

SymPy par la pratique(DRAFT(wip))

Exemple et exercice

K.I.A Derouiche

GITHUB.COM/KIADEROUICHE/MISC



Contents

0.1	Avant-Propos	6
0.2	Introduction	7
0.2.1	Pourquoi programmer en symbolique	7
1	Calcul formel	9
1.1	Un peu de théorie	10
1.2	Logiciel de système de calcul formel	10
1.2.1	Quelques logiciels de calcul formel	11
1.3	Bibliothèque SymPy	11
1.3.1	SymPyGamma	11
1.3.2	Besoin de rester dans le symbolique	11
1.3.3	Passage du symbolique au numérique	11
1.3.4	Faire des dessins	11
2	Ensemble	13
2.1	Ensemble	13
2.1.1	Ensembles	13
2.2	Premiers pas en Python symbolique	14
2.2.1	Variable et affectation	14
2.2.2	Contrôle du flux d'instructions	14
2.3	Les Fonctions	14
2.4	Structures de données	15
2.4.1	Logique	15
2.4.2	Ensembles	15

2.4.3	Polynômes	16
2.4.4	Nombres parfaits et nombres chanceux	16
2.5	Remarks	16
2.6	Corollaries	17
2.7	Propositions	17
2.7.1	Several equations	17
2.7.2	Single Line	17
2.8	Examples	17
2.8.1	Equation and Text	17
2.8.2	Paragraph of Text	17
2.9	Exercises	18
2.10	Problems	18
2.11	Vocabulary	18
3	POO	19
3.1	Programmation Orientée Objet	19
3.1.1	POO	19
3.1.2	Notions de COO et d'encapsulation	19
4	Convex Sets	21
4.1	Convexity	21
4.1.1	Cone	21
4.2	Convex Functions	22
4.2.1	Epigraph	22
4.3	Support Function	24
4.4	Operations Preserve Convexity of Functions	25
5	Nonlinear Problem	27
5.1	Équation différentielle	27
5.2	Chaos	27
5.2.1	Pendule simple	27
5.2.2	Pendule à deux bras	27
5.2.3	Mouvements d'un robot	27
5.3	Solution non linéaire d'équation algébrique	27
5.4	La théorie de catégorie	28
5.5	Mécanique quantique	28
5.6	Transport optimal	28

5.7	Le calcul des variations	28
5.8	Figure	28
6	Outils avancée	29
6.1	Programmation Orientée Objet	29
6.2	Décorateurs	29
6.2.1	Optimisation du code	29
6.2.2	Cython	30
6.2.3	Theano	31
6.3	Interface graphique	32

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 et terriblement efficace, conçu pour produire du code de qualité, portable et facile à intégrer. Ainsi la conception d'un programme Python est très rapide et offre au développeur une bonne productivité. En tant que langage dynamique, il est très souple d'utilisation et constitue un complément idéal à des langages compilés. Il reste un langage complet et autosuffisant, pour des petits scripts fonctionnels de quelques lignes, comme pour des applicatifs complexes de plusieurs centaines de modules.

Pourquoi ce livre ?

Il existe déjà de nombreux ouvrages excellents traduits de l'anglais qui traitent de Python voire en présentent intégralité des modules disponibles. Citons Python en concentré, le manuel de référence de Mark Lutz et David Ascher, aux éditions O'Reilly, ou encore Apprendre à programmer avec Python de Gérard Swinnen, aux éditions Eyrolles, inspiré en partie du texte How to think like a computer scientist (Downey, Elkner, Meyers), et comme son titre l'indique, très pédagogique. Alors, pourquoi ce livre ?

A qui s'adresse l'ouvrage?

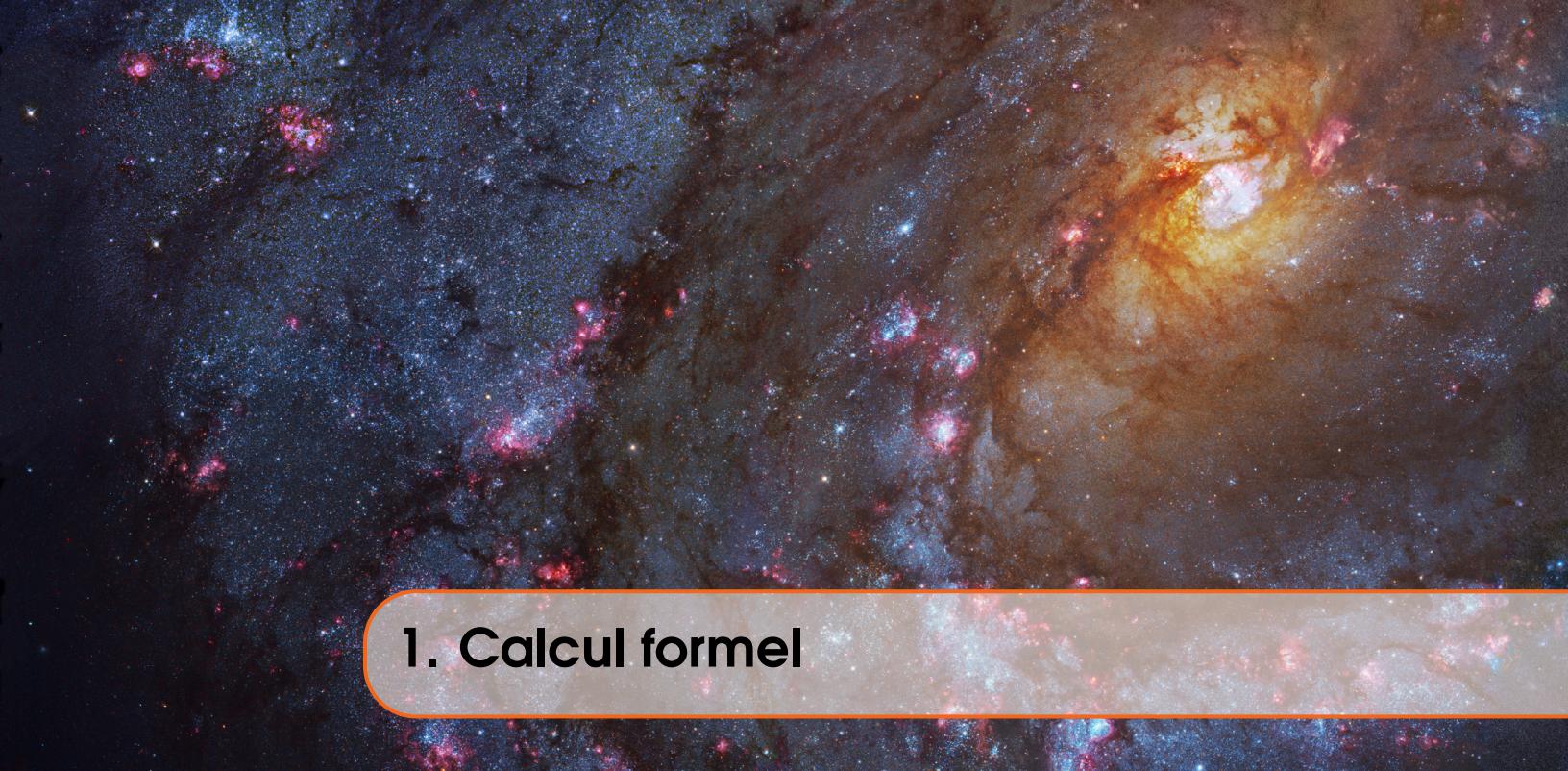
0.2 **Introduction**

Ce recueil d'exercices et de problèmes de programmation s'adresse aussi bien aux débutants qu'aux programmeurs confirmés. Il présente en effet plusieurs états d'esprit dont les deux principaux sont la programmation classique en Pascal pour les étudiants du premier cycle universitaire, et la programmation fonctionnelle en Lisp pour le second cycle.

Ce livre constitue un panorama (non exhaustif, mais suffisant) sur les langages de programmation, et offre une grande variété dans les sujets traités : graphiques, calcul matriciel, traitements de chaînes de caractères, graphes, intelligence artificielle...

La première partie du livre sera consacré à la résolution par une approche symbolique au divers questions posées au étudiants et toute personnes qui aiment savoir et voir s'initier pour des niveaux et des questions rencontrés, la deuxième partie du livre sera questions aux problèmes plus rencontrés pour des étudiants passionnée des questions entre mathématiques et technologies, chercheurs et développeurs d'applications scientifiques, la troisième partie plus consacré aux questions poussées

0.2.1 **Pourquoi programmer en symbolique**



1. Calcul formel

L'approche La simulation numérique est devenue essentielle dans de nombreux domaines tels que la mécanique des fluides et des solides, la météo, l'évolution du climat, la biologie ou les semi-conducteurs. Elle permet de comprendre, de prévoir, d'accéder là où les instruments de mesures s'arrêtent.

Ce livre présente des méthodes performantes du calcul scientifique : matrices creuses, résolution efficace des grands systèmes linéaires, ainsi que de nombreuses applications à la résolution par éléments finis et différences finies. Alternant algorithmes et applications, les programmes sont directement présentés en langage C++. Ils sont sous forme concise et claire, et utilisent largement les notions de classe et de généralité du langage C++.

Le contenu de ce livre a fait l'objet de cours de troisième année à l'école nationale supérieure d'informatique et de mathématiques appliquées de Grenoble (ENSIMAG) ainsi qu'au mystère de mathématiques appliquées de l'université Joseph Fourier. Des connaissances de base d'algèbre matricielle et de programmation sont recommandées. La maîtrise du contenu de cet ouvrage permet d'appréhender les principaux paradigmes de programmation du calcul scientifique. Il est alors possible d'appliquer ces paradigmes pour aborder des problèmes d'intérêt pratique, tels que la résolution des équations aux dérivées partielles, qui est abordée au cours de ce livre. La diversité des sujets abordés, l'efficacité des algorithmes présentés et leur écriture directe en langage C++ font de cet ouvrage un recueil fort utile dans la vie professionnelle d'un ingénieur.

Le premier chapitre présente les bases fondamentales pour la suite : présentation du langage C++ à travers la conception d'une classe de quaternions et outils d'analyse asymptotique du temps de calcul des algorithmes. Le second chapitre aborde l'algorithme de transformée de Fourier rapide et développe deux applications à la discrétisation d'équations aux dérivées partielles par la méthode des différences finies. Le troisième chapitre est dédié aux matrices creuses et à l'algorithme du gradient conjugué. Ces notions sont appliquées à la méthode des éléments finis. En annexe sont groupés des exemples de génération de maillage et de visualisation graphique.

S'il est cependant recommandé de maîtriser les notions du premier chapitre pour aborder le reste du livre, les chapitres deux et trois sont complètement indépendants et peuvent être abordés séparément. Ces chapitres sont complétés par des exercices qui en constituent des développements,

ainsi que des notes bibliographiques retraçant l'historique des travaux et fournissant des références sur des logiciels et librairies récents implémentant ou étendant les algorithmes présentés.

1.1 Un peu de théorie

(source: Wikipédia) 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.

1.2 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 1.2.1 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 1.1** soit $G = \{x \in \mathbb{R}^2 : |x| < 3\}$ et noté par: $x^0 = (1, 1)$; en considère la fonction:

$$f(x) = \begin{cases} e^{|x|} & \text{si } |x - x^0| \leq 1/2 \\ 0 & \text{si } |x - x^0| > 1/2 \end{cases} \quad (1.1)$$

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:

1.2.1 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 mieu intégré

1.3 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 fourni 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

1.3.1 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

1.3.2 Besoin de rester dans le symbolique

Le symbolique est une grande importance d'un point de vue technique, car il permet de limité les risques de bug dans l'exécution des programmes, dans le contexte de la vérification formelle si en prend le programme suivant:

1.3.3 Passage du symbolique au numérique

Généralement, le symbolique parmi c'est

1.3.4 Faire des dessins

2. Ensemble

2.1 Ensemble

2.1.1 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'appui 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.

Exercice 2.1 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 2.1 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
```

2.2 Premiers pas en Python symbolique

This is an example of theorems.

2.2.1 Variable et affectation

Exercise 2.2 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 2.2 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.

2.2.2 Contrôle du flux d'instructions

This is a theorem consisting of just one line.

Exercise 2.3 A set $\mathcal{D}(G)$ in dense in $L^2(G)$, $|\cdot|_0$. ■

Solution 2.3

2.3 Les Fonctions

This is an example of a definition. A definition could be mathematical or it could define a concept.

Exercise 2.4 Écrire une fonction cube qui retourne le cube de son argument ■

Exercise 2.5 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 2.6 Écrire une fonction $maFonction$ qui retourne $f(x) = 2x^3 + x - 5$ ■

Exercise 2.7 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 $\text{borneInf} < \text{borneSup}$. 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 2.8 Écrire une fonction *volMasse* Ellipsoide 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 2.9 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 2.10 Pour l'ensemble N de nombres naturels et une opération binaire $f : N \times N \rightarrow N$, 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 2.4 le deuxième et le troisième

2.4 Structures de données

2.4.1 Logique

Exercise 2.11 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? ■

2.4.2 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 certaine 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 2.12 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 2.5 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
```

2.4.3 Polynômes

Exercise 2.13 Considérons le polynôme $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3$, où $a_i \neq 0 \forall i$. Le nombre minimum de multiplications nécessaires pour évaluer p sur une entrée x est:

2.4.4 Nombres parfaits et nombres chanceux

Exercise 2.14 $\sqrt{12}$

2.5 Remarks

This is an example of a remark.

R The concepts presented here are now in conventional employment in mathematics. Vector spaces are taken over the field $\mathbb{K} = \mathbb{R}$, however, established properties are easily extended to $\mathbb{K} = \mathbb{C}$.

2.6 Corollaries

This is an example of a corollary.

Corollary 2.6.1 — Corollary name. The concepts presented here are now in conventional employment in mathematics. Vector spaces are taken over the field $\mathbb{K} = \mathbb{R}$, however, established properties are easily extended to $\mathbb{K} = \mathbb{C}$.

2.7 Propositions

This is an example of propositions.

2.7.1 Several equations

Proposition 2.7.1 — Proposition name. It has the properties:

$$|||\mathbf{x}|| - ||\mathbf{y}||| \leq ||\mathbf{x} - \mathbf{y}|| \quad (2.1)$$

$$||\sum_{i=1}^n \mathbf{x}_i|| \leq \sum_{i=1}^n ||\mathbf{x}_i|| \quad \text{where } n \text{ is a finite integer} \quad (2.2)$$

2.7.2 Single Line

Proposition 2.7.2 Let $f, g \in L^2(G)$; if $\forall \varphi \in \mathcal{D}(G)$, $(f, \varphi)_0 = (g, \varphi)_0$ then $f = g$.

2.8 Examples

This is an example of examples.

2.8.1 Equation and Text

■ **Example 2.1** Let $G = \{x \in \mathbb{R}^2 : |x| < 3\}$ and denoted by: $x^0 = (1, 1)$; consider the function:

$$f(x) = \begin{cases} e^{|x|} & \text{si } |x - x^0| \leq 1/2 \\ 0 & \text{si } |x - x^0| > 1/2 \end{cases} \quad (2.3)$$

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

2.8.2 Paragraph of Text

■ **Example 2.2 — Example name.** Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. ■

2.9 Exercises

This is an example of an exercise.

Exercise 2.15 This is a good place to ask a question to test learning progress or further cement ideas into students' minds. ■

2.10 Problems

Problem 2.1 What is the average airspeed velocity of an unladen swallow?

2.11 Vocabulary

Define a word to improve a students' vocabulary.

Vocabulary 2.1 — Word. Definition of word.

3. POO

3.1 Programmation Orientée Objet

Notation 3.1. Given an open subset G of \mathbb{R}^n , the set of functions φ are:

1. Bounded support G ;
2. Infinitely differentiable;

a vector space is denoted by $\mathcal{D}(G)$.

3.1.1 POO

Exercise 3.1 Définir une classe Vecteur2D avec un constructeur fournissant les coordonnées par défaut d'un vecteur du plan (par exemple : $x = 0$ et $y = 0$). Dans le programme principal, instanciez un Vecteur2D sans paramètre, un Vecteur2D avec ses deux paramètres, et affichez-les.

■

Solution 3.1 en utilise le module `sympy.geometry` ce module fait appel à tout les outils et theories qui peuvent être utiliser dans le cadre de la géométrie dans le Plan.

```
from sympy.geometry
```

Exercise 3.2 Enrichissez la classe Vecteur2D précédente en lui ajoutant une méthode d'affichage et une méthode de surcharge d'addition de deux vecteurs du plan. Dans le programme principal, instanciez deux Vecteur2D , affichez-les et affichez leur somme.

■

Solution 3.2

3.1.2 Notions de COO et d'encapsulation

4. Convex Sets

4.1 Convexity

4.1.1 Cone

■ **Definition 4.1.1 — Cone.** A set $K \in \mathbb{R}^n$, when $x \in K$ implies $\alpha x \in K$.

A non convex cone can be hyper-plane.

For convex cone $x + y \in K, \forall x, y \in K$.

Cone don't need to be "pointed". e.g.

Direct sums of cones $C_1 + C_2 = \{x = x_1 + x_2 | x_1 \in C_1, x_2 \in C_2\}$.

■ **Example 4.1** $S_1^n \{X | X = X^n, \lambda(x) \geq 0\}$

A matrix with positive eigenvalues.

Operations preserving convexity

Intersection $C \cap_{i \in \mathbb{I}} C_i$

Linear map Let $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a linear map. If $C \in \mathbb{R}^n$ is convex, so is $A(C) = \{Ax | x \in C\}$

Inverse image $A^{-1}(D) = \{x \in \mathbb{R}^n | Ax \in D\}$

Operations that induce convexity

Convex hull on $S = \cap \{C | S \in C, C \text{ is convex}\}$

■ **Example 4.2** $Co\{x_1, x_2, \dots, x_m\} = \{\sum_{i=1}^m \alpha_i x_i | \alpha \in \Delta_m\}$

For a convex set $x \in C \Rightarrow x = \sum \alpha_i x_i$.

■ **Theorem 4.1.1 — Carathéodory's theorem.** If a point $x \in \mathbb{R}^d$ lies in the convex hull of a set P , there is a subset P' of P consisting of $d+1$ or fewer points such that x lies in the convex hull of P' . Equivalently, x lies in an r -simplex with vertices in P .

4.2 Convex Functions

Definition 4.2.1 — Convex function. Let $C \in \mathbb{R}^n$ be convex, $f : C \rightarrow \mathbb{R}$ is convex on f if $x, y \in C \times C$. $\forall \alpha \in (0, 1)$, $f(\alpha x + (1 - \alpha)y) \leq f(\alpha x) + f((1 - \alpha)y)$

Definition 4.2.2 — Strictly Convex function. Let $C \in \mathbb{R}^n$ be convex, $f : C \rightarrow \mathbb{R}$ is strictly convex on f if $x, y \in C \times C$. $\forall \alpha \in (0, 1)$, $f(\alpha x + (1 - \alpha)y) < f(\alpha x) + f((1 - \alpha)y)$

Definition 4.2.3 — Strongly convex. $f : C \rightarrow \mathbb{R}$ is strongly convex with modulus $u \geq 0$ if $f - \frac{1}{2}u\|\cdot\|^2$ is convex.

Interpretation: There is a convex quadratic $\frac{1}{2}u\|\cdot\|^2$ that lower bounds f .

■ **Example 4.3** $\min_{x \in C} f(x) \leftrightarrow \min \bar{f}(x)$ Useful to turn this into an unconstrained problem.

$$\bar{f}(x) = \begin{cases} f(x) & \text{if } x \in C \\ \infty & \text{elsewhere} \end{cases}$$

Definition 4.2.4 A function $f : \mathbb{R}^n \rightarrow \mathbb{R} \cup \infty \bar{\mathbb{R}}$ is convex if $x, y \in \mathbb{R}^n \times \mathbb{R}^n$, $\forall x, y, \bar{f}(\alpha x + (1 - \alpha)y) \leq f(\alpha x) + f((1 - \alpha)y)$

Definition 1 is equivalent to definition 2 if $f(x) = \infty$.

■ **Example 4.4** $f(x) = \sup_{j \in J} f_j(x)$

4.2.1 Epigraph

Definition 4.2.5 — Epigraph. For $f : \mathbb{R}^n \rightarrow \bar{\mathbb{R}}$, its epigraph $epi(f) \in \mathbb{R}^{n+1}$ is the set $epi(f) = \{(x, \alpha) | f(x) \in \alpha\}$

Next: a function is convex i.f.f. its epigraph is convex.

Definition 4.2.6 A function $f : C \rightarrow \mathbb{R}$, $C \in \mathbb{R}^n$ is convex if $\forall x, y \in C$, $f(ax + (1 - a)x) \leq af(x) + (1 - a)f(y) \quad \forall a \in (0, 1)$.

Strict convex: $x \neq y \Rightarrow f(ax + (1 - a)x) < af(x) + (1 - a)f(y)$

(R) f is convex $\Rightarrow -f$ is concave.

Level set: $S_\alpha f = \{x | f(x) \leq \alpha\}$.

$S_\alpha f$ is convex $\Leftrightarrow f$ is convex.

Definition 4.2.7 — Strongly convex. $f : C \rightarrow \mathbb{R}$ is strongly convex with modulus μ if $\forall x, y \in C$, $\forall \alpha \in (0, 1)$, $f(ax + (1 - \alpha)y) \leq af(x) + (1 - \alpha)f(y) - \frac{1}{2\mu}\alpha(1 - \alpha)\|x - y\|^2$.

(R)

- f is 2nd-differentiable, f is convex $\Leftrightarrow \nabla^2 f(x) \succ 0$.
- f is strongly convex $\Leftrightarrow \nabla^2 f(x) \succ \mu I \Leftrightarrow x \geq \mu$

Definition 4.2.8 — 2. $f : \mathbb{R}^n \rightarrow \bar{\mathbb{R}}$ is convex if $x, y \in \mathbb{R}, \alpha \in (0, 1), f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y)$.

The effective domain of f is $\text{dom } f = \{x | f(x) < +\infty\}$

■ **Example 4.5 — Indicator function.** $\delta_C(x) = \begin{cases} 0 & x \in C \\ +\infty & \text{elsewhere} \end{cases}$.
 $\text{dom } \delta_C(x) = C$

Definition 4.2.9 — Epigraph. The epigraph of f is $\text{epif} = \{(x, \alpha) | f(x) \leq \alpha\}$

The graph of epif is $\{(x, f(x)) | x \in \text{dom } f\}$.

Definition 4.2.10 — III. A function $f : \mathbb{R}^n \rightarrow \bar{\mathbb{R}}$ is

Theorem 4.2.1 $f : \mathbb{R}^n \rightarrow \bar{\mathbb{R}}$ is convex $\iff \forall x, y \in \mathbb{R}^n, \alpha \in (0, 1), f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y)$.

Proof. \Rightarrow take $x, y \in \text{dom } f, (x, f(x)) \in \text{epif}, (y, f(y)) \in \text{epif}$. ■

■ **Example 4.6 — Distance.** Distance to a convex set $d_C(x) = \inf\{\|z - x\| | z \in C\}$. Take any two sequences $\{y_k\}$ and $\{\bar{y}_k\} \subset C$ s.t. $\|y_k - x\| \rightarrow d_C(x)$, $\|\bar{y}_k - \bar{x}\| \rightarrow d_C(\bar{x})$. $z_k = \alpha y_k + (1 - \alpha)\bar{y}_k$.

$$\begin{aligned} d_C(\alpha x + (1 - \alpha)\bar{x}) &\leq \|z_k - \alpha x - (1 - \alpha)\bar{x}\| \\ &= \|\alpha(y_k - x) + (1 - \alpha)(\bar{y}_k - \bar{x})\| \\ &\leq \alpha\|y_k - x\| + (1 - \alpha)\|\bar{y}_k - \bar{x}\| \end{aligned}$$

Take $k \rightarrow \infty, d_C(\alpha x + (1 - \alpha)\bar{x}) \leq \alpha d_C(x) + (1 - \alpha)d_C(\bar{x})$

■ **Example 4.7 — Eigenvalues.** Let $X \in S^n := \{n \times \text{nsymmetricmatrix}\}$. $\lambda_1(x) \geq \lambda_2(x) \geq \dots \geq \lambda_n(x)$.

$$f_k(x) = \sum_i^n \lambda_i(x).$$

Equivalent characterization

$$\begin{aligned} f_k(x) &= \max\{\sum_i v_i^T X v_i | v_i \perp v_j, i \neq j\} \\ &= \max\{\text{tr}(V^T X V) | V^T V = I_k\} \end{aligned}$$

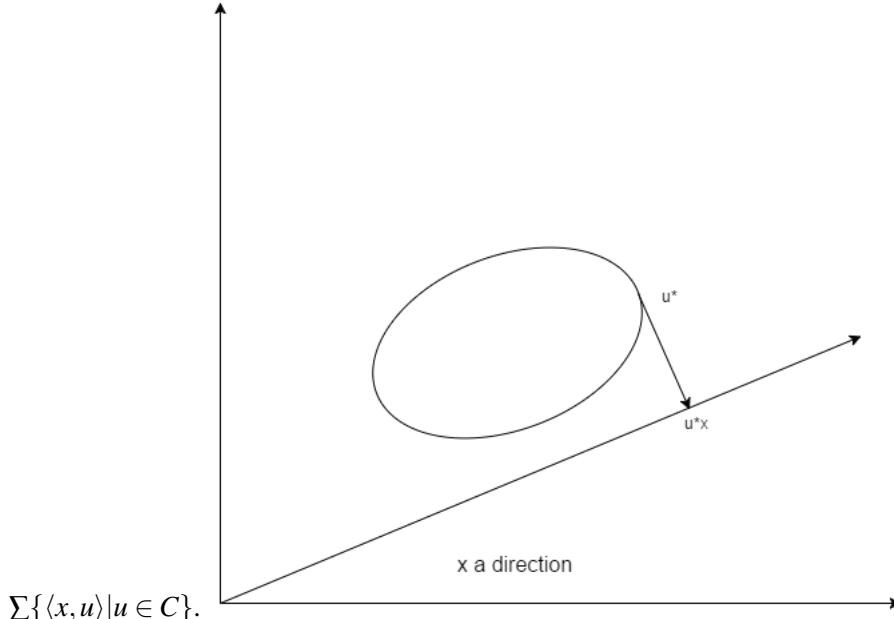
$\max\{\text{tr}(V^T X V)\}$ by circularity

Note $\langle A, B \rangle = \text{tr}(A, B)$ is true for symmetric matrix.

$$\langle A, A \rangle = |A|_F^2 = \sum_i A_{ii}^2$$

4.3 Support Function

Take a set $C \in \mathbb{R}^n$, not necessarily convex. The support function is $\sigma_C = \mathbb{R}^n \rightarrow \bar{\mathbb{R}}$. $\sigma_C(x) =$



Fact 4.3.1 The support function binds the supporting hyper-plane.

Supporting functions are

- Positively homogeneous

$$\sigma_C(\alpha x) = \alpha \sigma_C(x) \forall \alpha > 0$$

$$\sigma_C(\alpha x) = \sup_{u \in C} \langle \alpha x, u \rangle = \alpha \sup_{u \in C} \langle x, u \rangle = \alpha \sigma_C(x)$$

- Sub-linear (a special case of convex, linear combination holds $\forall \alpha$).

$$\sigma_C(\alpha x + (1 - \alpha)y) = \sup_{u \in C} \langle \alpha x + (1 - \alpha)y, u \rangle \leq \alpha \sup_{u \in C} \langle x, u \rangle + (1 - \alpha) \sup_{u \in C} \langle y, u \rangle$$

■ **Example 4.8 — L2-norm.** $\|x\| = \sup_{u \in C} \{\langle x, u \rangle, u \in \mathbb{R}^n\}$.

$$\|x\|_p = \sup\{\langle x, u \rangle, u \in B_q\} \text{ where } \frac{1}{p} + \frac{1}{q} = 1. B_q = \{\|x\|_q \leq 1\}.$$

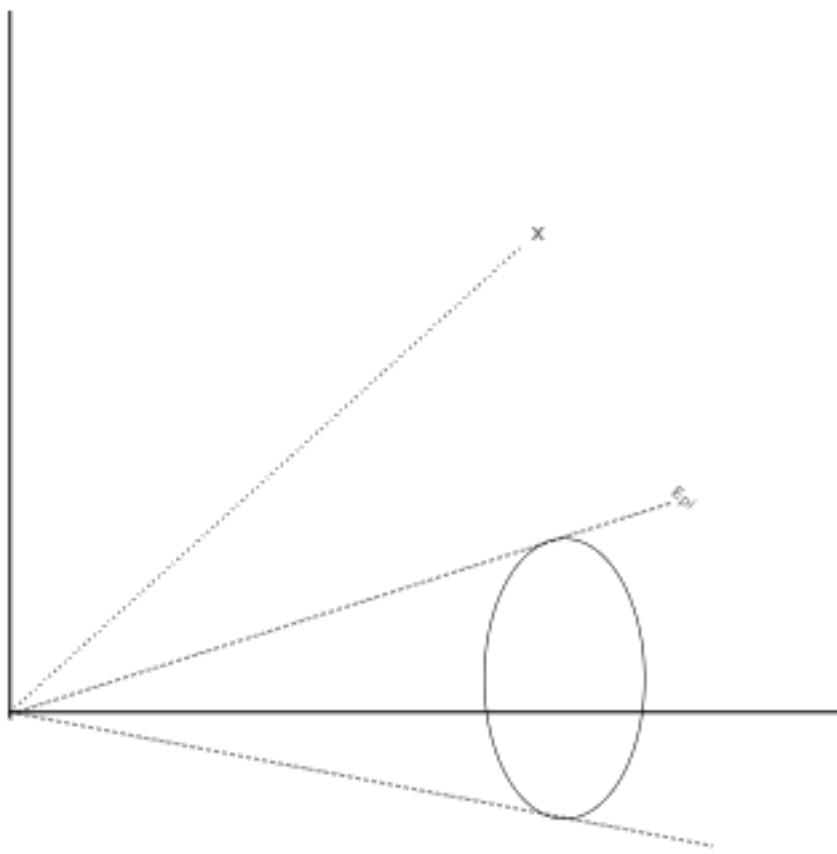
The norm is

- Positive homogeneous
- sub-linear
- If $0 \in C$, σ_C is non-negative.
- If C is central-symmetric, $\sigma_C(0) = 0$ and $\sigma_C(x) = \sigma_C(-x)$

■ **Fact 4.3.2 — Epigraph of a support function.** $epi\sigma_C = \{(x, t) | \sigma_C(x) \leq t\}$. Suppose $(x, t) \in epi\sigma_C$.

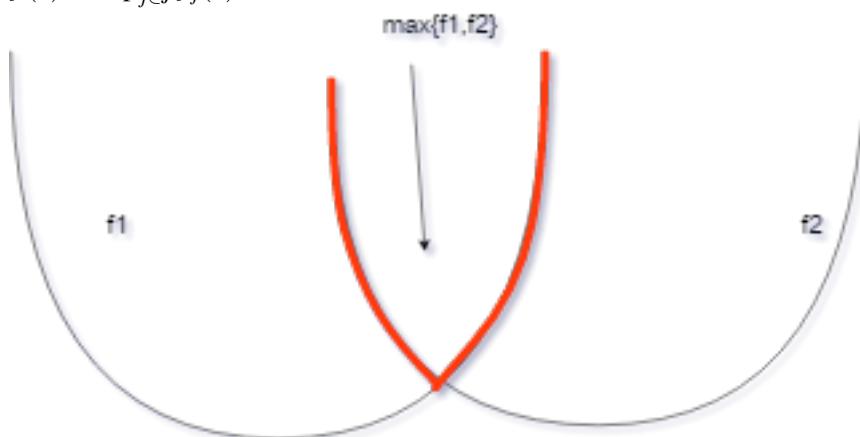
Take any $\alpha > 0$. $\alpha(x, t) = (\alpha x, \alpha t)$.

$$\alpha \sigma_C(x) = \alpha \sigma_C(x) \leq \alpha t. \alpha(x, t) \in epi\sigma_C$$



4.4 Operations Preserve Convexity of Functions

- Positive affine transformation
 $f_1, f_2, \dots, f_k \in \text{cvx} \mathbb{R}^n$
 $f = \alpha_1 f_1 + \alpha_2 f_2 + \dots + \alpha_k f_k$
- Supremum of functions. Let $\{f_i\}_{i \in I}$ be arbitrary family of functions. If $\exists x \sup_{j \in J} f_j(x) < \infty \Leftrightarrow f(x) = \sup_{j \in J} f_j(x)$



- Composition with linear map.
 $f \in \text{cvx} \mathbb{R}^n, A : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a linear map. $f \circ A(x) = f(Ax) \in \text{cvx} \mathbb{R}^n$

$$\begin{aligned}f \circ A(x) &= f(A(\alpha x + (1 - \alpha)y)) \\&= f(A\alpha x + (1 - \alpha)Ay) \\&\leq \alpha f(Ax) + (1 - \alpha)f(Ay)\end{aligned}$$



5. Nonlinear Problem

Les sujets de ce chapitre sont du néanmoins axées sur des questions ou l'approche mathématique et physique et demandé

5.1 Équation différentielle

Sans aucun doute

5.2 Chaos

Prenons une pause dans l'apprentissage de nouvelles techniques et algorithmes informatiques pour un peu, et passer du temps en utilisant ce que nous avons appris jusqu'à présent pour enquêter sur quelque chose d'intéressant. Nous allons commencer avec quelque chose de familier: le simple pendule.

5.2.1 Pendule simple

Le pendule simple figure

5.2.2 Pendule à deux bras

5.2.3 Mouvements d'un robot

Qu'est ce qu'il faut savoir quand en veut modélisé le comportement d'un robot?. Et bien la réponse est tout simplement des mathématiques

5.3 Solution non linéaire d'équation algébrique

Qu'est ce que non-linéaire et qu'est ce que une équation algébrique

Une équation algébrique est un polynôme de la forme $P(x)$

$$\exp(-x) \sin(x) = \cos(x) \quad (5.1)$$

5.4 La théorie de catégorie

L'interet de la théorie de catégorie dans les mathématiques modernes et l'informatique théorique et qui dépasse de loin, cette dernière pour être

5.5 Mécanique quantique

5.6 Transport optimal

C'est quoi le *transport optimal*? exemple simple..., le domaine du transport optimal est très

5.7 Le calcul des variations

5.8 Figure

Treatments	Response 1	Response 2
Treatment 1	0.0003262	0.562
Treatment 2	0.0015681	0.910
Treatment 3	0.0009271	0.296

Table 5.1: Table caption



6. Outils avancée

6.1 Programmation Orientée Objet

6.2 Décorateurs

Les décorateurs un mécanisme incontournable pour écrire de très bon code et purement lisible et portable

6.2.1 Optimisation du code

Sans aucun doute l'usage de la programmation symbolique avec ce que en a vue plus haut, ralentisse grandement l'exécution du programme, donc en gagne sur le coté sûreté, élégance et maintenance du code et d'autre part en perd complètement la vitesse; penser à des centaines de lignes de code si vous voulez programmer un robot, voiture ou des objets connectés qui implémentent des algorithmes mathématiques et qui demandent beaucoup de ressource est un temps de retour très élevées

6.2.2 Cython

Cython (<http://www.cython.org/>) est un métalangage qui permet de combiner du code Python et des types de données C, pour concevoir des extensions compilables pour Python. Dans un module Cython, il est possible de définir des variables C directement dans le code Python et de définir des fonctions C qui prennent en paramètre des variables C ou des objets Python. Cython contrôle ensuite de manière transparente la génération de l'extension C, en transformant le module en code C par le biais des API C de Python. Toutes les fonctions Python du module sont alors automatiquement publiées. Le gain de temps dans la conception introduit par Cython est considérable : toute la mécanique habituellement mise en œuvre pour créer un module d'extension est entièrement gérée par Cython. Ainsi, la fonction max() du module calculs.c précédemment présentée devient :

Les fichiers Cython ont par convention l'extension pyx, en référence à l'ancien nom.

setup.py pour calculs.pyx

```
from distutils.core import setup
from distutils.extension import Extension
from Cython.Distutils import build_ext

extension = Extension("calculs", ["calculs.pyx"])

setup(name="calculs", ext_modules=[extension], cmdclass={'build_ext': build_ext})
```

6.2.3 Theano

Theano est une bibliothèque pour l'accélération du code lent en Python, très importante et intéressante elle offre une syntaxe très particulière.

6.3 Interface graphique

Quelle bibliothèque choisir: sous Python en à le choix entre différente, Tkinter, Gtk, Qt, wx et ftk, et il existe encore d'autre bibliothèques qui sont conçu pour le calcul et application scientifique éditer par Thought[...] dans cette section nous allons. Une autre approche serait d'utiliser les ipywidgets avec Jupyter ou JupyterLab notebook, les ipywidgets sont très intéressante approches pour des graphiques interactives rapide d'usages et bénéfiques sur le plan de présentation par exemple quand en veut exporté ou les partagés