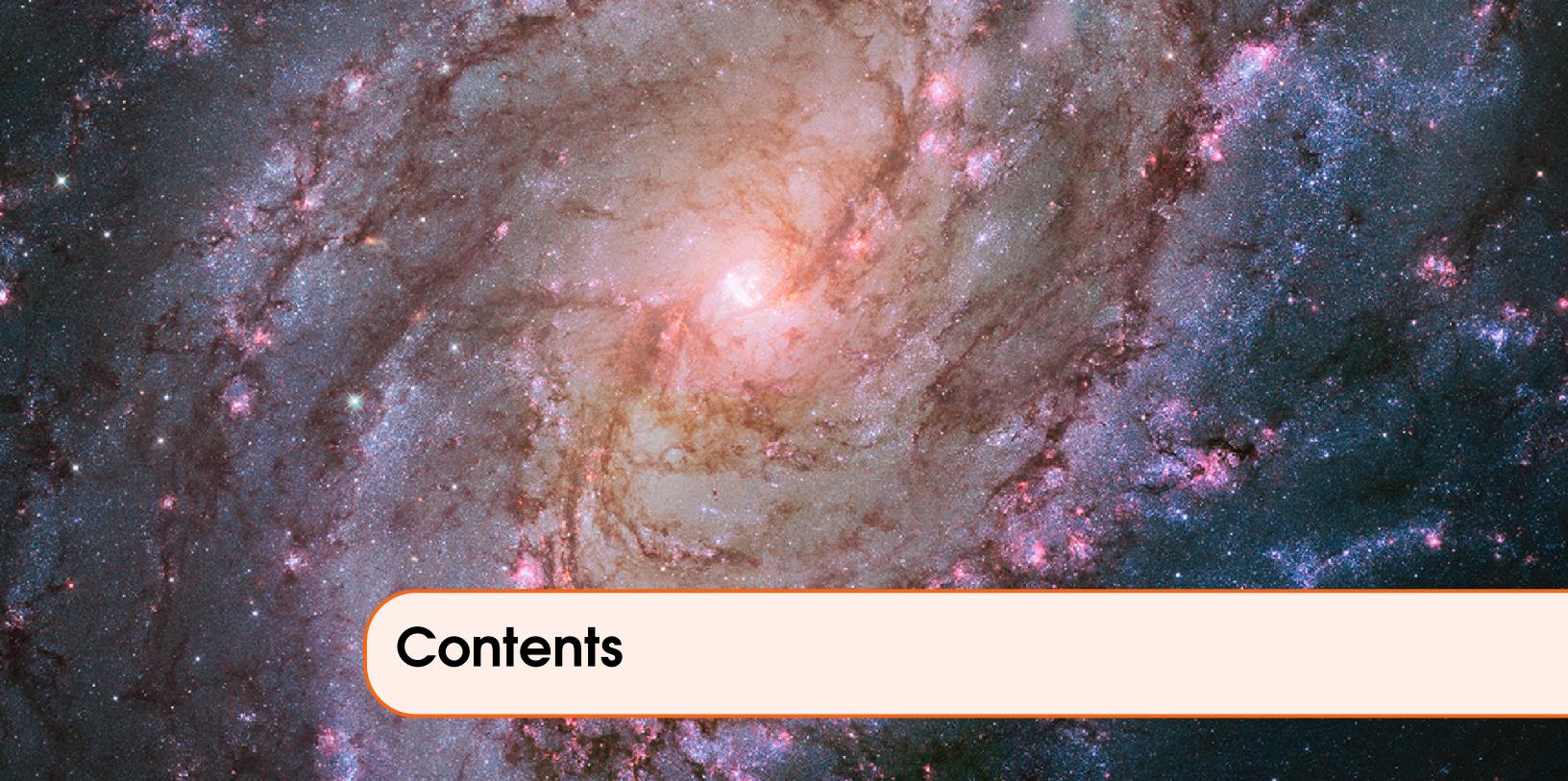


SymPy par la pratique

Exemple et exercice avancée

K.I.A Derouiche



Contents

0.1 Avant-Propos

Ce livre traite de SymPy, une bibliothèque de calcul symbolique entièrement écrite en Python un langage de programmation de haut niveau, orienté objet, totalement libre, conçu pour produire du code de qualité, portable et facile à intégrer. Ainsi la conception d'un programme scientifique ou symbolique avec SymPy et Python est très rapide et offre au développeur une bonne productivité. En tant que bibliothèque pythonienne elle repose sur un langage dynamique, très souple d'utilisation et constitue un complément idéal à des langages compilés. Elle reste une bibliothèque complète et autosuffisant, pour des petits scripts fonctionnels de quelques lignes, comme pour des applicatifs complexes de plusieurs centaines de modules.

Pourquoi ce livre ?

Il n'existe pas beaucoup d'ouvrages qui traitent du calcul symbolique en générale par rapport aux calculs numériques ou des ouvrages consacrés aux bibliothèques symboliques écrites en Python. En particulier, gravitent autour de SymPy mis à part un livre de 50 pages, quelques chapitres ou des lignes de codes cités à titre d'exemples. Citons le livre de référence de Svein Linge et Hans Petter Langtangen *Programming for Computations – Python A Gentle Introduction to Numerical Simulations with Python*, aux éditions Springer, ou encore une version du livre de 50 pages *Instant SymPy Starter* de Ronan Lamy, aux éditions Packt Publishing Limited. Le livre est *Instant SymPy Starter* de Ronan Lamy, c'est un guide de démarrage rapide. La documentation en ligne de SymPy est bonne, mais il serait plus facile de commencer avec ce livre. Alors, pourquoi ce livre ?

Si ce livre présente comme celui de Ronan Lamy les notions de la bibliothèque, celui-ci ajoute des exemples originaux, des choix dans la présentation des classes, et une approche globale particulière et détaillée, il tente également d'ajouter à ce socle des éléments qui participent de la philosophie de la programmation en Python scientifique, aller plus loin dans le développement non scientifique, mettre en valeur l'intérêt et l'importance, à savoir :

- des conventions de codage ;
- combiné l'approche symbolique et numérique ;
- des bonnes pratiques de programmation et des techniques d'optimisation ;

Même si chacun de ces sujets pourrait à lui seul donner matière à des ouvrages entiers, les réunir dans un seul et même livre contribue à fournir une vue complète de ce qu'un développeur d'application scientifique en particulier et Python averti et son chef de projet mettent en œuvre quotidiennement.

A qui s'adresse l'ouvrage?

Cet ouvrage s'adresse bien sûr aux développeurs de tous horizons mais également aux étudiants, chercheurs, enseignants et chefs de projets. Ils ne trouveront pas dans ce livre de bases de programmation; une pratique minimale préalable est indispensable de Python, quel que soit le langage utilisé. Il n'est pour autant pas nécessaire de maîtriser la programmation orientée objet et la connaissance d'un langage impératif est suffisante. Les développeurs Python débutants – ou les développeurs avertis ne connaissant pas encore cette bibliothèque – trouveront dans cet ouvrage des techniques et sujets avancés, les patterns efficaces et l'application de certains design patterns objet, topologie, théorie des catégories, machine learning. Les étudiants et enseignants trouveront un ouvrage ouvert sur l'apprentissage par l'exercice résolus et une interprétation d'exercices mathématiques les chercheurs trouveront un outil léger et efficace à travers des approches poussées liées aux questions récentes en connections avec les mathématiques pures, appliquées et la physique.

théorique. Les chefs de projets trouveront des éléments pratiques pour augmenter l'efficacité de leurs équipes pluridisciplinaires, notamment la présentation des principaux modules à la fois issues de la bibliothèque standard, graphique et numérique.

Indication

Quand un théorème est exposé ce dernier sera démontrai

0.2 Calcul formel

Le calcul formel, ou parfois calcul symbolique, est le domaine des mathématiques et de informatique qui s'intéresse aux algorithmes opérant sur des objets de nature mathématique par le biais de représentations finies et exactes. Ainsi, un nombre entier est représenté de manière finie et exacte par la suite des chiffres de son écriture en base 2. Étant données les représentations de deux nombres entiers, le calcul formel se pose par exemple la question de calculer celle de leur produit.

Le calcul formel est en général considéré comme un domaine distinct du calcul scientifique, cette dernière appellation faisant référence au calcul numérique approché à l'aide de nombres en virgule flottante, là où le calcul formel met l'accent sur les calculs exacts sur des expressions pouvant contenir des variables ou des nombres en précision arbitraire (en). Comme exemples d'opérations de calcul formel, on peut citer le calcul de dérivées ou de primitives, la simplification d'expressions, la décomposition en facteurs irréductibles de polynômes, la mise sous formes normales de matrices, ou encore la résolution des systèmes polynomiaux.

Sur le plan théorique, on s'attache en calcul formel à donner des algorithmes avec la démonstration qu'ils terminent en temps fini et la démonstration que le résultat est bien la représentation d'un objet mathématique défini préalablement. Autant que possible, on essaie de plus d'estimer la complexité des algorithmes que l'on décrit, c'est-à-dire le nombre total d'opérations élémentaires qu'ils effectuent. Cela permet d'avoir une idée a priori du temps d'exécution d'un algorithme, de comparer l'efficacité théorique de différents algorithmes ou encore éclairer la nature même du problème.

0.2.1 Logiciel de système de calcul formel

Dans cette section en va exposer les systèmes de calcul formel, leur intérêt qui à vue un renouveau ces dernières années à cause de l'émergence de technique, technologie et nouvelle approche de programmation pour le domaine scientifique et industriel, hormis le fait que le logiciel de calcul formel en soient sont un outil pédagogique indispensable pour les scientifiques et les ingénieurs

Definition 0.2.1 Un logiciel de système formel est un outil qui facilite le calcul symbolique. La partie principale de ce système est la manipulation des expressions mathématiques sous leur forme symbolique.

■ **Example 0.1** soit $G = \{x \in \mathbb{R}^2 : |x| < 3\}$ et noté par: $x^0 = (1, 1)$; en considère la fonction:

$$f(x) = \begin{cases} e^{|x|} & \text{si } |x - x^0| \leq 1/2 \\ 0 & \text{si } |x - x^0| > 1/2 \end{cases} \quad (1)$$

The function f has bounded support, we can take $A = \{x \in \mathbb{R}^2 : |x - x^0| \leq 1/2 + \varepsilon\}$ for all $\varepsilon \in]0; 5/2 - \sqrt{2}[$. ■

cet exemple se traduit en forme symbolique avec la bibliothèque SymPy:

0.2.2 Quelques logiciels de calcul formel

qui exprime ce qui nous permet notre choix pour un CAS qui possède des caractéristiques techniques et sur le plan du coût très important quand peut résumer dans les points suivants:

1. Leger et
2. S'appuie sur le langage de programmation Python
3. Portabilité dans toute transparence

L'un des systèmes qui peut nous permettre d'écrire cette exemple avec un ordinateurs avec SymPy qui semble mieux intégré

0.2.3 Pourquoi choisir SymPy?

Part I

Premier pas vers SymPy

1. Premier pas vers SymPy

Ce chapitre d'introduction présente la tournure d'esprit de la bibliothèque mathématique SymPy. Les autres chapitres de cette partie développent les notions de base de SymPy: effectuer des calculs numériques ou symboliques en analyse, opérer sur des vecteurs et des matrices, écrire des programmes, manipuler des listes de données, construire des graphiques, etc. Les parties suivantes de cet ouvrage approfondissent quelques branches des mathématiques dans lesquelles l'informatique fait preuve d'une grande efficacité.

1.1 La bibliothèque SymPy

1.1.1 Le cas de la bibliothèque SymPy

Dans un cas plus simple l'exemple 1.1 se formule beaucoup plus dans un outil comme SymPy est une bibliothèque de calcul formel elle est aussi un environnement pour l'apprentissage de l'algèbre, l'analyse, géométrie, combinatoire, cryptographie, mécanique classique et quantique pour le lycée et l'université mais aussi un environnement de développement et de recherche. SymPy écrit entièrement en Python un langage de programmation facile à apprendre et adapté à l'apprentissage, elle fournit aux étudiant *SymPyGamma* une application web notamment des primitives générales de traitement des expressions algébriques (développement, factorisation, ...), des aides à l'organisation des objets mathématiques intervenant dans la résolution d'un problème ainsi qu'une assistance à la preuve. Il permet au professeur de préparer et de suivre le travail de l'élève. Différentes maquettes ont été développées et testées auprès d'élèves. Dans la plus récente, nous nous sommes attachés à explorer une nouvelle forme d'activité algébrique. Alors que le calcul en papier crayon et les logiciels standards considèrent les expressions de façon isolée, l'environnement que nous développons organise en réseau les différentes expressions intervenant dans la résolution d'un problème. L'ordinateur peut facilement mettre à jour ce réseau quand l'utilisateur modifie certains de ses éléments. Il devient ainsi possible, pour aborder un problème générique, d'explorer facilement des cas particuliers et de conduire une généralisation. Les relations entre expressions algébriques sont mieux mises en évidence du fait de leur invariance dans les modifications du réseau. De façon très concise, Casyopée peut être défini

1.1.2 Travaillez avec SymPy

1.1.3 Installation de SymPy

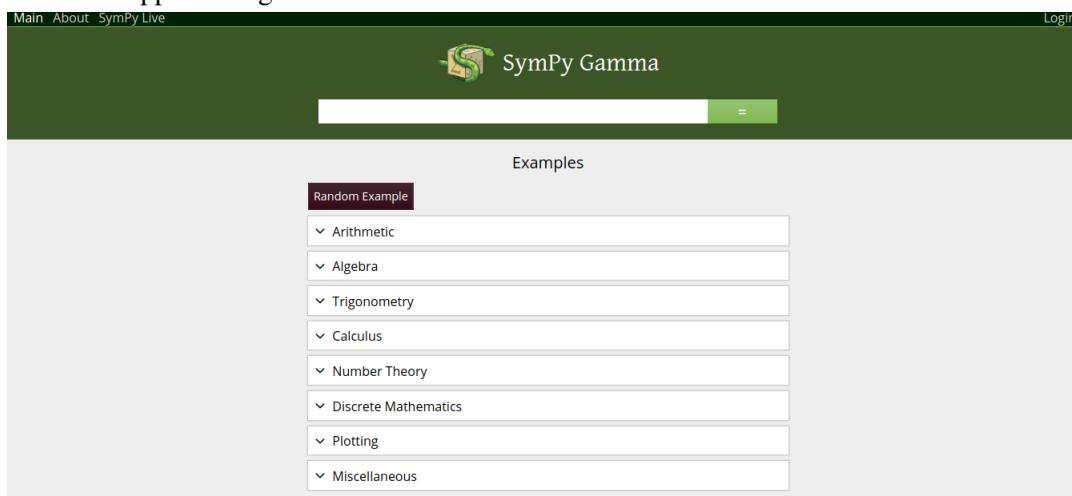
1.1.4 isympy

```
IPython console for SymPy 1.3 (Python 3.6.7-64-bit) (ground types: python)
These commands were executed:
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
>>> init_printing()
```

Documentation can be found at <http://docs.sympy.org/1.3/>

1.1.5 SymPyGamma

Est une interface onWeb marche avec un navigateur contient plusieurs catégorie liée de calcul, dynamique. L'Intérêt de cette outil qu'il est facilement partageable adapté pour l'enseignement et surtout l'auto-apprentissage



1.1.6 SymPyLive

SymPy Live est SymPy qui s'exécute sur Google App Engine. Ceci est juste un shell Python standard, avec les commandes suivantes exécutées par défaut

1.2 SymPy comme calculatrice

Contrairement à Sage[], Maple, Octave et les autres logiciel de calcul formel, SymPy, effectue des calculs directe numériques de deux manière différents en passant par la méthode,

1.2.1 Premier calcul

Dans la suite du livre, nous présentons les calculs sous la forme suivante, qui imite l'allure d'une session de SymPyLive à travers la ligne de commande isympy:

```
In [1]: from sympy import *
In [2]: 1+1
Out[2]: 2
```

Variables Python

Lorsque l'on veut conserver le résultat d'un calcul, on peut l'affecter à une variable :

Variables Symboliques

Les objets mathématiques manipulés par SymPy sont symboliques ils sont représentés exactement loin de toute approximation numérique, SymPy permet une manipulation avec des expressions contenant des variables, comme $x^2 + zy^3 + z^2$ ou encore $\sin(x) - \exp(x)$. Les variables symboliques du mathématicien x, y, z apparaissant dans ces expressions diffèrent, avec SymPy, des variables du développeur $\sin(2) = 0.9092974268256817$ que nous manipulons sous Python section précédente. SymPy diffère notamment, sur ce point, d'autres systèmes de calcul formel comme Maple ou Maxima, Sage c'est inspiré de SymPy sur ce point.

La documentation officiel présente la différence entre valeur numérique gérer par la bibliothèque standard Python math à travers l'exemple de la racine carré $\sqrt{8}$ sans évaluation, posons $x = 8$

```
In [1]: import math
In [2]: math.sqrt(x)
Out[2]: 2.8284271247461903
```

Les variables symboliques doivent être explicitement déclarées avant d'être employées

Dans cet exemple, la commande SR.var('z') « construit » et renvoie une variable symbolique dont le nom est z. Cette variable symbolique est un objet Sage comme un autre : elle n'est pas traitée différemment d'expressions plus complexes comme $\sin(x) + 1$. Ensuite, cette variable symbolique est affectée à la variable « du programmeur » z, ce qui permet de s'en servir comme de n'importe quelle expression pour construire des expressions plus complexes.

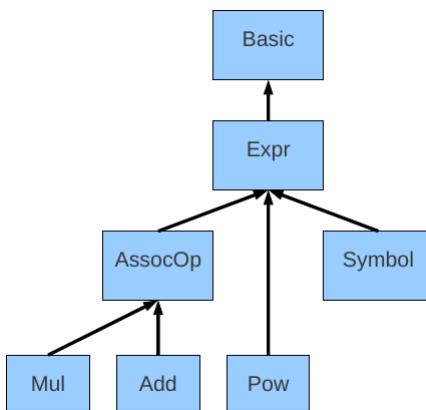
```
In [1]: from sympy import *
In [2]: x = symbols('x')
In [3]: type(x)
Out[3]: sympy.core.symbol.Symbol
```

```
In [4]: x+3
Out[4]: x + 3
```

1.2.2 Structure de données dans SymPy

Le moteur symbolique de SymPy tire parti de l'orientation des objets (notamment l'héritage) pour créer une base de code facilement extensible. Toutes les classes dérivent des fonctionnalités, telles que la possibilité de se comparer à d'autres objets, à partir de méthodes de la super-classe Basic. Les

objets pouvant faire l'objet d'opérations algébriques acquièrent cette capacité grâce à un ensemble de méthodes d'une classe appelée `Expr`. Ces objets `Expr` peuvent être conservés dans des objets conteneur (qui contiennent également la sous-classe `Expr`) `Mul`, `Add` et `Pow`; les objets conteneur sont instanciés à l'aide de l'opérateur Python, tel que la fonction de surcharge, qui permet au constructeur de la classe conteneur d'être appelé chaque fois que l'opérateur binaire approprié est utilisé (* pour `Mul`, + pour Ajouter et ** pour `Pow`). De cette manière, des objets supplémentaires peuvent être ajoutés en créant simplement une sous-classe qui hérite des fonctionnalités de la classe `Expr`. Ces sous-classes bénéficient gratuitement de certaines fonctionnalités, telles que la possibilité de comparer, de multiplier, d'ajouter, etc. Voici comment SymPy crée un environnement modifiable, maintenable, et donc facile à étendre. Grâce à la possibilité d'hériter des propriétés de classes supérieures, la quantité de code nécessaire pour développer, par exemple, un système modélisant la mécanique quantique et la notation Dirac décroissant de manière significative.



1.2.3 Variable et affectation

Exercise 1.1 Affectez les variables temps t et distance d par les valeurs 6.892 et 19.7. Calculez et affichez la valeur de la vitesse. Améliorez l'affichage en imposant un chiffre après le point décimal.

Solution 1.1 Pour, affectez des variables est les rendre symbolique comme c'est décrit dans le mémo ou il sera expliquer temps t et distance d par les valeurs 6.892 et 19.7. Calculez et affichez la valeur de la vitesse. Améliorez l'affichage en imposant un chiffre après le point décimal.

1.2.4 Substitution

Une des choses les plus courantes que vous pourriez vouloir faire avec une expression mathématique est la substitution. La substitution remplace toutes les occurrences de quelque chose dans une expression par quelque chose d'autre. SymPy utilisant la méthode `subs`.

1.2.5 Contrôle du flux d'instructions

This is a theorem consisting of just one line.

Exercise 1.2 A set $\mathcal{D}(G)$ in dense in $L^2(G)$, $|\cdot|_0$.

Solution 1.2



2. Analyse et Algèbre

Ce chapitre présente à travers des exemples simples les fonctions de base utiles en analyse et en algèbre. Les lycéens et étudiants trouveront matière à remplacer le crayon et le papier par le clavier et l'écran tout en relevant le même défi intellectuel de compréhension des mathématiques. Cet exposé des principales commandes de calcul avec Sage se veut accessible aux élèves de lycée.

2.1 Expressions symboliques et simplification

2.1.1 Expressions symboliques

La bibliothèque SymPy permet d'effectuer toutes sortes de calculs d'analyse à partir d'expressions symboliques combinant des nombres, des variables symboliques, les quatre opérations, et des fonctions usuelles comme `sqrt`, `exp`, `log`, `sin`, `cos`, etc. Une expression symbolique peut être représentée par un arbre comme ceux de la. Il est important de comprendre qu'une expression symbolique est une formule et non pas une valeur ou une fonction mathématique. Ainsi, SymPy ne reconnaît pas que les deux expressions suivantes sont égales

2.1.2 Transformation d'expressions

2.1.3 Fonctions mathématiques usuelles

La plupart des fonctions mathématiques se retrouvent dans SymPy, en particulier les fonctions trigonométriques, le logarithme et l'exponentiel : elles sont rassemblées dans le tableau

2.2 Équations

Ceci est la documentation officielle du module `solveset` dans les solveurs. Il contient les questions fréquemment posées sur notre nouveau module pour résoudre des équations.

2.2.1 Résolution explicite

2.2.2 Équations sans solution explicite

2.3 Inéquations

2.4 Analyse

Dans cette section, nous présentons succinctement les fonctions couramment utiles en analyse réelle. Pour une utilisation avancée ou des compléments, on renvoie aux chapitres suivants notamment ceux qui traitent de l'intégration numérique (ch. 14), de la résolution des équations non linéaires (ch. 12), et des équations différentielles (ch. 10)

2.4.1 Les Fonctions

Il sert également de constructeur pour les classes de fonctions non définies.

```
from sympy import Function, Symbol
```

Exercise 2.1 Écrire une fonction cube qui retourne le cube de son argument



Exercise 2.2 Écrire une fonction *volumeSphere* qui calcule le volume d'une sphère de rayon *r* fourni en argument et qui utilise la fonction cube . Tester la fonction *volumeSphere* par un appel dans le programme principal.



Exercise 2.3 Écrire une fonction maFonction qui retourne $f(x) = 2x^3 + x - 5$



Exercise 2.4 Écrire une fonction tabuler avec quatre paramètres : *fonction* , *borneInf* , *borneSup* et *nbPas* . Cette procédure affiche les valeurs de *fonction* , de *borneInf* à *borneSup* , tous les *nbPas* . Elle doit respecter *borneInf* < *borneSup*. Tester cette fonction par un appel dans le programme principal après avoir saisi les deux bornes dans une floatbox et le nombre de pas dans une integerbox (utilisez le module easyguiB).



Exercise 2.5 Écrire une fonction *volMasse* Ellipsoïde qui retourne le volume et la masse d'un ellipsoïde grâce à un tuple. Les paramètres sont les trois demi-axes et la masse volumique. On donnera à ces quatre paramètres des valeurs par défaut.

On donne: $v = \frac{3}{4}\pi abc$

Tester cette fonction par des appels avec différents nombres d'arguments.



Exercise 2.6 Une fonction $f(x)$ est linéaire et a une valeur de 29 à $x = -2$ et 39 à $x = 3$. Trouver sa valeur à $x = 5$.



Exercise 2.7 Pour l'ensemble N de nombres naturels et une opération binaire $f : NxN \rightarrow N$, on appelle un élément $z \in N$ une identité pour f , si $f(a, z) = a = f(z, a)$, pour tout $a \in N$. Lesquelles des opérations binaires suivantes ont une identité?:

1. $f(x, y) = x + y - 3$
2. $f(x, y) = \max(x, y)$
3. $f(x, y) = x^y$

Solution 2.1 le deuxième et le troisième

2.4.2 Sommes

2.4.3 Limites

2.4.4 Suites

2.4.5 Développements limités

2.4.6 Séries

On peut utiliser les commandes précédentes pour effectuer des calculs sur les séries. Donnons quelques exemples.

2.4.7 Dérivation

La fonction derivative (qui a pour alias diff) permet de dériver une expression symbolique ou une fonction symbolique

2.4.8 Dérivées partielles

La commande diff permet également de calculer des dérivées n-ièmes ou des dérivées partielles.

2.4.9 Intégration

2.5 Calcul matriciel

Dans cette section, on décrit les fonctions de base utiles en algèbre linéaire :opérations sur les vecteurs, puis sur les matrices. Pour plus de détails, on renvoie au chapitre 8 pour le calcul matriciel symbolique et au chapitre 13 pour le calcul matriciel numérique.

2.5.1 Résolution de systèmes linéaires

2.5.2 Calcul vectoriel

Les fonctions de base utiles pour la manipulation des vecteurs sont résumées dans le tableau 2.5. On peut se servir de ces fonctions pour traiter l'exercice suivant.

2.5.3 Calcul matriciel

2.5.4 Réduction d'une matrice carrée



3. Graphiques

La visualisation de fonctions d'une ou deux variables, d'une série de données, facilite la perception de phénomènes mathématiques ou physiques et permet de conjecturer des résultats efficacement. Dans ce chapitre, on illustre sur des exemples les capacités graphiques de SymPy.

3.1 Courbes en 2D

Une courbe plane peut être définie de plusieurs façons : comme graphe d'une fonction d'une variable, par un système d'équations paramétriques, par une équation en coordonnées polaires, ou par une équation implicite. Nous présentons ces quatre cas, puis donnons quelques exemples de tracés de données.

- 3.1.1 Représentation graphique de fonctions
- 3.1.2 Courbe paramétrée
- 3.1.3 Courbe en coordonnées polaires
- 3.1.4 Courbe définie par une équation implicite
- 3.1.5 Tracé de données
- 3.1.6 Tracé de solution d'équation différentielle
- 3.1.7 Développée d'une courbe
- 3.2 Courbes en 3D

Part II

Logique et ensemble

4. Logique

La bibliothèque fournit un module pour faire de la logique mathématique indépendamment vue comme extension des classes fournit par le builtins de Python.



5. Théorie des ensemble

5.1 Un petit rappel pour les ensembles dans Python

La théorie d'ensemble existe sous Python, il suffit simplement de tapez

```
ens = set()
```

ou écrire simplement

```
ens = {1, 2, 4}
```

5.2 Ensemble avec SymPy

La notion d'objet immuable en Python est fondamentale, une structure qui rappel les ensembles en mathématiques que soit fini ou infini est *set*, importante, bien que dans le cadre de SymPy elle s'appuie entièrement sur Python avec certain modification, avec la collection d'objet.

La fonction set accepte donc en argument un objet de type quelconque et s'efforce de le traduire dans un ensemble. Lorsqu'on ne passe aucun argument à set (option 2), ou qu'on lui passe une liste vide, set renvoie naturellement un ensemble vide; on aurait pu utiliser aussi bien, de la même manière, set(), set(), ou même set("") pour arriver au même résultat.

Exercise 5.1 Définir deux ensembles $X = \{a, b, c, d\}$ et $Y = \{s, b, d\}$, puis affichez les résultats suivants :

1. les ensembles initiaux.
2. le test d'appartenance de l'élément c à X .
3. le test d'appartenance de l'élément a à Y .
4. les ensembles $X - Y$ et $Y - X$.
5. l'ensemble $X \cup Y$ (union).
6. l'ensemble $X \cap Y$ (intersection).

Solution 5.1 Il faut noter qu'il existe une solution qui se base sur le Python builtins en utilisant la structure de donnée *sets*. Mais comme en n'est dans la logique en utilise

```
from sympy import FiniteSet

X = FiniteSet('a', 'b', 'c', 'd')
Y = FiniteSet('s', 'b', 'd')

class MyClass(Yourclass):
    def __init__(self, my, yours):
        bla = '5 1 2 3 4'
        print bla

class MyClass(Yourclass):
    def __init__(self, my, yours):
        bla = '5 1 2 3 4'
        print bla
```

5.2.1 Logique

Exercise 5.2 Dans la carte de Karnaugh ci-dessous, X indique un terme sans intérêt. Quelle est la forme minimale de la fonction représentée par la carte de Karnaugh?

5.2.2 Ensembles

La notion d'objet immuable en Python est fondamentale, une structure qui rappel les ensembles en mathématiques que soit fini ou infini est *set*, importante, bien que dans le cadre de SymPy elle s'appuie entièrement sur Python avec certain modification, avec la collection d'objet.

La fonction set accepte donc en argument un objet de type quelconque et s'efforce de le traduire dans un ensemble. Lorsqu'on ne passe aucun argument à set (option 2), ou qu'on lui passe une liste vide, set renvoie naturellement un ensemble vide; on aurait pu utiliser aussi bien, de la même manière, set(()), set(), ou même set("") pour arriver au même résultat.

Exercise 5.3 Définir deux ensembles $X = \{a, b, c, d\}$ et $Y = \{s, b, d\}$, puis affichez les résultats suivants :

1. les ensembles initiaux.
2. le test d'appartenance de l'élément c à X .
3. le test d'appartenance de l'élément a à Y .
4. les ensembles $X - Y$ et $Y - X$.
5. l'ensemble $X \cup Y$ (union).
6. l'ensemble $X \cap Y$ (intersection).

Solution 5.2 Il faut noter qu'il existe une solution qui se base sur le Python builtins en utilisant la structure de donnée *sets*. Mais comme en n'est dans la logique en utilise

```
from sympy import FiniteSet

X = FiniteSet('a', 'b', 'c', 'd')
Y = FiniteSet('s', 'b', 'd')

class MyClass(Yourclass):
    def __init__(self, my, yours):
        bla = '5 1 2 3 4'
        print bla

class MyClass(Yourclass):
    def __init__(self, my, yours):
        bla = '5 1 2 3 4'
        print bla
```

Part III

Algèbre et théorie des nombres

6. Anneaux et corps finis

6.1 Anneau des entiers modulo n

6.2 Corps finis

6.3 Quelques problèmes élémentaires de théorie des nombres

6.3.1 Théorème de Wilson

¹ énoncé du théorème

Theorem 6.3.1 Un entier p strictement plus grand que 1 est un nombre premier si et seulement s'il divise $(p-1)! + 1$, c'est-à-dire si et seulement si $(p-1)! + 1 \equiv 0 \pmod{p}$

Indication. Quatre démonstrations de ce résultat de Wilson. L'idée directrice des deux premières démonstrations est de remplacer ce calcul de congruence $(\mathbb{Z}/p\mathbb{Z})$ par un calcul dans \mathbb{F}_p ce qui va permettre d'utiliser les propriétés d'un corps. L'idée de la troisième démonstration est d'utiliser les théorèmes de Sylow dans le groupe symétrique σ_p , la quatrième est plutôt combinatoire qui repose sur l'identité algébrique $\sum_{i=0}^n (-1)^i C_n^p (x-i)^n = n!$ donnée en théorème l'auteur² ainsi le théorème sera simplement comme corollaire en remplaçant n par $p-1$

Proof. A venir! ■

Alain Connes dans son article "Autour du théorème de Wilson"³, donne une approximation du nombre π en somme de \sin

¹**Note Historique.** Le théorème de Wilson a été découvert à la fin du dixième siècle par le mathématicien arabe Ibn al-Haytham (965 – 1040). Le résultat ressurgit, sans démonstration, à la fin du dix-huitième siècle dans les écrits de Edward Waring qui l'attribue en 1770 à son élève John Wilson. L'année suivante, Lagrange en donne deux démonstrations dans son article [LAG]. En fait, Leibniz (1646 – 1716) connaissait déjà le résultat et sa démonstration mais ne les avait pas publiés (voir [RAS] pour de plus amples considérations historiques).

²<https://arxiv.org/pdf/math/0406086.pdf>

³Alain Connes, An essay on the Riemann Hypothesis, Open Problems in Mathematics. John Forbes Nash, Jr. Michael Th. Rassias Editors

R Contrairement au petit théorème de Fermat, le théorème de Wilson est une condition nécessaire et suffisante pour tester la primalité. Toutefois, cela conduirait à un test très lent informatiquement, car le calcul de $(p-1)!$ nécessite beaucoup d'opérations.

```
import sympy

def isPrime(n):
    if n == 4: return
    return bool(math.factorial(n>>1)%n)
```

7. Polynômes

7.1 Anneaux et polynômes

7.1.1 Introduction

Nous avons vu au chapitre 2 comment effectuer des calculs sur des expressions formelles, éléments de « l’anneau symbolique ». Dans ce chapitre en va manipuler le module dédié, `sympy.polys`, permettant de calculer des algèbres polynomiales sur divers domaines de coefficients implémentant un grand nombre de méthodes allant d’outils simples comme la division polynomiale à des concepts avancés comprenant les bases de Gröbner et la factorisation multivariée sur des domaines de nombres algébrique:

7.1.2 Construction d’anneaux de polynômes

En SymPy, les polynômes, comme beaucoup d’autres objets algébriques, sont en général à coefficients dans un anneau commutatif. C’est le point de vue que nous adoptons, mais la plupart de nos exemples concernent des polynômes sur un corps. Dans tout le chapitre, les lettres A et K désignent respectivement un anneau commutatif et un corps quelconques. La première étape pour mener un calcul dans une structure algébrique est souvent de construire R elle-même. On construit $\mathbb{Q}[x]$

7.2 Polynômes

7.2.1 Création et arithmétique de base

7.2.2 Vue d’ensemble des opérations sur les polynômes

7.2.3 Changement d’anneau

¹ Changement d’anneau. La liste exacte des opérations disponibles, leur effet et leur efficacité dépendent fortement de l’anneau de base. Par exemple, les polynômes de $\mathbb{Z}[x]$ possèdent une méthode `content` qui renvoie leur contenu, c’est-à-dire le pgcd de leurs coefficients ; ceux de $\mathbb{Q}[x]$

¹Contrairement à Sage, il peut y être que dans SymPy certain fonctionnalité ne soit pas directement accessible que par passage à la programmation

non, l'opération étant triviale. La méthode `factor` existe quant à elle pour tous les polynômes mais déclenche une exception `NotImplementedError` pour un polynôme à coefficients dans `SR`(le cas de Sage) ou dans $\mathbb{Z}/4\mathbb{Z}$. Par exemple Cette exception signifie que l'opération n'est pas disponible dans Sage pour ce type d'objet bien qu'elle ait un sens mathématiquement. Il est donc très utile de pouvoir jongler avec les différents anneaux de coefficients sur lesquels on peut considérer un « même » polynôme. Appliquée à un polynôme de $A[x]$, la méthode `change_ring` renvoie son image dans $B[x]$, quand il y a une façon naturelle de convertir les coefficients. La conversion est souvent donnée par un morphisme canonique de A dans B : notamment, `change_ring` sert à étendre l'anneau de base pour disposer de propriétés algébriques supplémentaires. Ici par exemple, le polynôme p est irréductible sur les entiers, mais se factorise sur R :

7.2.4 Itération

7.3 Arithmétique euclidienne

7.3.1 Divisibilité

7.3.2 Idéaux et quotients

7.3.3 Idéaux

7.4 Factorisation et racines

7.4.1 Factorisation

7.4.2 Recherche de racines

7.4.3 Résultant

7.4.4 Groupe de Galois

Par défaut le calcul de groupe de Galois n'est pas disponible dans SymPy, ce qui nous amène encore une fois de programmer en ajoutant des modules. Le groupe de Galois d'un polynôme irréductible $p \in \mathbb{Q}[x]$ est un objet algébrique qui décrit certaines « symétries » des racines de p . Il s'agit d'un objet central de la théorie des équations algébriques. Notamment, l'équation $p(x) = 0$ est résoluble par radicaux, c'est-à-dire que ses solutions s'expriment à partir des coefficients de p au moyen des quatre opérations et de l'extraction de racine n -ième, si et seulement si le groupe de Galois de p est résoluble.

7.5 Fractions rationnelles

7.5.1 Construction et propriétés élémentaires

La division de deux polynômes (sur un anneau intègre) produit une fraction rationnelle. Son parent est le corps des fractions de l'anneau de polynômes, qui peut s'obtenir par `Frac(R)` :

7.6 Séries formelles

Une série formelle est une série enGroupes de matrices.tière vue comme une simple suite de coefficients, sans considération de convergence. Plus précisément, si A est un anneau commutatif, on appelle séries formelles (en anglais formal power series) d'indéterminée à coefficients dans les sommes formelles $\sum_{n=0}^{\infty} a_n x^n$ où (a_n) est une suite quelconque d'éléments de A . Munies des opérations d'addition et de multiplication naturelles

$$\sum_{n=0}^{\infty} a_n x^n + \sum_{n=0}^{\infty} b_n x^n = \sum_{n=0}^{\infty} (a_n + b_n) x^n$$

,

$$\left(\sum_{n=0}^{\infty} a_n x^n \right) \left(\sum_{n=0}^{\infty} b_n x^n \right) = \sum_{n=0}^{\infty} \left(\sum_{i+j=n} a_i b_j \right) x^n$$

, les séries formelles forment un anneau noté $A[[x]]$.

les séries formelles forment un anneau noté Dans un système de calcul formel, ces séries sont utiles pour représenter des fonctions analytiques dont on n'a pas d'écriture exacte. Comme toujours, l'ordinateur fait les calculs, mais c'est à l'utilisateur de leur donner un sens mathématique. À lui par exemple de s'assurer que les séries qu'il manipule sont convergentes.

7.6.1 Opérations sur les séries tronquées

7.6.2 Développement de solutions d'équations

Face à une équation différentielle dont les solutions exactes sont trop compliquées à calculer ou à exploiter une fois calculées, ou tout simplement qui n'admet pas de solution en forme close, un recours fréquent consiste à chercher des solutions sous forme de séries. On commence habituellement par déterminer les solutions de l'équation dans l'espace des séries formelles, et si nécessaire, on conclut ensuite par un argument de convergence que les solutions formelles construites ont un sens analytique. SymPy peut être d'une aide précieuse pour la première étape. Considérons par exemple l'équation différentielle

■ **Example 7.1**

$$(x) = \sqrt{1+x^2}$$

Exercise 7.1 Considérons le polynôme $p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$, où $a_i \neq 0 \forall i$. Le nombre minimum de multiplications nécessaires pour évaluer p sur une entrée x est:

Ce chapitre traite de l'algèbre linéaire exacte et symbolique, c'est-à-dire sur des anneaux propres au calcul formel, tels que \mathbb{Z} , des corps finis, des anneaux de polynômes. Nous présentons les constructions sur les matrices et leurs espaces ainsi que les opérations de base, puis les différents calculs possibles sur ces matrices, regroupés en deux thèmes : ceux liés à l'élimination de Gauss et aux transformations par équivalence à gauche, et ceux liés aux valeurs et espaces propres et aux transformations de similitude.

8.1 Constructions et manipulations élémentaires

8.1.1 Espaces de vecteurs, de matrices

De la même façon que pour les polynômes, les vecteurs et les matrices sont manipulés comme des objets algébriques appartenant à un espace. Si les coefficients appartiennent à un corps K , c'est un espace vectoriel sur K ; s'ils appartiennent à un anneau, c'est un K -module libre. **Groupes de matrices.** On pourra par ailleur définir des sous-groupes de l'espace total des matrices. Ainsi le constructeur

8.1.2 Construction des matrices et des vecteurs

Les matrices et les vecteurs peuvent naturellement être générés comme des éléments d'un espace en fournissant la liste des coefficients en arguments. Pour les matrices, ceux-ci seront lus par ligne :

8.1.3 Manipulations de base et arithmétique sur les matrices

Indices et accès aux coefficients. L'accès aux coefficients ainsi qu'à des sous-matrices extraites se fait de façon unifiée par l'opérateur crochet $A[i, j]$, selon les conventions usuelles de Python. Les indices de ligne i et de colonne j peuvent être des entiers (pour l'accès à des coefficients) ou des intervalles sous la forme $1 : 3$ (on rappelle que par convention, en Python les indices commencent à 0, et les intervalles sont toujours inclusifs pour la borne inférieure et exclusifs pour la borne supérieure). L'intervalle « $:$ » sans bornes correspond à la totalité des indices possibles dans la