

Enseignement annuel d'informatique et algorithmique

L'objectif est d'asseoir les connaissances des étudiants en algorithmique et de les entraîner à l'utilisation de l'informatique en mathématiques au travers de thèmes empruntés au programme pour comprendre, illustrer et éclairer les notions introduites. Dès qu'un calcul numérique est envisagé, dès qu'un problème incite à tester expérimentalement un résultat, dès qu'une situation aléatoire peut être modélisée avec des outils informatiques, le recours à des algorithmes et des logiciels devra devenir naturel.

Les séances de travaux pratiques doivent se faire le plus souvent possible sur ordinateur. Les étudiants, au cours de leurs études ultérieures puis de leur parcours professionnel, seront amenés à utiliser des outils informatiques divers choisis pour leurs fonctionnalités, et seule une pratique régulière de ces outils informatiques peut leur permettre d'en acquérir la maîtrise. De plus, en adoptant cette démarche exploratoire permise par le dialogue interactif avec la machine, cette pratique peut s'avérer bénéfique pour les apprentissages et faciliter la compréhension de concepts plus abstraits.

Le langage retenu pour la programmation dans le programme des classes économiques et commerciales, option mathématiques approfondies, est Python.

Toute la richesse du langage Python ne peut pas être entièrement maîtrisée par un étudiant, aussi seules les fonctions et commandes introduites en figurant dans la sous-partie « Commandes exigibles » sont exigibles. Néanmoins, se contenter de ces seules commandes, en ignorant les nombreuses possibilités et commodités du langage, se révélerait rapidement contraignant et limitatif. De nouvelles commandes Python peuvent donc être introduites, mais cela devra se faire avec parcimonie, l'objectif principal de l'activité informatique reste la mise en pratique des connaissances mathématiques. On favorisera à cette occasion l'autonomie et la prise d'initiatives des étudiants grâce à l'utilisation de l'aide de Python, et à l'usage d'opérations de « copier-coller » qui permettent de prendre en main rapidement des fonctions nouvelles et évitent d'avoir à connaître par cœur la syntaxe de commandes complexes.

Seules les notions de Python indiquées dans le programme sont exigibles. La syntaxe précise des commandes devra être rappelée.

1 - Programmation d'algorithmes et de fonctions

<code>if ...:</code>	Structures conditionnelles.
<code>...</code>	
Emploi de <code>else</code> , <code>elif</code>	
<code>for k in range(a,b):</code>	T peut être une matrice, un vecteur, une chaîne de caractères. Les commandes <code>break</code> et <code>continue</code> ne sont pas exigibles.
<code>for k in T:</code>	
<code>while ...:</code>	Définition d'une fonction.
<code>def f(x,y):</code>	
<code>...</code>	
<code>return ...</code>	

2 - Commandes exigibles

Il s'agit de la liste des commandes utiles pour les travaux pratiques des deux années de formation. Il n'y a pas lieu d'introduire en première année les commandes qui relèvent de notions de seconde année.

a) Disponibles de base dans Python

Affectation : `nom = expression`

permet d'insérer un commentaire

+	-	*	/	**
---	---	---	---	----

==	>	<	>=	<=	!=
----	---	---	----	----	----

True	False	and	or	not
------	-------	-----	----	-----

`from ... import *, import ... as`

b) Dans la librairie `numpy`

Exemple d'importation : `import numpy as np`

`np.array, np.zeros, np.ones, np.eye,`
`np.linspace, np.arange`

+	-	*	/	**
---	---	---	---	----

==	>	<	>=	<=	!=
----	---	---	----	----	----

`a,b = np.shape(M)`

`np.dot, np.transpose`

`np.sum, np.min, np.max, np.mean,`
`np.cumsum, np.median, np.var, np.std`

`np.exp, np.log, np.sin, np.cos,`
`np.sqrt, np.abs, np.floor`

`np.e, np.pi`

c) Dans la librairie `numpy.linalg`

Exemple d'importation : `import numpy.linalg as al`

`al.inv, al.rank, al.matrix_power,`
`al.solve, al.eig`

d) Dans la librairie `numpy.random`

Exemple d'importation : `import numpy.random as rd`

L'expression peut être du type numérique, booléen, matriciel (`ndarray`) ou chaîne de caractères.

Les étudiants doivent savoir faire un usage judicieux des commentaires.

Opérations arithmétiques de base.

Comparaison, test.

Logique.

Importation d'une bibliothèque.

Création de vecteurs et de matrices. Extraction ou modification d'un élément, d'une ligne ou d'une colonne d'une matrice.

Opérations arithmétiques de base : coefficient par coefficient.

Comparaison de deux matrices (`M == N`), comparaison d'une matrice et d'un nombre (`M >= 1`).

Taille de la matrice `M`.

Syntaxes exigibles : `np.transpose(M)`, `np.dot(M1,M2)`. L'usage de méthode comme `M.transpose()`, `M1.dot(M2)` est non-exigible.

Ces opérations peuvent s'appliquer sur une matrice entière ou bien pour chaque colonne (ou chaque ligne). Exemple : `mean(M)`, `mean(M,0)`, `mean(M,1)`

Ces fonctions peuvent s'appliquer à des variables numériques ou vectoriellement (à des matrices ou vecteurs) élément par élément. On pourra utiliser la commande `f = np.vectorize(f)` mais elle n'est pas exigible.

`rd.random`, `rd.binomial`, `rd.randint`,
`rd.geometric`, `rd.poisson`,
`rd.exponential`, `rd.normal`, `rd.gamma`

On utilisera ces fonctions pour générer un nombre aléatoire ou bien un vecteur ou une matrice à coefficients aléatoires. Exemple : `rd.binomial(10,0.2)`, `rd.binomial(10,0.2,100)`, `rd.binomial(10,0.2,[100,10])`

e) Dans la librairie `scipy.special`

Exemple d'importation : `import scipy.special as sp`

`sp.ndtr`

Fonction Φ

f) Dans la librairie `matplotlib.pyplot`

Exemple d'importation : `import matplotlib.pyplot as plt`

`plt.plot`, `plt.show`

Représentations graphiques de fonctions, de suites. On pourra utiliser les commandes `xlim`, `ylim`, `axis`, `grid`, `legend` mais elles ne sont pas exigibles.

`plt.hist`

La maîtrise des options de cette fonction n'est pas exigible.

`plt.contour`

Tracés de lignes de niveau en lien avec `np.meshgrid`. La maîtrise de cette fonction n'est pas exigible.

`plt.quiver`

Tracés de gradients en lien avec `np.meshgrid`. La maîtrise de cette fonction n'est pas exigible.

g) Utilisation de la fonction Axes3D

Exemple d'importation :

```
from mpl_toolkits.mplot3d import Axes3D
ax = Axes3D(plt.figure())
```

`ax.plot_surface`

Représentation de surfaces en lien avec `np.meshgrid`. La maîtrise de cette fonction n'est pas exigible.

3 - Liste de savoir-faire exigibles en première année

Représentation graphique d'une fonction.

Calcul des termes et représentation graphique d'une suite.

Représentation des points (n, u_n) . Pour une suite définie par récurrence $u_{n+1} = f(u_n)$, représentation des termes de la suite à partir du graphe de f .

Calculs de valeurs approchées de la limite d'une suite ou de la somme d'une série.

On utilisera des structures répétitives et conditionnelles en exploitant l'étude mathématique. La détermination du rang d'arrêt du calcul résultera directement de l'étude mathématique ou d'un algorithme qui en découle.

Calcul approché de la racine d'une équation du type $f(x) = 0$.

Valeur approchée d'une intégrale par la méthode des rectangles.

Utilisation de la fonction `rd.random` pour simuler des expériences aléatoires.

Simulation d'échantillons de lois usuelles.

Série statistique associée à un échantillon.

Approche expérimentale de la loi de Gauss.

Calcul approché d'une probabilité.

Résolution de systèmes $MX = B$.

On utilisera différentes méthodes dont certaines résulteront d'une étude mathématique (suites récurrentes, encadrements, dichotomie).

Pour des fonctions f à primitive F connue, on pourra vérifier expérimentalement le lien entre primitive et intégrale, en comparant $F(b) - F(a)$ avec une valeur approchée de $\int_a^b f(t)dt$.

On pourra simuler ainsi des lois binomiale et géométrique.

On pourra utiliser les fonctions `rd.binomial`, `rd.randint`, `rd.geometric`, `rd.poisson`. Fréquences, fréquences cumulées, histogramme. Moyenne, médiane. Variance et écart-type empiriques.

Sur les lois usuelles, on pourra faire un lien entre fréquences et loi, fréquences cumulées et fonction de répartition, moyenne et espérance, variance empirique et variance.

On pourra comparer expérimentalement les lois $\mathcal{B}(n, \frac{\lambda}{n})$ et $\mathcal{P}(\lambda)$.

On pourra superposer la courbe de $x \mapsto \frac{e^{-x^2/2}}{\sqrt{2\pi}}$ et l'histogramme d'un échantillon de $\frac{X - np}{\sqrt{np(1-p)}}$, où $X \hookrightarrow \mathcal{B}(n, p)$.

Approche intuitive de l'estimation : si $P(A)$ est difficile à calculer, on peut simuler N fois l'expérience et assimiler $P(A)$ à la fréquence de réalisation de A .

On pourra programmer l'algorithme du pivot de Gauss sur un exemple. En pratique on utilisera plutôt la fonction `al.solve`.

TRAVAUX PRATIQUES DE MATHÉMATIQUES AVEC PYTHON

En première année, les élèves ont acquis les bases de manipulation du logiciel Python. L'objectif de l'enseignement d'informatique de seconde année est de permettre aux étudiants d'utiliser Python de manière judicieuse et autonome pour illustrer ou modéliser des situations concrètes en mobilisant leurs connaissances mathématiques.

Les séances de travaux pratiques doivent se faire le plus souvent possible sur ordinateur. Les étudiants, au cours de leurs études ultérieures puis de leur parcours professionnel, seront amenés à utiliser des outils informatiques divers choisis pour leurs fonctionnalités, et dès que seule une pratique régulière de ces outils informatiques peut leur permettre d'en acquérir la maîtrise. De plus, en adoptant cette démarche exploratoire permise par le dialogue interactif avec la machine, cette pratique peut s'avérer bénéfique pour les apprentissages et faciliter la compréhension de concepts plus abstraits.

Le programme d'informatique s'articule autour de quatre thèmes : statistiques descriptives bivariées, fonctions de plusieurs variables, simulation de lois, estimation ponctuelle ou par intervalle de confiance.

L'ordre dans lequel les thèmes sont abordés est libre, mais il est préférable de mener ces activités en cohérence avec la progression du cours de mathématiques.

Dans certains thèmes, il s'avérera nécessaire d'introduire de nouvelles notions ou approches mathématiques. Celles-ci devront être explicitées en préambule des séances d'informatique et ne pourront en aucun cas être exigibles des étudiants. Certaines seront propres à un thème particulier, d'autres (comme par exemple les méthodes de Monte-Carlo) pourront au contraire être envisagées de manière transversale. Toutes les précisions nécessaires devront toujours être données lors de leur utilisation.

Toute la richesse du langage Python ne peut pas être entièrement maîtrisée par un étudiant, aussi seules certaines fonctions et commandes sont exigibles. Néanmoins, se contenter de ces seules commandes, en ignorant les nombreuses possibilités et commodités du langage, se révélerait rapidement contraignant et limitatif. De nouvelles commandes Python peuvent donc être introduites, mais cela devra se faire avec parcimonie, l'objectif principal de l'activité informatique reste la mise en pratique des connaissances mathématiques. Dans les sujets, les commandes introduites devront être présentées en préambule et toutes les précisions nécessaires seront données lors de leur utilisation et leur interprétation. On favorisera à cette occasion l'autonomie et la prise d'initiatives des étudiants grâce à l'utilisation de l'aide de Python, et à l'usage d'opérations de « copier-coller » qui permettent de prendre en main rapidement des fonctions nouvelles et évitent d'avoir à connaître par cœur la syntaxe de commandes complexes.

L'objectif de ces travaux pratiques n'est pas l'écriture de longs programmes mais l'assimilation de savoir-faire et de compétences spécifiés dans la liste des exigibles et rappelés en préambule de chaque thème.

Les exemples traités dans un thème devront être tirés, autant que possible, de situations réelles (traitement de données économiques, sociologiques, historiques, démographiques, en lien avec le monde de l'entreprise ou de la finance, etc), en faisant dès que possible un rapprochement avec les autres disciplines.

I - Liste des exigibles

1 - Commandes

Les commandes exigibles ont été listées dans le programme de première année. On rappellera dans les sujets toutes les syntaxes des commandes non exigibles.

2 - Savoir-faire et compétences

C1 : Produire et interpréter des résumés numériques et graphiques d'une série statistique (simple, double) ou d'une loi.

C2 : Modéliser et simuler des phénomènes (aléatoires ou déterministes) et les traduire en langage mathématique.

C3 : Représenter et exploiter le graphe d'une fonction d'une, deux variables.

C4 : Représenter et interpréter différents types de convergences.

C5 : Utiliser la méthode de Monte-Carlo sur des exemples pertinents (calcul approché d'intégrales, de probabilités).

C6 : Porter un regard critique sur les méthodes d'estimation et de simulation.

II - Liste des thèmes

1 - Statistiques descriptives bivariées

(Durée indicative : 3 heures. Compétences développées : **C1** et **C6**)

Série statistique à deux variables, nuage de points associé.

Point moyen (\bar{x}, \bar{y}) du nuage.

Covariance et coefficient de corrélation empiriques, droites de régression.

On tracera le nuage de points et les droites de régression et on pourra effectuer des pré-transformations pour se ramener au cas linéaire.

On différenciera les variables explicatives des variables à expliquer.

2 - Fonctions de plusieurs variables

(Durée indicative : 3 heures. Compétences développées : **C2** et **C3**)

Graphe d'une fonction de deux variables, lignes de niveau, plan affine tangent au graphe. Dérivées partielles et dérivées directionnelles, représentation du gradient.

Position du graphe par rapport au plan affine tangent au graphe, lien avec les valeurs propres de la matrice hessienne, points selles.

Étude d'extrema locaux et globaux. Extrema sous contrainte linéaire.

À cette occasion, on pourra mettre en évidence l'orthogonalité du gradient avec les tangentes aux lignes de niveau du graphe d'une fonction de deux variables.

Programmation de fonctions variées permettant de mettre en évidence les notions d'extrema locaux ou globaux, avec ou sans contrainte. On pourra prendre des exemples issus de l'économie ou de la finance.

3 - Simulation de lois

(Durée indicative : 6 heures. Compétences développées : **C1**, **C2**, **C3** et **C6**)

Dans toutes les simulations effectuées, on pourra comparer les échantillons obtenus avec les distributions théoriques, en utilisant des diagrammes en bâtons et des histogrammes. On pourra aussi tracer la fonction de répartition empirique et la comparer à la fonction de répartition théorique.

Méthode d'inversion.

Application de la méthode d'inversion pour la simulation par exemple des lois exponentielles ou de Cauchy.

On pourra mettre en évidence, grâce aux simulations, qu'une variable aléatoire suivant une loi de Cauchy n'admet pas d'espérance.

Méthodes de simulation d'une loi géométrique.

Simulations informatiques d'une loi normale par utilisation du théorème limite central appliqué à différentes lois.

Utilisation d'une loi de Bernoulli et d'une boucle `while`, utilisation d'une loi exponentielle et de la fonction `floor`, utilisation de la librairie `numpy.random`.

Comparaison entre différentes méthodes de simulation d'une loi normale.

Utilisation de la librairie `numpy.random`.

On pourra s'intéresser au cas particulier de 12 variables aléatoires indépendantes suivant une même loi uniforme.

4 - Estimation ponctuelle et par intervalle de confiance

(Durée indicative : 6 heures. Compétences développées : **C2**, **C4**, **C5** et **C6**)

Méthode de Monte-Carlo : principe, garanties d'approximation.

Cette méthode permet d'estimer des quantités qu'il est difficile de calculer explicitement mais qu'il est facile d'approcher par simulation (probabilités d'événements, espérances de variables aléatoires).

Ainsi, on pourra estimer par exemple les valeurs prises par la fonction de répartition de la somme ou du produit de deux variables aléatoires.

On pourra justifier par simulation la validité de l'approche par intervalle de confiance asymptotique à partir d'un certain rang.

Comparaison de différents estimateurs ponctuels d'un paramètre.

On pourra utiliser des données issues de situations réelles ou créer plusieurs jeux de données par simulation. Dans ce dernier cas, on pourra comparer les lois des estimateurs par exemple à l'aide d'histogrammes.

Comparaison des intervalles de confiance d'un paramètre obtenus par différentes méthodes.

Estimation par intervalle de confiance du paramètre d'une loi de Bernoulli et de l'espérance d'une loi normale.

La comparaison pourra se faire en calculant les demi-largeurs moyennes des intervalles et leurs niveaux de confiance.