Guillaume Saupin

70 CONCEPTS MATHÉMATIQUES

expliqués avec PYTHON

DUNOD

Couverture : Florie Bauduin
Crédits iconographiques pour la couverture:
© Shutterstock / Julia Tim, GoodStudio, Pavlo S, cash1994, VectorHot, vectorwin

11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-085425-7

Table des matières

Chapitre 1

Démarrer sur de bonnes bases

1.1 Préambule

Ce livre se destine aux lecteurs désireux d'assimiler les grandes idées mathématiques à travers une approche de type « apprendre en construisant ».

Cette approche, théorisée par Papert Seymour au sein de son laboratoire au MIT, met en avant le bénéfice pédagogique obtenu en construisant les systèmes étudiés.

Cette méthode s'applique à de nombreux domaines, comme par exemple la mécanique automobile, mais peut aussi être mise en œuvre pour des matières plus théoriques comme les mathématiques et la physique. Pour la mécanique, il s'agira par exemple de construire une boîte de vitesses, en imprimant les pièces en 3D.

Pour les mathématiques, le sujet de cet ouvrage, le matériau de construction naturel est la programmation. Les langages de programmation permettent en effet de donner vie aux concepts mathématiques les plus complexes, en permettant non seulement de les matérialiser sous forme graphique, mais aussi de les manipuler et d'expérimenter avec eux.

Tout au long de ce livre, le lecteur verra comment donner vie aux nombreuses et fructueuses idées mathématiques à l'aide de quelques lignes de code.

Motivations

Une image vaut mille mots

La raison principale qui a motivé la rédaction de ce livre est la matérialisation qu'apporte l'informatique aux mathématiques. De la même manière qu'un schéma permet souvent d'expliquer un concept ou une idée complexe rapidement, le code informatique permet de concrétiser rapidement toute idée mathématique. Mieux encore, de la même façon qu'il est possible de raisonner sur une idée en déplaçant des éléments d'un schéma, et en faisant évoluer sa réflexion graphiquement, le code permet cette plongée du monde abstrait des mathématiques dans un médium malléable. Ce livre illustre donc tous les concepts mathématiques à l'aide du nombre minimal de lignes de code permettant de leur donner chair.

Matérialiser les abstractions

Les mathématiques ont cette image d'une science compliquée et abstraite, construite par des esprits rêveurs et imaginatifs. Cette représentation est majoritairement inexacte et ne résiste pas à

une étude biographique et épistémologique des mathématiciens et de leurs découvertes. Gauss s'est fait connaître dans toute l'Europe en calculant la trajectoire de Cérès, et Euler appliquait son immense maîtrise des mathématiques au calcul des pensions de retraite de son institut de recherche.

Ces derniers ont bien souvent développé leurs théories pour répondre à des problématiques concrètes, et offrir des solutions à ces problèmes.

Ce qui crée cette impression de complexité découle de l'enseignement qui en est fait, où deux millénaires de mathématiques sont abordés en peu de temps. Le filtre des siècles a introduit toutes ces abstractions, qui ont l'avantage d'accélérer les calculs et les raisonnements, mais qui cachent la mécanique, la matérialité des concepts sur lesquels elles reposent. Implémenter ces concepts en Python permet de leur rendre chair.

Mécanisation

Une autre perspective qu'apporte la programmation aux mathématiques, c'est la mécanisation qui en découle. En effet, dès lors qu'un concept est implémenté et exécuté sur un ordinateur, il se réduit au final en une succession d'opérations simples, mécaniquement appliquées. Cette représentation est très fructueuse, surtout lorsqu'il est question d'intelligence artificielle.

Un formidable accélérateur

La conséquence majeure de l'utilisation de l'informatique, pour découvrir les mathématiques, est l'accélération formidable qu'elle apporte. Tous les points cités ci-dessus contribuent à réduire le temps d'acquisition des notions mathématiques, tout en renforçant la mémorisation. L'approche suivie dans cet ouvrage permettra donc au lecteur d'acquérir rapidement les nombreux concepts mathématiques présentés.

Public

Basé essentiellement sur le programme de Licence de mathématiques, cet ouvrage en parcourt la plupart des notions. Il a donc été conçu pour pouvoir être abordé par un élève de premier cycle universitaire.

Cependant, il peut aussi profiter à tout élève en Master ou au-delà, évoluant dans le domaine des mathématiques, mais aussi des sciences de l'ingénieur ou de l'informatique, qui chercherait un récapitulatif synthétique des grandes notions mathématiques avec un prisme opérationnel.

Enfin, toute personne curieuse de comprendre en finesse les mathématiques, leur usage en science informatique, et finalement tous les outils et services qui peuplent nos ordinateurs, nos téléphones et nos vies trouvera ici matière à réflexion.

Notions

Les notions abordées tout au long de cet ouvrage sont essentiellement tirées du programme de Licence de mathématiques. Le champ des mathématiques est très large, et il a bien sûr fallu opérer un choix.

Ce dernier s'est fait en partie suivant les goûts de l'auteur, mais surtout sur la base de son expérience de près de deux décennies dans la recherche en mathématiques appliquées, en intelligence artificielle au sein du CEA mais aussi de start-ups.

Toutes les notions présentées ici seront donc utiles à tout ingénieur travaillant dans le domaine des sciences, de l'informatique, de la biologie...

Pour faciliter la navigation parmi ces concepts, ils ont été regroupés dans les grandes sections suivantes :

— Théorie des nombres
— Analyse
— Algèbre linéaire
— Statistique
— Géométrie
— Intelligence artificielle / *Machine Learning*
— Physique

Applications

Pour donner à voir le plein potentiel de ces notions et comment elles peuvent être mises en œuvre, plusieurs applications poussées de ces notions sont présentées en fin d'ouvrage.

Il s'agira par exemple de voir comment la programmation peut aider à comprendre la physique sous sa forme variationnelle, ou encore coder l'une des méthodes de *Machine Learning* les plus fructueuses de ces dernières années : le *Gradient Boosting* pour les arbres de décisions.

Prérequis

Les diverses notions introduites dans ce livre le sont progressivement et ne requièrent pas de connaissances préalables au-delà du niveau baccalauréat en mathématique. L'utilisation systématique de code informatique pour illustrer chaque notion abordée simplifie grandement leur abord, en les démystifiant et en leur donnant corps.

Le chapitre suivant introduit les principales notions utiles pour comprendre le langage Python. C'est un langage simple d'abord, dont la prise en main est rapide, et qui permet naturellement d'exprimer des concepts mathématiques.

C'est par ailleurs un langage largement répandu dans le monde professionnel, et apprendre à le maîtriser ne peut être que bénéfique.

Remerciements

L'auteur tient à remercier en tout premier lieu ses parents, qui ont su développer son goût pour les sciences en général et les mathématiques en particulier, et qui lui ont permis de plonger dans le monde de l'informatique très tôt.

Il n'oublie pas non plus sa femme et ses deux filles, pour l'avoir encouragé dans cette expérience solitaire qu'est la rédaction d'un livre. Il espère que ses deux filles seront rapidement deux lectrices intéressées par cet ouvrage.

Enfin, que soit remerciée l'éditrice qui a bien voulu croire en ce projet et a permis sa réalisation.

1.2 Mathématique, programmation et langage

L'objet de ce livre est d'explorer les nombreux concepts mathématiques avec le support de la programmation, afin de disposer d'un laboratoire et des instruments utiles à cette exploration.

Langages de programmation

Depuis l'émergence de l'informatique dans les années 1950, de nombreux langages de programmation ont été développés. Le site *github*, l'une des deux grandes plateformes d'hébergement de code open source en compte plus d'une cinquantaine.

Parmi les plus répandus se trouvent Python, Javascript, Java, TypeScript, Go, C++, Rust, Lisp...

Domaines de prédilection

Chacun de ces langages a son domaine de prédilection, même si techniquement ils sont substituables l'un à l'autre. Le langage Python est par exemple beaucoup utilisé pour assurer les fonctionnalités *back-end* des applications web, tandis que Javascript se cantonne plutôt aux aspects interface web.

Python bénéficie aussi d'un écosystème très vivace dans le domaine de la Data Science.

Le Go et le C++, étant de plus bas niveau et offrant de bien meilleures performances en temps de calcul se retrouvent plus généralement dans des applications gourmandes en temps de calcul. Rust met lui l'accent sur la sécurité et notamment la gestion de la mémoire. Il s'est ainsi imposé dans le développement des applications systèmes, dont il est crucial d'assurer la fiabilité.

Critères de choix

Parmi ce large choix de langages, il a fallu en choisir un qui soit adapté à l'ambition de ce livre : plonger dans l'univers des mathématiques et en expérimenter les concepts les plus intéressants.

Simplicité de mise en œuvre

Le langage retenu devait satisfaire plusieurs contraintes, à commencer par une simplicité d'utilisation. En effet, cet ouvrage se destinant à un public pas forcément encore aguerri à la programmation, le langage ne devait pas être un frein à la compréhension du propos.

Math friendly

Il devait aussi être *math friendly*, c'est-à-dire offrir nativement la possibilité de manipuler simplement les constituants de base des mathématiques : nombres, fonctions, ensemble... Les librairies disponibles sur étagère devaient aussi être nombreuses dans le domaine des mathématiques.

Interactivité

Pour permettre de manipuler aisément les divers notions et objets mathématiques, il était indispensable de s'appuyer sur un langage doté d'une REPL (*Read Eval Print Loop*), c'est-à-dire d'une console interactive. Cette dernière permet d'interagir dynamiquement avec le code. Tous les langages n'en disposent pas, et notamment les langages compilés comme le C++ ou le Go.

Capitalisation

Enfin, il était important que ce langage soit utile au lecteur et que les compétences acquises en mathématiques et en programmation ici puissent être transposées immédiatement dans d'autres domaines.

Pourquoi Python ?

Peu de langages répondent finalement à ces exigences. Afin d'incarner les divers concepts mathématiques abordés dans ce livre, le langage retenu est donc le Python.

Plusieurs raisons ont gouverné ce choix :

— La simplicité du langage : Python dispose d'une syntaxe facilement assimilable et ne s'embarrasse pas avec le typage des objets.

— L'existence d'une console : cette dernière permet de manipuler de manière dynamique le code écrit en interagissant de manière interactive.

— L'existence d'un écosystème très vigoureux : de nombreuses librairies sont disponibles librement en ligne ce qui offre énormément de possibilités.

Les bases de Python

Afin de pouvoir tirer parti du formidable laboratoire expérimental que constitue la programmation pour les mathématiques, il faut au préalable maîtriser un langage de programmation.

La maîtrise d'un langage informatique peut sembler délicate à première vue, et cela peut-être le cas, suivant le langage étudié. Fort heureusement, le langage retenu pour cet ouvrage est unanimement reconnu comme étant l'un des plus simples à maîtriser.

Cette simplicité découle de plusieurs points :

— la syntaxe est simple : en Python, pas de parenthèse, ni d'accolades. Les blocs de code sont identifiés visuellement suivant leur indentation.

— Le niveau d'abstraction vis-à-vis de la machine est important : inutile en Python de se soucier de l'allocation de la mémoire, de sa libération... Tout est fait automatiquement.

— La richesse des types natifs : entier, flottant, chaîne de caractères, liste, ensemble, dictionnaire... beaucoup d'objets informatiques sont supportés nativement.

Afin de permettre au lecteur de se familiariser avec le langage Python ou de rafraîchir ses souvenirs, les sections suivantes reviennent sur les bases de ce langage.

Les types

Cette rapide présentation du langage Python commence par la présentation des principaux types. Dans un langage de programmation, un type définit un objet particulier qui va pouvoir être manipulé par le langage. *A minima*, presque tous les langages de programmation supportent deux types : les entiers et les nombres réels.

Python est beaucoup plus complet et étoffe cette liste avec les booléens, les chaînes de caractères, les listes, les ensembles ou encore les dictionnaires. Le listing ci-dessous explique comment créer chacun de ces types natifs.

Principaux types natifs en Python

```
# a est un entier
a = 12
print(type(a))
#> <class 'int'>
```