et qu'ils soient numérotés.
2. Ajouter du texte faisant référence à ces «tables».
3. Construire une liste des tables.

Exercice 4 (*Mathématiques.....maths.tex*)

En utilisant le package `amsmath` reproduire le texte suivant, en proposant, le cas échéant, des macros personnelles appropriées :

1. Soit f une fonction définie sur l'intervalle $[-1, 1]$.
2. La plus belle égalité mathématique est, sans contexte :

$$e^{i\pi} + 1 = 0$$
mais certains préféreront l'écrire ainsi :

$$e^{i\pi} + 1 = 0$$
3. Vaut-il mieux écrire «L'ensemble des réels est noté \mathbb{R} .» ou «L'ensemble des réels est noté \mathbb{R} .» ?
4. Étudier et représenter graphiquement $f : x \mapsto \frac{2}{5}\sqrt{25 - x^2}$, soit

$$f : x \mapsto \frac{2}{5}\sqrt{25 - x^2}$$
5. On pose $A = \int_a^b f(t)dx$, soit

$$A = \int_a^b f(t)dx$$
6. On pose $A = S_n = 1 + \frac{1}{2} + \dots + \frac{1}{n} - \ln n$.
7. Démontrer la formule :

$$\tan(a - b) = \frac{\tan a - \tan b}{1 + \tan a \tan b}$$
8. Calculer :

$$K = \int_a^{\frac{\pi}{2}} e^t \sin t dt$$
9. On connaît la formule de Moivre :

$$\forall \theta \in \mathbb{R}, \forall n \in \mathbb{Z}, (\cos \theta + i \sin \theta)^n = \cos n\theta + i \sin n\theta$$

10. Pour tout $n \geq 1$, $\lim_{x \rightarrow +\infty} \frac{\ln x}{x^n} = 0$, soit
 $\lim_{x \rightarrow +\infty} \frac{\ln x}{x^n} = 0$
11. Soit $f(t) = \sin(\pi t^2)$
 (a) Montrer que, pour $t \in [0, 1]$, $|f'(t)| \leq 2\pi$.
 (b) En déduire une valeur approchée de $I = \int_0^1 f(t)dt$ à 10^{-3} près.
12. La distance $d(M_0, \mathcal{P})$ de M_0 à \mathcal{P} vérifie :
 $d(M_0, \mathcal{P}) = \frac{|ax_0+by_0+cz_0+d|}{\sqrt{a^2+b^2+c^2}}$
13. On appelle espérance de la loi P le nombre :
 $\mu = \sum_{i=1}^r p_i x_i$
14. Pour $1 \leq k \leq n$: $P(A_k \cap B) = P(A_k) \times P - A_k(B)$.
15. Le nombre de sous-ensembles à p éléments dans un ensemble qui en compte n est :

$$\binom{n}{p} = \frac{n!}{p!(n-p)!}$$

16. $\vec{u} \cdot \vec{v} = \|\vec{u}\| \|\vec{v}\| \cos(\widehat{\vec{u}, \vec{v}})$

17. Si $\vec{n} \neq \vec{0}$ alors

$$M \in \mathcal{D} \Leftrightarrow \overrightarrow{AM} \cdot \vec{n} = 0$$

Exercice 5 (Formules alignées).....maths-alignement.tex)

En s'aidant de la documentation du package `amsmath`, reproduire le texte suivant :

Pour tout $x \not\equiv \frac{3\pi}{2} [2\pi]$, on a

$$\begin{aligned} (1 + \sin x) \tan^2 x &= \frac{(1 + \sin x) \sin^2 x}{\cos^2 x} \\ &= \frac{(1 + \sin x) \sin^2 x}{1 - \sin^2 x} \\ &= \frac{(1 + \sin x) \sin^2 x}{(1 + \sin x)(1 - \sin x)} \\ &= \frac{\sin^2 x}{1 - \sin x} \end{aligned}$$

Exercice 6 (Approfondissements.....maths-approfondissements.tex)

En s'aidant éventuellement des fichiers de documentation

- du package `amsmath`
- du package `mathtools`
- du package `amsthm` (ou, mieux, du package `ntheorem`)

reproduire le texte suivant, en cas échéant, des macros personnelles appropriées :

1. On a :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2. Montrer que pour tout n entier naturel, on a :

$$\begin{cases} V_{n+1} = 0,8V_n + 0,2R_n \\ R_{n+1} = 0,1V_n + 0,8R_n. \end{cases}$$

3. Soit Ω un point du plan d'affixe ω et θ un réel. La rotation de centre Ω et d'angle θ associe, au point $M(z)$, le point $M'(z')$ tel que $z' - \omega = e^{i\theta}(z - \omega)$.

4. **Problème 1** Créer un problème, automatiquement numéroté $\langle\langle 1 \rangle\rangle$.

Problème 2 Créer 2^e problème, automatiquement numéroté $\langle\langle 2 \rangle\rangle$.

Problème 3 Créer un 3^e problème, automatiquement numéroté «3».

Problème 4 (long !) Résoudre tous les problèmes, depuis le problème 1 page ?? jusqu'au présent problème.

5. Dans la définition suivante, on prendra soin de faire figurer

- le symbole ε et non pas ϵ ;
- le symbole \leq et non pas \preceq ;
- un symbole «implique» de la bonne longueur ;
- une espace suffisant après la virgule.

Définition 1 Posons $S_n = \sum_{k=1}^n k$. Alors on :

$$S_n = \frac{n(n+1)}{2}$$

6. **Proposition 1** Posons $S_n = \sum_{k=1}^n k$. Alors on

$$S_n = \frac{n(n+1)}{2} \quad (1.1)$$

Démonstration 1 Par définition,

$$S_n = 1 + 2 + \cdots + (n-1) + n$$

Alors

$$\begin{aligned} 2S_n &= 1 + 2 + \cdots + (n-1) + n \\ &= n + (n-1) + \cdots + 2 + 1 \\ &= (1+n) + (2+n-1) + \cdots + (n-1+2) + (n+1) \\ &= \underbrace{(n+1) + \cdots + (n+1)}_{n \text{ fois}} \\ 2S_n &= n(n+1) \end{aligned} \quad (1.2)$$

Ce qui, par multiplication par $\frac{1}{2}$, prouve l'égalité (1.1)

7. On peut écrire la ligne (1.2) de façon plus élégante, ainsi :

$$\begin{aligned} 2S_n &= 1 + 2 + \cdots + (n-1) + n \\ &\quad + n + (n-1) + \cdots + 2 + 1 \end{aligned} \quad (1.3)$$

8. Voici un environnement de preuve plus élégant (non numéroté, avec un symbole indiquant où se trouve la fin de la preuve)

PREUVE par définition,

$$S_n = 1 + 2 + \cdots + (n-1) + n$$

Alors

$$\begin{aligned} 2S_n &= 1 + 2 + \cdots + (n-1) + n \\ &= n + (n-1) + \cdots + 2 + 1 \\ &= (1+n) + (2+n-1) + \cdots + (n-1+2) + (n+1) \\ &= \underbrace{(n+1) + \cdots + (n+1)}_{n \text{ fois}} \\ 2S_n &= n(n+1) \end{aligned}$$

ce qui, par multiplication par $\frac{1}{2}$, prouve l'égalité (1.1).

9. Il est bon de connaître la formule de Poincaré qui est tellement longue qu'elle ne tient pas sur une ligne :

$$\begin{aligned} \left| \bigcup_{i=1}^n A_i \right| &= \sum_{i=1}^n |A_i| - \sum_{(i,j) \in \mathbb{N}^2} |A_i \cap A_j| \\ &\quad + \sum_{(i,j,k) \in \mathbb{N}^3, 1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| - \cdots + (-1)^{n+1} |A_1 \cap \cdots \cap A_n| \end{aligned} \quad (1.4)$$

mais heureusement, il en existe une forme plus condensée :

$$\left| \bigcup_{i=1}^n A_i \right| = \sum_{k=1}^n (-1)^{k-1} \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq n} |A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k}|.$$

Exercice 7 Fusionner verticalement et horizontalement

Quadriques		
propre	à centre	ellipsoïde $x^2 + y^2 + z^2 = 1$
		hyperboloïde $x^2 + y^2 + z^2 = 1 \quad x^2 - y^2 - z^2 = 1$
	paraboloïde	elliptique $x^2 + y^2 = z$
		hyperbolique $x^2 - y^2 = z$
impropre	cône $x^2 - y^2 - z^2 = 0$	
	cylindre $x^2 + y^2 = 1$	
	deux plans sécants $x^2 + y^2 = 0$	
	deux plans parallèles $x^2 = 1$	
	deux plans confondus $x^2 = 0$	

Le tableau ci-dessus possède trois colonnes. Placer un texte verticalement dans un tableau ne devrait plus poser de problème. En revanche, l'entrée «impropre» en bas à gauche du tableau est plus compliquée puisqu'elle fusionne à la fois des lignes et des colonnes. Comment obtenir ce tableau ?

Exercice 8 Numérotation automatique des lignes

Le package `xcolor` avec l'option `table` permet de colorier une ligne sur deux de façon différente. En interne, il utilise le compteur `\rownum`. Cet exercice propose d'employer ce compteur pour réaliser une numérotation automatique des lignes.

Comment obtenir le résultat suivant, sans taper directement le numéro des lignes, mais en se servant d'une macro `\numline` qui fera automatiquement le travail ? la difficulté est que la numérotation ne doit pas débiter immédiatement et on prévoira un compteur `firstline` indiquant la première ligne à numérotter :

Satellites d'Uranus				
	Nom	Période(j)	Diamètre	Magnitude
1	Miranda	1,413	300	16,5
2	Ariel	2,520	800	14,4
3	Umbriel	4,144	600	15,3
4	Titania	8,706	1000	14,0
5	Obéron	13,463	1000	14,2

Exercice 9 Table des matières en couleurs

Supposons que l'on réalise un document de classe **book**. Comment faire pour obtenir une table des matières composée entièrement en vert, alors que le reste du document sera composé normalement en noir ?

Table des matières

I Le nombre et la grandeur	3
1 Sur la nature du raisonnement mathématique	5
2 La grandeur mathématique et l'expérience	7

Le premier titre provient d'une commande `\part` et les deux suivants de de commandes `\chapter`. Pour les plus curieux de nos lecteurs, il s'agit du début de la Science et l'Hypothèse de Poincaré.

LOGICIEL SCILAB

Scilab est un logiciel de calcul numérique développé par l'Institut National de Recherche en Informatique et en Automatique (INRIA) et distribué gratuitement sur presque tout type d'ordinateur (PC Windows, Linux, Unix, Macintosh). Pour plus d'information et pour télécharger ce logiciel, vous pouvez consulter le site Internet de INRIA :

<http://www.scilab.org>.

Depuis la version 2.7, Scilab propose un éditeur intégré permettant d'écrire les scripts sans sortir de l'environnement.

N.B. : On peut dans un premier temps utiliser Scilab comme une calculatrice matricielle.

2.1 Fonctionnement général

Utilisation de l'aide en ligne :

Une aide en ligne sur toutes les commandes SCILAB (les fonctions préprogrammées, les types, les types de données, etc) est disponible en cliquant sur le bouton **help** puis sur les différents sous-menus. On peut aussi taper les demandes d'aides au clavier, par exemple :

- **help()** : donne des informations générales sur l'aide en ligne
- **apropos motclef** : donne toutes les fonctions scilab qui ont un rapport avec le mot-clef spécifié
- **help nomdefonction** : ouvre la fenêtre d'aide pour utiliser la fonction Scilab nomdefonction

2.2 Type de données

- **Constantes spéciales :**

Elles sont précédées du caractère %. Voici la liste des plus utilisées. La liste complète est obtenue en tapant la commande **who**

`%F %T %s %nan %inf %t %f %eps %i %e %pi`

- **Vecteurs** Pour définir un vecteur, la syntaxe est comme suite :

```

-- > v = [2, -3 + %i, 7] // vecteur ligne
! 2. - 3. + i 7.!
-- > v' // vecteur transposé conjugué
ans=
! 2. !
! -3. - i !
! 7. !
-- > v.' // vecteur transposé
ans=
! 2. !
! -3. + i !
! 7. !

```

- vecteurs

```

-- > w = [-3; -3 - %i; 2] //vecteur colonne
w=
! 2. !
! -3. - i !
! 7. !
-- > v' + w // somme de deux vecteurs
ans =
! -1. !
! -6. - 2.i !
! 9. !
-- > v + w //produit scalaire euclidien
ans =
18.
-- > w'.* v //produit des composantes ( on peut aussi essayer avec ./)// ans=
! -6. 8. + 6.i 14.!

```

1. **Matrices** Les matrices suivent la même syntaxe que les vecteurs. Les composantes des lignes sont séparées par des virgules et chaque ligne est séparée de l'autre par un point virgule.

```

-- > // une manière de définir une matrice 3 × 3 :
-- > A = [1, 2, 3; 0, 0, atan(1); 5, 9, -1];
-- > // une autre syntaxe pour faire la même chose
-- > A = [1 2 3
          0 0 atan(1)
          5 9 -1]

```

2.2.1 Principales opérations sur les matrices.

Fonction	Description
<i>ones(i, j)</i>	matrice remplies de 1
<i>zeros</i>	matrice nulle
<i>eye(i, j)</i>	matrice identité
<i>toeplitz(u)</i>	crée une matrices de Toeplitz
<i>diag(u)</i>	matrice diagonale
<i>diag(U)</i>	extraire la diagonale de U
<i>triu(A)</i>	matrice triangulaire supérieure
<i>tril(A)</i>	matrice triangulaire inférieure
<i>linspace(a, b, n)</i>	vecteur de n composante
<i>linsolve(A, b)</i>	résoudre $Au + b = 0$
$A \setminus b$	résoudre $Au = b$
<i>cond(A)</i>	conditionnement d'une matrice
<i>det(A)</i>	déterminant d'une matrice
<i>rank(A)</i>	rang d'une matrice
<i>inv(A)</i>	inverse d'une matrice
<i>pinv(A)</i>	pseudo inverse d'une matrice
<i>svd(A)</i>	valeurs singulière d'une matrice
<i>norm(A)</i>	norme matricielle ou vectorielle
u'	transposé conjugué de u
$u.'$	transposé conjugué de u
$u * v$	multiplication matricielle
$u + v$	addition matricielle
$u - v$	soustraction matricielle
$u.*v$	multiplication point par point
$u./v$	division point par point

2.2.2 Autres objets mathématiques

Un grand nombre d'objets mathématiques ou logiques interviennent dans un programme de simulation (entier, réel, fonction, polynôme, matrice, booléen, ets...)

Instruction	Description
$a = 2$	affectation de 2 à a
$a = \%t$	affectation du booléen vrai à a
$p = poly([\%i \ - \%i], 'z')$	affectation du polynôme
$z = poly(0, 'z');$	
$q = 1 + 3 * z + 4.5 * z \wedge 2$	affectation du polynôme
$r = p/q$	affectation d'une fraction rationnelle à r
$A = [a + 1 \ 2; atan(1) \ -3]$	affectation d'une matrice réelle
$rand(3, 3, 'u')$	matrice aléatoire 3×3 (loi uniforme sur $[0, 1]$
$rand(3, 3, 'n')$	matrice aléatoire 3×3 (loi gaussienne centrée)
$eye(4, 4)$	matrice identité
$ones(1, 2) ; 1 : 2 : 7$	vecteurs lignes (1, 1) et (1, 3, 5, 7)
$B = toeplitz$	matrice triangulaire symétrique
$C = B' * B$	calcul de
$D = C. * B$	multiplication terme à terme
$f = C(1 : 4, 1)$ ou $f = C(:, 1)$	extraction des termes de la première colonne de C
$g = D \setminus f$	division à gauche de f par D
$def f('y = f(a, b)', 'y = a + b')$	définition de la fonction
	$f(a, b) = a + b$

2.3 Programmation

2.3.1 Ediction de scripts

Scilab possède un editeur intégré qu'on obtient en cliquant sur le bouton **Editor**(ou **Editeur** pour la version française). Il gère les fichiers que vous créez :

- S'il s'agit d'une fonction qui se termine par **Endfunction**, Le fichier prend automatiquement le suffixe **sci**.
- Si c'est un script, il prend le suffixe **sce**.

Le bouton **Execute** dans la fenêtre d'édition permet de lancer l'exécution du script courant dans la fenêtre de commandes.

Pour les fonctions, doivent d'abord être chargées (avec la commande **getf**) et utilisées avec la syntaxe d'appel.

2.3.2 Fonctions ou macros ("function" or "macros'...)

De manière générale, la syntaxe de définition d'une fonction externe est :

Visualisation

```
function    [y1, ..., ym] = toto(x1, ..., xn)
..
..
..
endfunction
```

où **toto** est le nom de la fonction, x_1, \dots, x_n , les n arguments d'entrée et $[y_1, \dots, y_m]$ les m arguments de sortie.

On peut créer le fichier "angle.sci" contenant les lignes :

Exemple 3

```
function    [s] = angle(x, y) // calcul de l'angle en degrés
s = 180 * atan(y/x) / %pi; // visualise le triangle
xpoly([x, 0, x], [0, 0, y], "lines")
endfunction
endfunction
```

puis dans la fenêtre de commandes on tape

```
getf("angle.sci", "c");
```

Si une des variables de la procédure n'est pas définie à l'intérieur de celle-ci et non plus en argument d'entrée, elle prend la valeur définie dans le programme appelant. Ceci permet d'appeler les fonctions avec moins de paramètres d'entrée que prévu. Ainsi

Exemple de fonction

```
-- > clear
-- > getf("angle.sci")
-- > angle(4)
! -- error
undefined variable : y
at line    2 of function angle    called by :
angle(4)
```

donne une erreur car le paramètre y n'est pas affecté. En revanche

Exemple de fonction

```
-- > clear
-- > getf("angle.sci")
-- > y = 2
y =
    2.
-- > angle(4)
ans =
    1.1071487
-- >
```

2.3.3 Boucles

Il y a deux type de boucle en Scilab : Les boucles **while** et les boucles **for**. La boucle **for** parcourt un vecteur d'indices et effectue à chaque pas toutes les instructions délimités par l'instruction **end**.

Exemple 4

```
-- > x = 1; for k = 1 : 4, x = x * k, end
```

$x =$	$x =$
1.	6.
$x =$	$x =$
2.	24.

La boucle **for** peut parcourir un vecteur (ou une matrice) en prenant comme valeur à chaque pas les éléments (ou colonnes) successifs.

Exemple 5

```
-- > V = [-1  3  0]
```

```
-- > x = 1; k = v, x = x + k, end
x =
    0
x =
    3.
x =
    3.
```

2.3.4 Boucles (while)

La boucle **while** effectue une suite de commandes tant qu'une condition est satisfaite.

Exemple 6

```
-- > x = 1; while x < 14, x = x + 5, end
x =
    6.
x =
   11.
x =
   16.
```

Ces deux types de boucle peuvent être interrompus par l'instruction **break**.

2.3.5 Tests

Dans **Scilab** on a la structure conditionnelle if-then-else-end agrémenté du **elseif** parfois bien utile. La syntaxe est par exemple :

Exemple 7

```
-- > x = 16
x =
   16.
-- > if x > 0 then, y = -x, else y = x, end
y =
  -16.
```

Dans le cas où le test doit départager un grand nombre (>2) de possibilité,... on peut utiliser le **select-case**, équivalent du **switch-case** en *C*.

Exemple 8

```
-- >n = round(10 * rand(1,1))
-- >select n
-- >case 0 then
-- >  printf('cas numero 0')
-- >case 1 then
-- >  printf('cas numero 1')
-- >else
-- >  printf('autre cas')
-- >end
```

2.3.6 Les opérations logiques dans le tests

français	anglais	test Scilab
et	and	&
ou	or	
non	not	~
égal	equal	==
différent	diferent	<> ou ~=
plus petit que	lower than	<
plus grand que	greater than	>
plus petit ou égal à	lower than ou equal	<=
plus grand ou égal à	larger than or equal	>=

2.3.7 Utilisation de fonctions Scilab

On a les commandes **fsolve** et **ode** qui servent à résoudre des système d'équations non linéaires et équations différentielles ordianaires respectivement.

- Pour **fsolve** : La syntaxe la plus simple est : **x=fsolve(x0,fct)**
et peut être précisée avec des arguments optionnels :
 $[x \ [v \ [inf0]]] = fsolve(x0, fct \ [fjac] \ [tol]).$
x0 : valeur initale de l'argument de la fonction
fct et **fjac** : des fonctions externes

tol : un scalaire réel pour la tolérance sur l'erreur
 x : vecteur contenant la solution du système
 v : vecteur contenant la valeur de la fonction en x
 inf0 : indicateur de fin : (0 : mauvais paramètre ; 1 : calcul correct avec une erreur inférieure à tol ; 2 : nombre maximum d'appels à la fonction atteint. ; etc ...)

- Pour **ode** : la syntaxe d'appel la plus simple est : **y=ode(y0,t0,t,f)** où *y0* est le vecteur des conditions initiales, *t0* : le temps initial, *t* : le vecteur de temps et *y* : la matrice contenant la solution

$y = [y(t(1)), y(t(2)), \dots]$

L'argument *f* de **ode** est une fonction externe (syntaxe d'appel : $\dot{y} = f(t, y)$) où *t* est un scalaire réel (temps) et *y* un vecteur de réels.

Syntaxe d'appel avec des paramètres optionnels :

$[y, w, iw] = \text{ode}([type], y0, t0, t, [rto1, atol], f, [jac])$

$[y, rd, w, iw] = \text{ode}('root', y0, t0, t, [rto1, atol], f, [jac])$

où type (\approx "adams", "stiff", "rk", "discrete", "roots", etc)

rto1, atol : vecteurs de constantes réelles de même taille que *y*.

2.4 Entrées/Sortie sous Scilab

2.4.1 Interaction avec l'utilisateur

Pour communiquer avec l'utilisateur du programme **Scilab**, on utilise la commande **input** dont la syntaxe est :

$$[x] = \text{input}('message', ['string'])$$

"message" : est une chaîne de caractères qui s'affiche dans la fenêtre de commande.

x est la variable où le programme stocke la valeur rentrée par l'utilisateur ;

Pour afficher des résultats avec un format, on utilise la commande

printf("format", variable)

Exemple 9

```

-- >a = 1/3;
-- >printf(" valeur scalire = %f", a);
    valeur scalaire = 0.333333
-- >a = 2;
-- >printf(" valeur entière = %d", a);
    valeur entière = 2
-- >a = "une chaîne de caractères";
-- >printf(" pour afficher %s", a);
    pour afficher une chaîne de caractères
-- >b = 1;
-- >a = "chaîne de caractères";
-- >printf(" pour afficher %d %s", b, a);
    pour afficher 1 chaîne de caractères

```

Des "interfaces utilisations" plus élaborées peuvent être programmées avec les menus. (Voir l'aide en ligne sur les commandes **uimenu**, **addmenu**, etc...)

2.4.2 Sauvegarde de résultats en binaire

bf who : Pour zn avoir la liste
save : Pour en avoir la liste
load : Pour sauver des variables dans un fichier.
clear ; Pour effacer toutes ou une partie des variable.

Exemple 10

```

-- >a = [1 : 2 : 10]           //définition du tableau a
    a =
    ! 1. 3. 5. 7. 9. !
-- > //sauvegarde dans le fichier binaire 'tab_a'
-- >save('tab_a', a)
-- >clear a                   //effacement du tableau a
-- >a                         //a n'existe plus
    ! --error                4
    undefined variable :a

-- > //récupération du tableau a sauvé dans 'tab_a'
-- >load('tab_a')
-- >a
    a =
    ! 1. 3. 5. 7. 9. !

```

2.5 Sorties graphiques

2.5.1 Les tracés en 2 dimensions

`.plot2d` est la commande de base pour tracer une courbe.

`plot2d(x,y,[style,chaîne,legende,rectangulaire])` :

- `x,y` : sont des matrices de tailles $[npt, nc]$ contenant les abscisses et les ordonnées. nc est le nombre de courbes et npt le nombre de points-le même,pour toutes les courbes.
- `style` est un vecteur de dimension $(1, nc)$.
- `chaîne` est une chaîne de trois caractères "xyz".

Ce premier exemple calcule automatiquement le repère

Exemple 11

```

x = [0 : 1 : 100]
y = sin(2 * %pi * /100;

```

```
z = 3 * y;
plot2d(x, y, [1], '121', y(x)');
```

Dans un autre exemple, le rère est imposé par les valeurss de cadre :

Exemple 12 *xbasc()*

```
cadre = [min(x), min(y), max(x), max(y)];
plot2d(x, y, [1], '111', y(x)', cadre);
```

2.5.2 Graphiques dans l'espace

La commande **plot3d** représente graphiquement une matrice à trois dimensions dont l'appel la plus simple est `plot3d(x,y,z)` où x , y et z sont trois matrices et z contient les valeurs des points de coordonnées (x, y) .

- **param3d** pour représenter des courbes paramétrées en trois dimensions.
- **contour** pour les courbes d'une fonction 3d donnée par une matrice.
- **fec** pour les courbes de niveaux d'une fonction donnée par ses valeurs nodales sur un maillage triangulaire.

Instruction	Description
<code>plot2d(x,y)</code>	tracé de la courbe passant par les points (x, y)
<code>plot2d1('ol1',x,y)</code>	idem avec echelle logarithmique sur les deux axes
<code>fplot2d(x,f)</code>	tracé de la courbe $(x, f(x))$
<code>champ(x,y,fx,fy)</code>	tracé du champ de vecteurs $(fx(x, y), fy(x, y))$
<code>plot3d(x,y,z)</code>	tracé de la surface passant par les points (x, y, z)
<code>param3d(x,y,z)</code>	tracé de la courbe paramétré passant en (x, y, z)
<code>contour(x,y,z,n)</code>	tracé de n courbes de niveau d'2 surface
<code>histplot(n,data)</code>	histogramme de l'échatillon data divisé en n classes
<code>xbasc()</code>	effacement des fenêtres graphiques
<code>xset()</code>	modification des options graphiques
<code>xsetech([x1,y1,x2,y2])</code>	découpage d'une fenêtre graphiques
<code>xstring(x,y,'coucou')</code>	inscription de caractères sur une figure

2.6 Exercices de Travaux Pratiques

Exercice 1 (Echange des valeurs de deux variables)

Etant données deux variables a et b (dont les valeurs sont déjà entrée dans la machine), écrire une ligne de commande (utilisant uniquement l'affectation $=$) qui échange les valeurs de a et b .

Exercice 2

1- Définir la matrice d'ordre n suivant (voir le détail de la fonction **diag** à l'aide du **Help**) :

$$A = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$$

2- Soit A une matrice carrée; que vaut **diag(diag(A))** ?

3- Les fonctions **tril** (rep. **triu** permet d'extraire la partie triangulaire inférieure (resp. supérieure) d'une matrice. Définir une matrice carrée A quelconque (par exemple avec **rand** et construire en une seule instruction, une matrice triangulaire T telle que $t_{ij} = a_{ij}$ pour $i > j$ (les parties strictement triangulaires inférieures de A et T sont égales) et telle que $t_{ii} = 1$ (T est à diagonale unité).

4- Soit X une matrice (ou vecteur...) que l'on a définie dans l'environnement. Écrire l'instruction qui permet de calculer la matrice Y (de même taille que X) dont l'élément en position (i, j) est égal à $f(X_{ij})$ dans les cas suivants :

(a) $f(x) = 2x^2 - 3x + 1$

(b) $f(x) = |2x^2 - 3x + 1|$

(c) $f(x) = (X - 1)(x + 4)$

(d) $f(x) = \frac{1}{1+x^2}$

5- Tracer le graphe de la fonction $f(x) = \frac{\sin x}{x}$ pour $x \in [0, 4\pi]$ (écrire un script).

Exercice 3

Écrire une fonction pour résoudre un système linéaire où la matrice est triangulaire supérieure. On pourra utiliser l'instruction **size** qui permet de récupérer les deux dimensions d'une matrice :

-- > $[n, m] = \text{size}(A)$

Dans un premier temps, on programmera l'algorithme classique utilisant deux boucles, puis on essaiera de remplacer la boucle interne par une instruction matricielle . Pour tester votre fonction, vous pourrez générer une matrice de nombres pseudo-aléatoires et n'en garder que la partie triangulaire supérieure avec l'instruction **triu** :

-- > $A = \text{triu}(\text{rand}(4, 4))$

Exercice 4

La solution du système d'équations différentielles du 1^{er} ordre :

$$\frac{dx}{dt}(t) = Ax(t), \quad x(0) = x_0 \in \mathbb{R}^n, \quad x(t) \in \mathbb{R}^n, \quad A \in \mathcal{M}_{nn}(\mathbb{R}^n)$$

peut être obtenue en utilisant l'expression l'exponentielle de matrice (cf votre cours d'analyse de 1^{ère} année) :

$$x(t) = e^{At}x_0$$

Comme Scilab dispose d'une fonction qui calcule l'exponentielle de matrice (**expm**), il y a sans doute quelque chose à faire. On désire obtenir la solution pour $t \in [0, T]$. pour cela, on peut la calculer en un nombre n suffisamment grand d'instantanés uniformément répartis dans cet intervalle $t_k = k\delta t$, $\delta t = T/n$ et l'on peut utiliser les propriétés de l'exponentielle pour alléger les calculs :

$$x(t_k) = e^{Ak\delta t}x_0 = e^{k(A\delta t)}x_0 = (e^{A\delta t})^k x_0 = A^{A\delta t}x(t_{k-1})$$

ainsi il suffit uniquement de calculer l'exponentielle de la matrice $A\delta t$ puis de faire n multiplications « matrice vecteur » pour obtenir $x(t_1), x(t_2), \dots, x(t_n)$. Écrire un script pour résoudre l'équation différentielle (un oscillateur avec amortissement) :

$$x'' + \alpha x' + kx = 0, \quad \text{avec exemple } \alpha = 0.1, \quad k = 1, \quad x(0) = x'(0) = 1$$

que l'on mettra évidemment sous la forme d'un système de deux équations du premier ordre. A la fin, on pourra visualiser la variation de x en fonction du temps, puis la trajectoire dans le plan de phase.

Exercice 5

Écrire une fonction $[i, info] = \text{intervalle_de}(t, x)$ pour déterminer l'intervalle i tel que $x_i \leq t \leq x_{i+1}$ par la méthode de la dichotomie (les composantes du vecteur x étant telles que $x_i < x_{i+1}$). Si $t \notin [x_1, x_n]$, la variable booléenne **info** devra être égale à %f (et %Mt dans le cas inverse).

Exercice 6

Écrire une fonction $[y] = \text{signal_fourier}(t, T, cs)$ qui renvoie un début de série de Fourier en utilisant les fonctions :

$$f_1(t, T) = 1, \quad f_2(t, T) = \sin\left(\frac{2\pi t}{T}\right), \quad f_3(t, T) = \cos\left(\frac{2\pi t}{T}\right), \quad f_4(t, T) = \sin\left(\frac{2\pi t}{T}\right), \quad f_5(t, T) = \cos\left(\frac{4\pi t}{T}\right), \dots$$

au lieu des exponentielles. T est un paramètre (la période) et le signal sera caractérisé (en dehors de sa période) par le vecteur cs de ces composantes dans la base f_1, f_2, f_3, \dots . On récupérera le nombre de fonction auxiliaire $[y] = f(t, T, k)$ pour calculer f_T . Enfin tout cela doit pouvoir s'appliquer sur

un vecteur (ou une matrice) d'instants t , ce qui permettra de visualiser facilement un tel signal :

```
-- >T = 1//une periode....
-- >t = linspace(0,T,101)//instants...
-- >cs = [0.1 1 0.2 0 0 0.1]//un signal avec une composante continue
-- >//du fondamental, pas d'harmonique 1 (periode 2T) mais une harmonique 2
-- >[y] = signal_fourier(t,T,cs); //calcul du signal
-- >plot(t,y)//et un dessin...
```

Exercice 7

On considère la suite (ν_n) définie pour tout entier naturel non nul n par : $\nu_n = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots + \frac{1}{n^2}$. Calculer 500 termes de cette suite et les représenter graphiquement.

Rechercher un entier n_0 tel que pour $n \geq n_0$, on ait $\nu_{n+1} - \nu_n \leq 10^{-5}$.

Exercice 8

Dichotomie : Recherche d'une valeur approchée du nombre d'or, solution positive d'équation $x^2 = x + 1$.

- 1) Définir la fonction $f : f(x) = x^2 - x - 1$. Faire tracer la courbe sur l'intervalle $[-5, 5]$, on remarque que la solution est entre 1 et 2.
- 2) Ecrire un programme permettant par dichotomie d'encadrer la solution cherchée dans un intervalle d'amplitude 10^{-4} , en partant des valeurs 1 et 2.

Exercice 9

Flutuation d'échantillonnage : Simuler 100 fois le lancer de 1000 dés. Calculer à chaque fois la fréquence d'apparition de 6. Faire apparaître graphiquement les 100 fréquences.

Définir ensuite une fonction permettant de faire les mêmes calculs en changeant le nombre de lancers.

Exercice 10 Écrire un programme prenant en entrée un trinôme $aX^2 + bX + C$, avec $a \neq 0$, représenté par le tableau de ses coefficients et affichant ses racines réelles si elles existent, ou "aucune racine réelle" sinon.

LOGICIEL R

3.1 Interface d'utilisation sous windows

3.1.1 Fonction des menus

Ce logiciel R forme un interface utilisateur simple. Elle est structurée autour d'une barre de menu et de diverses fenêtres.

- Le menu **file** (ou fichier) : contient les outils nécessaires à la gestion de l'espace de travail ;
- Le menu **Edit** (ou Edition) : contient les habituelles commandes de copier-coller et la boîte de dialogue pour la personnalisation de l'apparence de l'interface ;
- Le menu **Misc** traite de la gestion des objets en mémoire et permet permet d'arrêter une procédure en cours de traitement.
- Le menu **Package** : automatise la gestion et le suivi des librairies de fonctions ;
- Les menus **windows** (ou fenêtre) et **Help** (ou Aide) : assurent des fonctions similaires à celles qu'ils occupent dans les autres applications Windows (les fenêtres, l'aide en ligne et manuels de références de R)

3.1.2 D'autres fenêtres

Parmi les fenêtres, on a la console qui est la fenêtre principale où on réalise par défaut les entrées de commandes et sorties de résultats en mode texte. On y ajoute également un certain nombre de fenêtres telles que les fenêtre graphiques et les fenêtres d'informations (historique des commandes, aide, visualisation de fichier, etc...).

Ces dernières sont toutes appelées à partir de la console par des commandes spécifiques.

3.2 Les objets : Vecteurs, Matrices, Matrices à plus de deux dimensions, liste et structures de données

3.2.1 Objets

Les éléments de base du langage R sont des objets qui peuvent être des données (vecteurs, matrices,...), des fonctions, des graphiques,...

Les objets se différencient en mode (qui sont : `null` (objet vide), `logical`, `numeric`, `complex`, `character`) qui décrivent leur contenu et leur classe.

Les principales classes d'objets sont : `vector`, `matrix`, `array`, `factor`, `time-series`, `data.frame`, `list`

N.B. : On peut avoir des vecteurs, matrices, tableaux, variables,... de mode `null` (objet vide), `logical`, `numeric`, `complex`, `character`.

3.2.2 Les vecteurs

C'est l'objet de base dans R.

```
>a = c(5, 5.6, 1, 4, -5)    Création de l'objet a recevant un vecteur
                             numérique de dimension 5 et de coordonnées 5, 5.6, 1, 4, -5.

>a                           Affichage du vecteur a

>a[1]                        Affichage de la première coordonnée du vecteur a

>b = a[2 : 4]                 création du vecteur numérique
                             b de dimension 3 et de coordonnées 5, 1, -5

>d = a[c(1, 3, 5)]            Création d'un vecteur numérique d de dimension 3
                             et de coordonnées 5, 1, -5

>sum(d)                       Calcul de la somme de d

>length(d)                   Affichage de la dimension de d

>t(d)                         Transposition du vecteur d

>t(d)%*%e                     Produit scalaire entre les vecteurs d et e
```

3.2.3 Les matrices

Les matrices comme les vecteurs, sont de modes quelconque, mais elles ne peuvent pas contenir des éléments de nature différente. La syntaxe de création d'une matrice est :

matrix(vec,nrow=n,ncol=p,byrow=T)

où `vec` est le vecteur contenant, qui seront rangés en colonne (sauf si "`byrow=T`" est choisie) les éléments de la matrice.

```
>a = 1 : 20
>b = sample(1 : 10, 10)
>x1 = matrix(a, nrow = 5)    Création d'une matrice numérique x1 de dimension
                             5 × 4 ayant pour première ligne 1, 6, 11, 16
>x2 = matrix(a, nrow, byrow = T)  Création d'une matrice numérique x2 de dimension
                             5 × 4 ayant pour première ligne 1, 2, 3, 4
>x3 = t(x2)                    Transposition de la matrice x2
>b = x3 %* %x2                 produit de matriciel entre x3 et x2
>dim(x1)                       Affichage de la dimension de x1
```

3.2.4 Les matrices à plus de deux dimensions

Les matrices à plus de deux dimensions sont créées à l'aide de la commande suivante :

`array(vec, c(n, p, q, ...))`

où

- `vec` est le vecteur contenant les éléments de la matrice
- `c(n, p, q, ...)` désigne les dimensions :
 - `n` est le nombre de lignes,
 - `p` le nombre de colonnes,
 - `q` le nombre de lignes,
 - \vdots

```
>x = array(1 : 50, c(2, 5, 5))
>x
>x[1, 2, 2]
>dim(x)
>aperm(x)  Transposition généralisée de x, x[i,j,k] devient x[k,j,i]
```

3.2.5 Les listes

La construction d'une liste passe par la fonction `list(nom1=el1,nom2=el2,...)` , l'utilisation des noms étant facultative.

```
>li = list(num = 1 : 5, y = "couleur", a = T)
>li
>li$num
>li$a
>li[[1]]
>li[[3]]
>a = matrix(c(6, 2, 0, 2, 6, 0, 0, 0, 36), nrow = 3)
>res = eigen(a, symmetric = T)    Diagonalisation de a
>res$values
>res$vectors
```

3.2.6 Les structures de données

Les tableaux de données (`data.frame`) constituent une classe particulière de listes consacrée au stockage des données destinées à l'analyse. Pour créer un tableau de données, on peut regrouper les tableaux de même longueur à l'aide de la commande

`data.frame(nom1=var1,nom2=var2,...)`.

On peut ainsi transformer une matrice en tableau de données en utilisant la commande `as.data.frame(mat)`.

Exemple 13

```
>v1 = sample(1 : 12, 30, rep = T)  Echantillonnage avec remise
                                dans les entiers de 1 et 12
>v2 = sample(LETTERS[1 : 10], 30, rep = T)
>v3 = runif(30)  30 réalisations indépendantes d'une loi
                  uniforme sur [0, 1]
>v4 = rnorm(30)  30 réalisations indépendantes d'une loi
                  normale de moyenne 0 et variance 1
>v1; v2; v3; v4
>xx = data.frame(v1, v2, v3, v4)
>xx
```

3.3 Quelques fonctions usuelles

Loi	Nom	Paramètres	Valeurs par défaut
Beta	beta	shape1,shape2	
Binomiale	binom	size,prob	
Cauchy	cauchy	location,scale	0,1
Khi-Deux	chisq	df	
Exponentielle	exp	1/mean	1
Fisher	f	df1,df2	
Gamma	gamma	shape,1/scale	-1
Géométrique	geom	prob	
Hypergéométrique	hyper	m,n,k	
Log-Normale	lnom	mean,sd	0,1
Logistique	logis	location,scale	0,1
Normale	norm	mean,sd	0,1
Poisson	pois	lambda	
Student	t	dt	
Uniforme	unif	min,max	0,1
Weibull	weibull	shape	

Pour chacune de ces distributions, on dispose de quatre commandes préfixées par une des lettres **d,p,q,r** et suivi du nom de distribution :

- **dnomdist** : fonction de densité pour une distribution probabilité continue et de la fonct. de probabilité ($\mathbb{P}(X = k)$);
- **pnomdist** : fonction de répartition ($\mathbb{P}(X \leq x)$);
- **qnomdist** : fonction de quantité;
- **rnomdist** : génère des réalisations aléatoires indépendantes de la distribution **nomdist**.

3.3.1 Quelques fonctions de statistique exploratoire

```

>data(women)
>names(women)
>attach(women)
>mean(height)  Calcul de la moyenne empirique de
                 variable quantitative height
>var(height)    Calcul de la variance empirique de height
                 estimateur non biaisé (diviseur  $n - 1$ )
>sd(height)     Calcul de l'écart-type de height
>median(height) Calcul de la médiane empirique de height
>quantile(height) Calcul des quantiles empiriques de height
>summary(weight) Résumé de weight
>summary(women)  Résumé de women
>hist(weight, nclass = 15) Histogramme de weight constitué de 15 classes
>cor(height, weight) Calcul du coefficient de corrélation linéaire
>                               empirique entre weight et height
>v1 = rnorm(100)
>hist(v1)
>v2 = factor(sample(letters[1 : 4], 100, rep = T))
>table(v2)       Résumé de la variable qualitative v2
>barplot(table(v2)) Diagramme en barre de v2
>sd(height)      Diagramme en secteur v2

```

3.4 Construction d'une nouvelle fonction

On peut définir une nouvelle fonction soit directement à partir de la console, soit via un éditeur de texte externe grâce à la commande *fix(nom_fonction)*. La seconde possibilité permet la correction du code en cours d'édition, tandis que la première s'effectue ligne par ligne, sans retour en arrière.

La syntaxe générale de la définition d'une nouvelle fonction par l'expression

```
nom_fonction-fonction(arg1[-expr1], arg2[-expr2], ...)
{
  bloc d'instructions
}
```

Les accolades définissent le début et la fin du code source de la fonction ; les crochets ne font pas partie de l'expression mais indiquent le caractère facultatif des valeurs par défaut des arguments.

On peut créer également une fonction personnalisée à partir d'une fonction existente grâce

```
nom_fonction2-edit(nom_fonction1); fix(nom_fonction2)
```

Exemple

```
>x = 2
>carre = function(x) { x - x * x; x }
>carre(2)
>x
>fix(carre)
```

On peut ajouter des commentaires au code en les faisant précéder du symbole #.

3.4.1 Quelques éléments de programmation

Cette partie concerne les commandes **if**, **while**, **for**

Exemple 14

```
>bool = T
>i = 0
>while(bool == T) {i = i + 1; if (i > 10) {bool = F}}
>i
>s = 0
>for (i in 1 : 1000) {s = s + x[i]}
>s
>un = rep(1, 10000)
>t(un)% * %x
>s = 0
>system.time(for (i in 1 : 10000) {s = s + x[i]})[3]
>system.time(t(un)% * %x)[3]
```

N.B. : Le logiciel **R** peut avoir des problème de mémoire de ce fait, il est préférable de les remplacer par les outils de calcul matriciel.

3.5 Les entrées/Sorties et gestion des objets créés

3.5.1 Les entrées/Sorties

Les données que l'on souhaite analyser proviennent de source externes sous forme de fichiers. Les objets créés doivent pouvoir être sauvegardés dans des fichiers afin d'être transportables.

- **Formats propriétaire** : `save()` autorise la sauvegarde de n'importe quelle liste d'objet en mémoire ;
- **Fichiers textes ASCII** : ils sont prisent en charge par la commande `scan()`
- **Logiciels statistiques** : La librairie **foreign** offre ces outils pour une sélection des logiciels statistiques plus courants, à savoir MINITAB, S-PLUS, SAS, SPSS et STATS.

3.5.2 gestion des objets créés

Lors de l'exécution de **R**, les fichiers **.RData** et **.Rhistory** sont automatiquement créés dans le répertoire de travail. Losqu'on quitte **R** à l'aide de la commande `q()`, les nouveaux objets créés peuvent être sauvegardé dans les fichier : **.Rdta** ou **.Rhistory** en mode texte.

- `history()` : permet de visualiser la suite de commandes que l'on a tapée
- `getwd()` : permet de connaître le répertoire de travail courant au cours d'une session ;
- `ls()` : permet de visualiser la liste des objets créés
- `rm()` : permet de détruire des objets

3.6 Graphisme avec R

Le fenêtre graphiques sous Windows sont nommées **windows**. On peut ouvrir une fenêtre graphique avec la commande `x11()`.

Pour ouvrir un fichier, on utilise : `postscript("mesgraphiques.eps")` ou `pdf("mesgraphiques.pdf")`.

La commande `dev.list()` permet d'afficher la liste des dispositifs graphiques ouvertes. Cependant les graphiques seront tracés dans le dispositif graphique actif ; ainsi, les commandes (ou fonctions) :

- `dev.cur()` permet de connaître ce dispositif.
- `dev.set(i)` permet de rend le ième dispositif actif
- `dev.off()` ferme le dispositif actif,
- `dev.off(i)` ferme le ième dispositif.

D'autre part, on peut partitionner la fenêtre graphique active à l'aide des commandes `split.screen(x)` (x est un vecteur) ou `layout(m.widths=w.heights)h)` (m matrice, w vecteurs, h vecteur).

Pour visualiser une fenêtre graphique partitionnée avec `layout`, on utilise `layout.show(n)` où n est le nombre de sous-fenêtres.

3.6.1 Les principales commandes graphiques

Les principales commandes graphiques sont données dans le tableau suivant.

<code>plot(x)</code>	Tracer le graphe des valeurs de x ordonnées sur l'axe des abscisses.
<code>plot(x,y)</code>	Tracer le graphe de y en fonction de x
<code>sunflowerplot(x,y)</code>	Idem mais les points superposés sont dessinés sous forme de fleurs dont le nombre de pétales correspond au nombre de points
<code>pie(c)</code>	Tracer un graphe en camembert.
<code>boxplot(x)</code>	Tracer le graphe en "boîtes et moustaches" de x .
<code>stripplot(x)</code>	Tracer le graphe des valeurs de x sur une ligne
<code>interaction.plot(f1,f2,x,fun=mean)</code>	Tracer le graphe des moyennes de x en fonction des valeurs des facteurs $f1$ (sur l'axe des abscisses) et $f2$ (plusieurs graphes).
<code>coplot(x~ y z)</code>	Tracer le graphe bivarié de x et y pour chaque de z
<code>matplot(x,y)</code>	Tracer le graphe bivarié de la 1ère colonne de x contre la 1ère colonne de y , la 2ème colonne de x contre la 2ème colonne de y ...
<code>pairs(x)</code>	Si x est une matrice ou un data.frame, tracer tous les graphes bivariés entre les colonnes de x
<code>plot.ts(x),ts.plot(x)</code>	Tracer le graphe d'une série temporelle x en fonction du temps.
<code>hist(x,freq=T)</code>	Tracer un histogramme des fréquences de x .
<code>barplot(x)</code>	Tracer un histogramme des valeurs de x .
<code>qqnorm(x)</code>	Tracer les quantiles de x en fonction de ceux attendus d'une loi normale.
<code>qqplot(x,y)</code>	Tracer les quantiles de y en fonction de ceux de x .
<code>contour(x,y,z)</code>	Tracer des courbes de niveau.
<code>filled.contour(x,y,z)</code>	Idem mais les aires entre contours sont colorées
<code>image(x,y,z)</code>	Idem mais en couleur.
<code>persp(x,y,z)</code>	Idem mais en 3D
<code>symbols(x,y,...)</code>	Dessiner aux coordonnées données par x et y des symboles (étoiles, cercles, boxplots...).

3.6.2 Les options

Pour les fonctions, les options sont les suivantes :

<pre>axes=TRUE ou FASTE Tpe="n"."p"."1"."b"."h"."s"... col="blue",col.axis,col.main.. bg="yellow" xlim=c(0,10),ylim=c(0,20) xlab="abscisse",ylab="ordonnée" main="title" sub="sstable" bty="n","o"... cex=1.5, cex.axis, cex.main... font=1. font.axis... las="0" lty="1" lwd=1.5 pch="+","o" pe=1.5 tck,tcl mfcol=c(3,2),mfrow=c(3,2)</pre>	<p>Si TRUE, les axes et le cadre sont tracé</p> <p>Précise le type de graphe dessiné. type="n" supprime le graphe.</p> <p>Précise la couleur du graphe, des axes, du titre...</p> <p>Précise la couleur du fond.</p> <p>Précise les limites des axes.</p> <p>Précise les annotations des axes.</p> <p>Précise le titre du graphe.</p> <p>Précise le sous-titre du graphe.</p> <p>Contrôle comment le cadre est tracé. bty="n" supprime le cadre</p> <p>Contrôle la taille des caractères.</p> <p>Précise la police du texte.</p> <p>Contrôle comment sont orientés les annotations des axes.</p> <p>Contrôle le type de lignes tracées.</p> <p>Contrôle la largeur des lignes.</p> <p>Contrôle le type de symbole utilisé pour le tracé des points.</p> <p>Contrôle la taille en points du texte et des symboles.</p> <p>Précise la longueur des graduations sur les axes.</p> <p>Partitionne le graphe en 3 lignes et 2 colonnes. figures sont remplies colonnes par colonnes ou lignes par lignes.</p>
--	--

3.7 Exercices de Travaux Pratiques sous R

Exercice 1

1) Créer les vecteurs suivants :

- y_0 est constitué de la suite des entiers de 1 à 9
- On pose $d = 4$, y_1 contient trois fois la valeur de d , puis trois fois celle de d^2 , puis trois fois celle de \sqrt{d} ,
- y_2 est une suite arithmétique prenant ses valeurs entre 1 et 20 avec un pas de deux.
- y_3 contient 10 chiffres compris entre 1 et 30 avec un intervalle constant.

2) Extraite de y_3

- le 3^{ième} élément
- tous les éléments sauf le 3^{ième}
- tous les éléments inférieurs à 12

3) Comparer les commandes suivantes

- `matrix(y3,nrow=2)`
- `matrix(y3,nrow=2,byrow=T)`

4) Construire une matrice qui contient sur la première ligne y_10 et sur la seconde y_1

Exercice 2

1) Construire la matrice Z suivante :

$$Z = \begin{pmatrix} 2 & 4 & 1 & 2 \\ 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 \\ 11 & 12 & 35 & 7 \end{pmatrix}$$

2) Affiche l'élément de Z contenu dans

- La première ligne et troisième colonne
- La première ligne de Z
- La troisième colonne de Z
- la sous-matrice après avoir enlevé la première ligne et la première colonne de Z

Exercice 3

- 1) Construire une fonction qui calcule la valeur de la fonction $f : x \mapsto \sin(x)^2 + \sqrt{|x-3|}$.
- 2) Tracer la courbe représentative de la fonction f sur le domaine $[-6, 3]$
- 3) reprendre les mêmes questions pour

$$g : x \mapsto \begin{cases} \sin(x)^2 \log(x) & x > 0 \\ \sin(x)^2 x & x \leq 0 \end{cases}$$

Exercice 4

X suit une loi binomiale de paramètres $(50, 1/3)$. Calculer la probabilité des événements suivants

$$[X = 1] ; [X \leq 5] ; [X \geq 15] ; [X \notin \{15, 3, 4, 10\}] ; [X \in 2\mathbb{N}].$$

Exercice 5

- 1) Créer une fonction qui pour un couple donné $(n, p) \in \mathbb{N} \times [0, 1]$, évalue le maximum de l'erreur commise lorsque l'on approche la loi binomiale par la loi de Poisson

$$M_{n,p} = \max_{k=0,\dots,n} |P(X_n = k) - P(Y_n = k)|$$

où X_n suit une loi binomiale de paramètres (n, p) et Y_n suit une loi de poisson de paramètre np .

- 2) Pour $p = 1/2$, représenter graphiquement l'erreur en fonction de n
- 3) Pour $n = 40$, représenter graphiquement l'erreur en fonction de p

Exercice 6

Soit X une variable aléatoire qui suit une loi normale standard $\mathcal{N}(0, 1)$. Calculer la probabilité des événements suivants

- 1) $[X \leq 1]$; $[X \geq 2.6]$; $[0.5 < X \leq 1]$;
- 2) Calculer le quantile d'ordre $a = 0.75$, c'est-à-dire la valeur de x telle que

$$P(X \leq x) = a$$

- 3) Représenter graphiquement la densité et la fonction de répartition de la loi de X
- 4) Simuler un échantillon $(x_1; \dots; x_{100})$ de taille $n = 100$ suivant la loi de X
- 5) Représenter les valeurs de l'échantillon simulé.
- 6) Créer une liste qui contient la moyenne empirique $\frac{1}{100} \sum_{i=1}^{100} x_i$, le minimum de $(x_1; \dots, x_{100})$ et le maximum de $(x_1; \dots, x_{100})$.

Exercice 7 (Sans utiliser les boucles FOR)

On se donne $x = (x_1, \dots, x_n)$ une séquence de longueur n .

- 1) Construire une fonction qui à pour paramètre d'entrée x et retourne le scalaire

$$S_n = \frac{1}{n} \sum_{i=1}^n x_i^2 - \left(\frac{1}{n} \sum_{i=1}^n x_i \right)^2$$

- 2) Construire une fonction qui a pour paramètre d'entrée x et retourne le vecteur (S_1, \dots, S_n)
- 3) Représenter graphiquement (S_1, \dots, S_n) pour le vecteur x constitué de nombres aléatoires iid suivant la loi uniforme sur $[0, 1]$.

Exercice 8 (Sans utiliser les boucles FOR)

Soit $N = (N_{i,j})$ une matrice. On note N_i la somme des termes de la i -ème ligne, N_j la somme des termes de la j -ème colonne et n la somme des termes de la matrice. Construire une fonction qui retourne la quantité suivante

$$\sum_i \sum_j \frac{(N_{i,j} - \frac{N_i N_j}{n})^2}{\frac{N_i N_j}{n}}.$$

Exercice 9

On construit une partition de l'intervalle $]0, 1]$ en prenant $\bigsqcup_{i=1}^p A_i$ avec $A_1 =]0, a_1]$, $A_i =]a_{i-1}, a_i]$ pour $i = 2 \dots p-1$ et $A_p =]a_{p-1}, 1]$.

- 1) En utilisant une boucle **while**, construire une fonction qui pour un réel donné x et une suite a retourne

- s'il existe, l'entier i tel que $x \in A_i$,
- un message d'avertissement sinon

2) Expliquer le code suivant

```
ind = function(x,a)
{
  if (x <= 0 | x > 1) stop('x n'est pas dans ]0,1[')
  sum(x - a > 0) + 1
}
```

```
find = function(x,a)
{
  sapply(x,ind,a)
}
```

Exercice 10 Pour $(n = 100, p = 0.5)$, puis $(n = 1000, p = 0.5)$, $(n = 10000, p = 0.5)$, $(n = 1000, p = 0.3)$, $(n = 1000, p = 0.8)$

1. Simuler un échantillon de n variables aléatoires de Bernouilli, de paramètre p ,
 - (a) en utilisant la fonction **rbinom**
 - (b) en utilisant la fonction **runif**
 - (c) en utilisant la fonction **sample**
2. Calculer les fréquences de 0 et de 1 dans l'échantillon,
 - (a) en utilisant la fonction **sum**
 - (b) en utilisant la fonction **which**
 - (c) en utilisant la fonction **table**
3. Représenter les fréquences de 0 et de 1 par un diagramme en barres (fonction **barplot**). Représenter par un double diagramme en barre, les fréquences empiriques de 0 et de 1 en bleu et les probabilités théoriques $(1 - p)$ et p en rouge.
4. Utiliser votre échantillon pour simuler n parties d'un jeu de pile et face où la probabilité de gagner 1 euro est p , la probabilité de perdre 1 euro est $1 - p$. Calculer les valeurs successives de la fortune d'un joueur dont la fortune initiale est nulle (fonction **cumsum**). Représenter graphiquement ces valeurs (fonction **plot**).
5. Calculer les valeurs successives de la moyenne empirique des i premières valeurs de l'échantillon initial, pour i allant de 1 à n . Représenter graphiquement ces valeurs en bleu et superposer sur le même graphique la droite horizontale d'ordonnée p en rouge (fonction **abline**).

Bibliographie

- [1] Benjamin BAYART (GEUT), Joli manuel pour LATEX 2 ϵ .
(Voir l'adresse web : <http://www.ntg.nl/doc/bayart/lxmanuel.pdf>)
- [2] Tobias Oetiker, Hubert Partl, Irene Hyna et Elisabeth Schlegl, Une courte (?) introduction à Latex 2 ϵ . Traduit en français par Mathieu Herrb. Version 3.20 Novembre 2001.
- [3] Bruno Pinçon. Introduction à Scilab. Version 0.996.
(Voir l'adresse web : <http://cermics.enpc.fr/bl/PS/pinson.pdf>)
- [4] Marie Postel. Introduction au logiciel Scilab. Version révisée janvier 2009.
(<http://www.ann.jussieu.fr/postel/scilab/NoticeScilab.pdf>)
- [5] Jean-Michel MARIN. Initiation au logiciel R.
(<http://www.ceremade.dauphine.fr/xian/Noise/R.pdf>)
- [6] Emanuel Paradis. R pour débutants. Institut des Sciences de l'Evolution. université Montpellier II. F-34095 Montpellier cedex 05, France.
(Voir l'adresse web : <http://cran.r-project.org/doc/contrib/Paradis-rdebuts...fr.pdf>)