

Michaël Baudin

Méthodes numériques avec Python

Théorie, algorithmes, implémentation
et applications avec Python 3

DUNOD

Graphisme de couverture : Elizabeth Riba
Illustration de couverture : © Lauren Suryanata - Shutterstock.com

© Dunod, 2023
11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-085338-0

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2° et 3° a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

Avant-propos

Le calcul scientifique en Python

L'objectif de ce manuel est de présenter les principes mathématiques, les applications et la mise en œuvre de méthodes de calcul scientifique en Python. Ce texte évoque, tour à tour, la pratique et la théorie : nous montrons l'utilisation de ces méthodes dans le langage Python en utilisant les bibliothèques `NumPy` et `SciPy` et l'analyse théorique sur laquelle le calcul s'appuie.

Partout où cela est possible et pertinent, nous présenterons des applications réelles plutôt que des exemples simplifiés ou théoriques. En effet, l'objectif de ce texte est de montrer comment utiliser au mieux ces outils en pratique et, en particulier, dans la pratique du chercheur et de l'ingénieur.

Utilisation du manuel

Ce livre s'adresse aux étudiants en deuxième année de licence ou au-delà, aux ingénieurs et aux chercheurs. Pour en tirer profit, il est utile d'avoir déjà suivi un cours de Python et de posséder des bases en analyse et en algèbre.

Le manuel peut être utilisé pour un cours d'un semestre composé de séances de cours, de travaux dirigés et de travaux pratiques. Cela constitue le contenu pour un module d'environ 10 à 15 séances de 3 heures, ce qui représente un total compris entre 30 et 50 heures d'enseignement.

Méthode pédagogique

Une des sources d'inspiration initiales de ce texte est le livre de Cleve Moler [104]. Ce livre s'appuyait lui-même sur une généalogie de livres dont le plus ancien est [46], un livre pionnier mêlant les méthodes, les algorithmes et les applications. Traduire en langage Python l'esprit de cette introduction au calcul numérique est rapidement apparu une approche incomplète dans la mesure où ces ouvrages laissent, d'une part, peu de place à la théorie et, d'autre part, aucune aux démonstrations. Puisque les démonstrations sont au cœur des mathématiques, nous avons complété le texte par des définitions et des théorèmes, associés à des preuves. L'objectif est de fournir au lecteur non seulement des méthodes applicables, des scripts Python fonctionnels, mais aussi une compréhension détaillée des calculs.

Dans le but de pouvoir concentrer la première lecture sur les concepts et non pas sur les détails, les démonstrations sont généralement présentées à travers les exemples

Nombre d'étoiles	Difficulté
Aucune	Mise en œuvre directe du cours
*	Un argument est requis
**	Plusieurs arguments sont requis
***	Plusieurs raisonnements se combinent

TABLE 1 – Difficultés des exercices.

ou détaillées sous la forme d'exercices.

Les exercices qui accompagnent la partie théorique du cours constituent une partie indispensable de l'apprentissage. Cela est une conséquence du choix de déplacer une partie de l'analyse de la section principale vers les exercices, dans le but de concentrer la présentation principale sur l'essentiel. Ces exercices sont triés par ordre de difficulté croissante, en fonction des critères indiqués dans la table 1.

Certains lecteurs trouvent que les démonstrations sont parfois trop détaillées. La qualité d'une démonstration tient essentiellement dans la qualité de son argumentation. En pratique, ce niveau de détail est bienvenu pour d'autres lecteurs, car cela leur permet de comprendre un point qui leur a échappé.

Partout où le soupçon d'une astuce mathématique se laissait entrevoir, nous avons préféré l'analyse rendant à la démonstration son caractère logique et cohérent avec l'objectif poursuivi. C'est la raison pour laquelle, parmi plusieurs démonstrations possibles d'un théorème, nous avons toujours préféré la plus riche du point de vue pédagogique, même si ce n'est pas la plus courte. Il s'ensuit des démonstrations aussi rigoureuses que possible, mais occasionnellement longues : le lecteur nous le pardonnera plus volontiers (nous l'espérons), pourvu que cette démonstration ait pu révéler des mécanismes intéressants.

Une partie significative de ce texte est consacrée au conditionnement des problèmes que nous allons traiter. En effet, bien qu'il occupe une place importante dans le calcul impliquant des nombres à virgule flottante, le conditionnement n'est pas un concept lié uniquement au calcul numérique, mais un concept mathématique associé à l'idée plus générale de changement de la solution lorsque les données sont perturbées. Cette idée, associée au concept de *problème bien posé*, ne peut pas être ignorée lorsqu'on utilise des méthodes numériques.

À chaque fois que cela est possible et pertinent, nous illustrons le concept mathématique par une figure, que celle-ci soit créée par une librairie de dessin vectoriel ou qu'elle soit créée en Python. L'exercice, pour le lecteur, consiste alors à observer la figure en détail pour faire le lien entre la figure et la définition ou le théorème. C'est un exercice que j'ai souvent demandé à mes étudiants, en précisant la grille de lecture d'un tel graphique : l'axe des abscisses, l'axe des ordonnées, le contenu du graphique et le lien avec le concept mathématique.

Dans certains contextes, la bibliographie francophone a des traditions dont certaines sont appropriées et d'autres non. Lorsque le présent manuel dévie de la tradition, nous expliquerons notre choix de manière explicite. C'est, par exemple, le cas pour la formule de Taylor page 38 ainsi que pour le théorème de la valeur intermédiaire page 363.

Certaines sections présentent des concepts plus avancés et dont la compréhension peut être difficile lors d'une première lecture, en fonction du niveau du lecteur. Nous avons indiqué ces chapitres par un astérisque (*) dans le titre. C'est le cas par exemple

pour le chapitre 6.6, qui peut être passée lors d’une première lecture. Au contraire, le lecteur désireux d’approfondir ses connaissances pourra utiliser ces indications pour identifier les chapitres les plus avancés. Par exemple, cela peut être utile pour un enseignant qui souhaiterait dispenser ce cours à un niveau supérieur.

Valeurs numériques

En langue française, le séparateur décimal est la virgule « , », ce qui est différent de la langue anglaise dans laquelle le séparateur décimal est le point « . ». Nous allons voir par la suite que le langage Python utilise la convention anglaise pour le séparateur décimal et utilise la virgule pour d’autres objectifs. Dans le contexte de ce livre, à la croisée des mathématiques et de l’informatique, dans le but de rendre le texte cohérent, nous avons adopté la convention anglaise pour le séparateur décimal, c’est-à-dire le point.

Un livre de calcul scientifique ne contenant aucune valeur numérique serait une absurdité. En effet, nous allons analyser en détail pourquoi les nombres flottants avec lesquels la plupart des utilisateurs de Python réalisent des calculs ont approximativement 16 chiffres significatifs. Toutefois, écrire toutes les valeurs numériques avec une précision aussi importante n’apporte pas nécessairement d’information pertinente.

Pour cette raison, nous arrondissons tous les résultats numériques au plus proche, avec 4 chiffres significatifs. Cela signifie que, lorsque le nombre mathématique est « 1.23456 », nous écrirons dans le texte « 1.235 ». De même, le nombre « 0.001234 » possède 4 chiffres significatifs, car les zéros devant le premier chiffre non nul ne comptent pas. Nous utiliserons occasionnellement la notation scientifique « 1.234×10^{-3} ».

Une précision importante est que nous présentons 4 chiffres significatifs y compris dans la plupart des résultats des sessions Python (mais Python utilise davantage de chiffres significatifs dans ses calculs). C’est une des raisons pour lesquelles, si le lecteur reproduit les scripts Python présentés dans ce texte, les valeurs numériques peuvent être légèrement différentes. Une autre raison fréquente est que les sessions présentées dans ce texte ont été exécutées sur différents systèmes (Linux et Windows), ce qui peut produire des résultats légèrement différents. Ces différences sont la plupart du temps sans conséquence. Plus de détails sur ce sujet seront présentés dans le chapitre 4 consacré aux nombres flottants.

Thèmes

Les différentes méthodes que nous allons détailler dans ce livre forment un réseau, de telle sorte que certaines méthodes s’appuient les unes sur les autres. La figure 1 présente certains de ces liens. Tentons de les éclaircir.

D’une manière générale, deux approches mathématiques se font face : l’algèbre et l’analyse. Dans notre contexte, deux sujets sont plus difficiles, plus fondamentaux ou plus importants que d’autres, car ils sont à la base de nombreuses autres techniques : il s’agit des systèmes d’équations linéaires (algèbre) et des équations différentielles ordinaires (analyse). D’un côté, les systèmes d’équations linéaires sont un thème essentiel en algèbre linéaire et ouvrent la porte à de nombreuses applications ainsi qu’à d’autres sujets, en particulier en algèbre linéaire numérique. D’un autre côté, les équations dif-

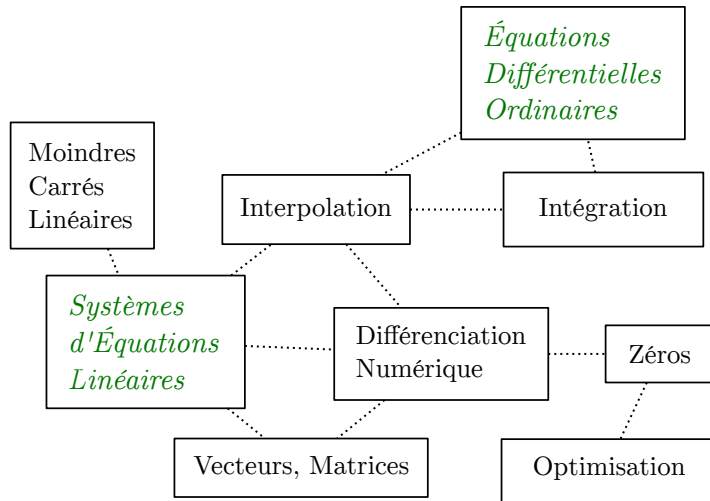


FIGURE 1 – Liens entre les méthodes numériques.

férentielles ordinaires sont à la base de nombreux modèles physiques et ouvrent la porte aux équations aux dérivées partielles.

La figure 1 montre bien que certains thèmes sont en réalité à la croisée de l’algèbre et de l’analyse. Dans le contexte de l’interpolation par exemple, représenter une fonction sur une base de fonctions donnée nécessite de considérer un espace vectoriel de dimension infinie. Puisque ce n’est pas pratique, nous ne représentons souvent que la projection de cette fonction sur un espace vectoriel de dimension finie, pour lequel l’algèbre fournit les outils de calcul appropriés. Nous laissons le soin au lecteur d’établir de nouveaux liens entre ces méthodes.

Scripts d’accompagnement

Les bibliothèques NumPy et SciPy sont d’excellentes bibliothèques de calcul scientifique et nous les utiliserons tout au long de cet ouvrage. Toutefois, dans certaines circonstances, nous pensons que les fonctionnalités dont nous avons besoin pour un thème précis sont mieux illustrées avec des algorithmes simplifiés, moins robustes, mais dont le code source est plus explicite.

C’est la raison pour laquelle nous avons développé une collection de scripts Python pédagogiques accompagnant le cours. Ces scripts sont disponibles sur le dépôt github.com/mbaudin47/menum_code. La table 2 présente la liste des modules. Ils sont disponibles dans le répertoire `Scripts-Eleves/Py3` du dépôt. Pour les utiliser, il est nécessaire de mettre à jour la variable `PYTHONPATH`, de telle sorte que l’environnement Python puisse connaître le nom du répertoire contenant les scripts. Configurer cette variable peut dépendre du système d’exploitation. Sur Linux, par exemple, on peut exécuter la ligne suivante dans le terminal :

```
1 export PYTHONPATH="/home/monlogin/Scripts-Eleves/Py3:$PYTHONPATH"
```

Ces modules pédagogiques sont utiles pour l’étude des algorithmes, mais je ne recommande à personne de les utiliser pour des études d’ingénierie ou bien pour

<code>floats.py</code>	Nombres à virgule flottante
<code>fzero.py</code>	Équations non linéaires
<code>interp.py</code>	Interpolation
<code>leastsq.py</code>	Moindres carrés linéaires
<code>linalg.py</code>	Systèmes d'équations linéaires
<code>numdiff.py</code>	Différentiation numérique
<code>odes.py</code>	Équations différentielles ordinaires
<code>optim.py</code>	Optimisation
<code>quadrature.py</code>	Intégration

TABLE 2 – Modules fournis avec le manuel.

la recherche : leur objectif n'est ni la performance, ni la précision, ni la souplesse d'utilisation, ni la robustesse. En effet, il existe une différence importante entre un algorithme et son implémentation (nous reviendrons sur ce thème dans la section 5.9). Toutefois, ces modules sont documentés et testés.

Pour chaque chapitre, les scripts sont disponibles dans le sous-répertoire **Scripts**. Pour le chapitre intégration, par exemple, les scripts sont disponibles dans le répertoire **Integration/Scripts**.

Pour les scripts les plus simples, nous fournissons parfois un script « squelette », contenant une base de travail que l'élève peut compléter durant les séances de travaux pratiques en Python. Dans ce script, le travail consiste à remplacer les instructions `TODO` par du code valide. L'objectif est de ne pas avoir à débiter par un script vide. Le script suivant¹, par exemple, présente un modèle de script pour l'intégration numérique. Nous indiquons dans une note de bas de page le nom du fichier correspondant au script présenté dans l'exemple ou dans la figure correspondante.

```

1 # Import des modules
2 TODO
3
4 def myfunB(x):
5     y = TODO
6     return y
7
8 Q, fcount = adaptsim_gui(myfunB, 0.0, 1.0)
```

Le script suivant² présente la solution du problème.

```

1 import numpy as np
2 from quadrature import adaptsim
3
4 def myfunB(x):
5     y = 1.0 / np.sqrt(1.0 + x ** 4)
6     return y
7
8 Q, fcount = adaptsim(myfunB, 0.0, 1.0)
```

Occasionnellement, nous distinguons une fonction mathématique de son implémentation en Python. Dans ce cas, nous employons un changement de fonte pour

¹Script Python : `Integration/Scripts/Py3/exemples-squelette.py`.

²Script Python : `Integration/Scripts/Py3/exemples.py`.

$\mathbb{R}$	Nombres réels, par ex. $x = 1.234$
$\mathbb{R}^*$	Nombre réels non nuls : $\{x \in \mathbb{R}, x \neq 0\}$
$\mathbb{C}$	Nombres complexes, par ex. $z = 1 + 2i$
$\mathbb{N}$	Entiers naturels, par ex. $i = 1, 2, 3, \dots$
$\mathbb{Z}$	Entiers relatifs, par ex. $m = \dots, -2, -1, 0, 1, 2, \dots$
$\mathbb{Z}^*$	Entiers relatifs non nuls : $\{k \in \mathbb{Z} \text{ tels que } k \neq 0\}$
$\text{Re}(z)$	La partie réelle du nombre complexe $z \in \mathbb{C}$
$\text{Im}(z)$	La partie imaginaire du nombre complexe $z \in \mathbb{C}$
x, y, z	Des nombres réels
i, j, k	Des entiers
$[a, b]$	L'ensemble : $\{x \in \mathbb{R} \mid a \leq x \leq b\}$
(a, b)	L'ensemble : $\{x \in \mathbb{R} \mid a < x < b\}$
<code>x</code>	Variable Python

TABLE 3 – Notations.

α	A	Alpha	ι	I	Iota	ρ	P	Rho
β	B	Beta	κ	K	Kappa	σ	Σ	Sigma
γ	Γ	Gamma	λ	Λ	Lambda	τ	T	Tau
δ	Δ	Delta	μ	M	Mu	υ	Y	Upsilon
ϵ	E	Epsilon	ν	N	Nu	ϕ	Φ	Phi
ζ	Z	Zeta	ξ	Ξ	Xi	χ	X	Chi
η	H	Eta	o	O	Omicron	ψ	Ψ	Psi
θ	Θ	Theta	π	Π	Pi	ω	Ω	Omega

TABLE 4 – Lettres grecques.

signifier cette distinction. Par exemple, nous notons $\sin$ la fonction mathématique trigonométrique et `sin` son implémentation en Python.

Notations

Nous utiliserons les notations présentées dans la table 3. La table 4 présente les lettres grecques utilisées dans l'ouvrage. Dans le contexte de l'algèbre linéaire, les conventions présentées dans la table 5 sont adoptées. Elles correspondent à celles employées dans [70] page 2 à une exception : nous notons en caractères gras les vecteurs ainsi que le fait [132]. Ce livre a été préparé avec $\text{\LaTeX}$ ³.

Remerciements

J'adresse des remerciements tout particuliers à Jean-Marc Martinez qui m'a fait confiance pour commencer à enseigner ce type de méthodes et sans qui ce livre n'existerait probablement pas. Ses commentaires et correction sur ce manuel ont été très bénéfiques. Mes remerciements vont à Bernard Hugueney qui m'a encouragé à approfondir ce sujet. Je remercie les élèves de l'Institut supérieur d'électronique de Paris

³<https://www.latex-project.org/>

α	Scalaire (réel ou complexe) : lettre grecque
$\mathbf{x}$	Vecteur : lettre latine, minuscule, en gras
$\langle \mathbf{x}, \mathbf{y} \rangle$	Produit scalaire des vecteurs $\mathbf{x}$ et $\mathbf{y}$
$\mathbb{R}^n$	Ensemble des vecteurs réels de dimension n
$\mathbb{R}^{m \times n}$	Ensemble des matrices réelles à m lignes et n colonnes
$\mathbf{x}$	Vecteur : lettre latine, minuscule, en gras
diag	Matrice diagonale
$\mathbf{I}$	Matrice identité
rang	Rang d'une matrice
$\ \mathbf{x}\ $	Norme du vecteur $\mathbf{x}$
P	Matrice de permutation
Q	Matrice orthogonale
L	Matrice triangulaire inférieure
U	Matrice triangulaire supérieure
A^{-1}	Inverse de A
$\kappa(A)$	Conditionnement de la matrice A
$A^\dagger$	Pseudo-inverse de A

TABLE 5 – Notations pour l'algèbre linéaire.

(ISEP) qui, par leurs questions et leurs commentaires, m'ont stimulé dans la rédaction de ce document. Mes remerciements s'adressent également aux enseignants qui m'ont accompagné sur le module de méthodes numériques, en particulier Omar Al Hammal. Je remercie le Dr. Jean-Pierre de Mondenard pour sa relecture du chapitre consacré aux applications et en particulier à la prévention de la commotion cérébrale. Je souhaite remercier Régis Lebrun pour ses suggestions et corrections. Je tiens à remercier Vincent Chabridon pour ses encouragements et ses corrections. Je remercie enfin mon épouse Vi-An Baudin pour ses relectures, commentaires et corrections détaillées.

J'invite le lecteur à me transmettre ses suggestions à

`methodes.numeriques.python@gmail.com`.

Michaël Baudin,
Août 2022

Table des matières

I	Prolégomènes au calcul scientifique	17
1	À quoi sert le calcul numérique ?	19
1.1	Prévention de la commotion cérébrale	19
1.2	Réel et modèle	21
1.3	Exemple d'essais en soufflerie	23
1.4	Essais difficiles, coûteux ou impossibles	24
1.5	Conclusion	25
1.6	Notes et références	26
2	Python	27
2.1	Variables	28
2.2	Listes et tuples	28
2.3	Conditions et boucles	30
2.4	Fonctions et modules	31
2.5	Conclusion	33
3	Outils mathématiques	35
3.1	Notation de Landau	36
3.2	Formule de Taylor	38
3.3	Condition d'optimalité	44
3.4	Conclusion	44
3.5	Notes et références	44
3.6	Exercices	45
4	Flottants	49
4.1	Représentation binaire des entiers	49
4.2	Les nombres à virgule flottante	52
4.3	Des fonctions utiles	56
4.4	Flottants normalisés, dénormalisés	57
4.5	Flottants extrêmes et arrondi	59
4.6	En Python	62
4.7	Bit implicite (*)	64
4.8	Erreurs d'arrondi	65
4.9	Conclusion	67
4.10	Notes et références	68
4.11	Exercices	68
5	Convergence, erreurs et conditionnement	71

5.1	Convergence	71
5.2	Erreur absolue et erreur relative	75
5.3	Erreur directe et inverse	78
5.4	Conditionnement	80
5.5	Les fonctions $\log$ et $\exp$	83
5.6	Conditionnement dans le cas vectoriel (*)	85
5.7	Conditionnement de la somme (*)	87
5.8	Applications	88
5.8.1	Fiabilité d'une batterie	88
5.8.2	Attaque cryptographique	90
5.9	Conclusion	91
5.10	Notes et références	91
5.11	Exercices	91
6	Vecteurs, Matrices	97
6.1	Vecteurs	97
6.2	Produit scalaire et norme vectorielle	100
6.3	Matrices	103
6.4	Produit matrice-vecteur	105
6.5	Norme matricielle	109
6.6	Vecteur optimal (*)	111
6.7	Matrices particulières	114
6.8	Produit tensoriel	115
6.9	Matrice identité, inverse et permutation	115
6.10	Orthogonalité	118
6.11	Application : robotique	119
6.12	Conclusion	121
6.13	Notes et références	121
6.14	Exercices	121
II	Méthodes numériques	127
7	Systèmes d'équations linéaires	129
7.1	Dépendance et indépendance	130
7.2	Rang et inverse	132
7.3	Conditionnement d'un système d'équations linéaires	134
7.4	Méthode de Gauss	138
7.5	Décomposition LU avec permutations	142
7.6	Utiliser la décomposition $PA=LU$	146
7.7	La permutation des lignes	148
7.8	Analyse matricielle du pivot de Gauss	150
7.9	Le facteur de croissance (*)	152
7.10	Chiffres significatifs dans la solution	155
7.11	Effet géométrique d'une perturbation	157
7.12	Application : calcul d'un réseau électronique	161
7.13	Conclusion	163
7.14	Notes et références	163
7.15	Exercices	163

8	Interpolation	177
8.1	Le polynôme de Lagrange	180
8.2	Implémentation de l'interpolateur de Lagrange	184
8.3	Nœuds de Chebyshev	186
8.4	Matrice de Vandermonde	187
8.5	Interpolation linéaire par morceaux	190
8.6	Erreur d'interpolation	193
8.7	La constante de Lebesgue (*)	198
8.8	Conditionnement de l'interpolation (*)	200
8.9	Les splines (*)	201
8.10	Application à un problème de fiabilité	203
8.11	Conclusion	206
8.12	Notes et références	207
8.13	Exercices	208
9	Différentiation	213
9.1	Différences finies	216
9.2	Limitation de la précision	219
9.3	Origine de la limitation de la précision	228
9.4	Conditionnement	229
9.5	Implémentation	230
9.6	Extrapolation de Richardson (*)	232
9.7	Améliorer la précision (*)	235
9.8	Contre-exemple	239
9.9	Application : chute dans un fluide	241
9.10	Conclusion	244
9.11	Notes et références	244
9.12	Exercices	245
10	Moindres carrés linéaires	251
10.1	Hypothèses de calcul	252
10.2	Matrice de conception et modèles	255
10.3	Moindres carrés et norme euclidienne	257
10.4	Conditionnement	259
10.5	Méthode des équations normales	262
10.6	Conditionnement des équations normales	266
10.7	Décomposition QR (*)	268
10.8	Application : résistivité du cuivre	270
10.9	Conclusion	272
10.10	Notes et références	272
10.11	Exercices	273
11	Intégration	277
11.1	Conditionnement de l'intégrale	278
11.2	Règles de quadrature	280
11.3	Règles plus précises	285
11.4	Formules de Newton-Cotes	290
11.5	Le problème	293
11.6	Méthode composite	296

11.7	Quadrature adaptative (*)	299
11.8	Performance (*)	303
11.9	Traiter les difficultés	304
11.10	Fonction paramétrique	308
11.11	Applications : intégrer la loi normale	309
11.12	Conclusion	311
11.13	Notes et références	311
11.14	Exercices	312
12	Équations différentielles ordinaires	319
12.1	EDO et quadrature	321
12.2	Systèmes d'EDO	322
12.3	Méthodes à un pas	326
12.3.1	Introduction	326
12.3.2	Méthode d'Euler	327
12.3.3	Méthode de Runge, méthode de Heun	330
12.3.4	Les méthodes de Runge-Kutta	332
12.4	Erreurs	334
12.5	Méthodes adaptatives (*)	337
12.5.1	Principe d'une méthode adaptative	337
12.5.2	Méthodes de Runge-Kutta d'ordre 2 et 3	339
12.5.3	Contrôle du pas de discrétisation	340
12.6	Équations raides (*)	344
12.7	EDO paramétrée	344
12.8	Champ de vecteurs	345
12.9	Applications	347
12.9.1	Un modèle de frasil	347
12.9.2	L'attracteur de Lorenz	351
12.10	Conclusion	353
12.11	Notes et références	353
12.12	Exercices	354
13	Équations non linéaires	361
13.1	Conditionnement d'une équation non linéaire	362
13.2	Méthode de la bisection (dichotomie)	363
13.3	Implémentation	365
13.4	Nombre d'itérations de la bisection	368
13.5	Méthode de Newton	369
13.6	Convergence de la méthode de Newton	373
13.7	Méthode de la sécante	375
13.8	La condition d'arrêt	377
13.9	Application : gaz réels	378
13.10	Conclusion	380
13.11	Notes et références	380
13.12	Exercices	381
14	Optimisation	383
14.1	Propriétés des minima	384
14.2	Méthode du nombre d'or	387

Table des matières	15
14.3 Conditionnement de l'optimisation	390
14.4 Application : conduction de la chaleur	393
14.5 Conclusion	394
14.6 Notes et références	394
14.7 Exercices	394
15 Conclusion	399
Bibliographie	403
Index	413

Première partie

Prolégomènes au calcul
scientifique

Chapitre 1

À quoi sert le calcul numérique ?

Les objectifs de ce chapitre sont multiples. Premièrement, ce chapitre a pour but de définir le calcul numérique et sa position par rapport aux autres outils scientifiques qui lui sont souvent associés. Deuxièmement, il tente de présenter le calcul numérique en action, à travers ses applications et ses enjeux. Enfin, il tente de montrer l'importance que cette discipline, à la croisée des mathématiques appliquées, de l'informatique et de la plupart des sciences applicatives (par exemple la physique et la chimie), a prise dans le monde contemporain.

Pour mieux cerner les applications pratiques de la simulation numérique, on peut énumérer quelques exemples de problèmes d'ingénierie classiques. Il s'agit parfois de réduire le poids (kg), d'assurer la fiabilité ou de réduire le coût (€). Presque toutes les industries ont recours au calcul numérique : nous allons voir quelques-unes de ces applications.

1.1 Prévention de la commotion cérébrale chez les cyclistes

La conception des casques de cyclisme peut faire appel à la simulation numérique. Comme nous allons le voir, ce problème d'ingénierie révèle beaucoup de caractéristiques de la pratique du calcul numérique. En 2006, S. Kleiven a utilisé un modèle numérique de la masse cérébrale permettant de prévoir, dans le volume crânien, la pression en fonction de l'espace et du temps lorsqu'un crâne est soumis à un impact [83]. Ce modèle est fondé sur la méthode des éléments finis, une méthode numérique avancée souvent utilisée en mécanique. Dans la figure 1.1, le volume de la tête est décomposé en différentes sous-parties. En commençant par les parties les plus externes, ces volumes sont la boîte crânienne, les lobes, le cervelet et la moelle spinale¹. Puisque le comportement mécanique de chaque partie est spécifique (par exemple, la boîte crânienne se déforme moins facilement que la matière grise), il est nécessaire de spécifier localement les propriétés mécaniques de chaque partie. Dans ce contexte,

¹En ancienne nomenclature, la moelle spinale est nommée moelle épinière et le fluide cérébrospinal est nommé fluide céphalorachidien.

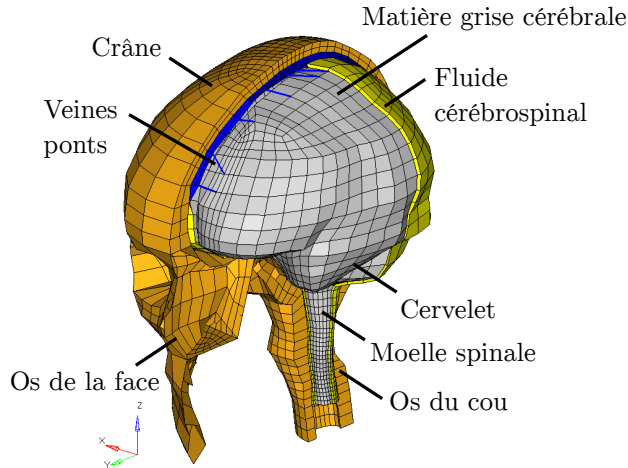


FIGURE 1.1 – Discretisation d'une tête humaine [84]. Le crâne, le liquide cérébrospinal et la matière grise. Illustration reproduite avec l'aimable autorisation de S. Kleiven.

on utilise le module d'Young qui caractérise la relation entre la déformation et la contrainte exercée [85]. Chaque volume est associé à un maillage spécifique, qui est une discrétisation de sa géométrie.

Ces travaux faisaient suite à des recherches initiées par le neurochirurgien suédois Hans von Holst à la fin des années 1990 pour identifier des améliorations dans les casques de protection individuels. Il a alors contacté l'École royale polytechnique (KTH) de Stockholm pour initier un programme de recherche sur ce thème, ce qui a mené Peter Halldin, un étudiant de cet institut, à commencer une thèse en biomécanique. C'est dans ce contexte que le travail de recherche de S. Kleiven débutera quelques années plus tard [85] et mènera à l'article de 2006 que nous sommes en train d'analyser.

Les impacts pris en compte dans l'étude sont caractérisés par leur direction frontale, occipitale (à l'arrière du crâne) ou latérale et par leur intensité. L'objectif final de ce travail était en particulier de pouvoir prévoir l'apparition de la commotion cérébrale liée à l'impact sur la tête et pouvant mener, dans certains cas, à des séquelles temporaires, définitives, ou à la mort.

Le chercheur souhaitait confronter les résultats du modèle numérique à la réalité. Il était évidemment hors de question de projeter des cobayes humains contre des obstacles pour observer l'apparition d'une commotion cérébrale. Pour résoudre cette difficulté, l'auteur a utilisé dans [83] des données issues de mesures expérimentales sur des cadavres. Toutefois, ces données doivent être corrigées pour prendre en compte le changement potentiel de rigidité de la matière cérébrale après la mort. De plus, les mesures pourraient être affectées par la présence d'air dans le fluide céphalo-rachidien, ce qui rend difficile les comparaisons avec la simulation numérique.

Dans [84, 82], l'auteur a utilisé des images de football américain captées par la télévision pour observer si on pouvait reproduire par la simulation les traumatismes observés sur les joueurs. Il a utilisé des indicateurs comme l'accélération angulaire, l'accélération de translation et la puissance d'impact pour observer si ces indicateurs sont de bons prédicteurs de la pression intracrânienne et des déformations associées à