MAO: CALCUL SCIENTIFIQUE

FILIPA CAETANO, ANTOINE LEVITT
VERSION PRÉLIMINAIRE, JUIN 2024

TABLE DES MATIÈRES

1. Programme	2
2. Introduction	2
2.1. Modélisation	3
2.2. Analyse numérique	3
2.3. Calcul scientifique	4
3. Arithmétique machine	4
4. Algèbre linéaire	5
4.1. Normes matricielles	5
4.2. Rayon spectral	5
4.3. Conditionnement et stabilité	6
5. Équations linéaires	7
5.1. Systèmes denses	7
5.2. La méthode de Richardson	8
5.3. Préconditionnement	9
5.4. Méthodes de Krylov	9
5.5. En pratique	9
TD/TP	9
6. Équations non-linéaires	10
6.1. Méthode de Newton	10
TD/TP	11
7. Optimisation	11
7.1. Optimisation sans contraintes	11
7.2. Optimisation sous contrainte	12
8. EDO	13
8.1. Méthode d'Euler	14
8.2. Commentaires et extensions	15
TD/TP	15
9. Introduction aux EDP	16
9.1. Modélisation	16
9.2. Classification	16
9.3. Solution par Fourier	17
TD/TP	17
10. EDP elliptiques	18
10.1. Caractère bien posé	18
10.2. Discrétisation	18
10.3. Analyse d'erreur	19
11. EDP d'évolution	20
11.1. Discrétisation en espace	21

1. PROGRAMME

Le présent document forme des notes de cours volontairement concises. On se référera aux ouvrages de référence pour plus de détails.

Équipe enseignante :

- Filipa Caetano
- Antoine Levitt

Contact : prénom.nom@universite-paris-saclay.fr.

Syllabus (très prévisionnel...) :

- AL Lundi 29 avril : CM : introduction, normes matricielles, conditionnement
- AL Mardi 30 avril : TD/TP : arithmétique machine, algèbre linéaire
- AL Jeudi 2 mai : TD/TP : algèbre linéaire
- AL Lundi 6 mai : CM/TD : systèmes linéaires
- AL Mardi 7 mai : TD/TP : systèmes linéaires
- FC Lundi 13 mai : CM : équations non-linéaires et optimisation
- FC Mardi 14 mai : TD/TP : équations non-linéaires et optimisation
- FC Jeudi 16 mai : TD/TP : équations non-linéaires et optimisation
- FC Mardi 21 mai : EDO
- AL Jeudi 23 mai : Partiel
- FC Lundi 27 mai : EDO
- FC Mardi 28 mai : EDO
- FC Jeudi 30 mai : EDP
- FC Lundi 3 juin : Transport
- FC Mardi 4 juin : Transport
- FC Jeudi 6 juin : Transport
- AL Lundi 10 juin : EDP
- AL Mardi 11 juin : EDP
- AL Jeudi 13 juin : EDP
- AL Lundi 17 juin : EDP
- AL Mardi 18 juin : EDP
- AL Jeudi 20 juin : EDP
- FC Lundi 24 juin : examen

Bibliographie (disponible à l'agrégation)

- Programme de l'agrégation : Francis Filbet, Analyse numérique
- EDO : Jean-Pierre Demailly, Analyse numérique et équations différentielles
- Algèbre linéaire, optimisation, EDP : Grégoire Allaire, analyse numérique et optimisation
- EDP, approximation numérique : Brigitte Lucquin, Équations aux dérivées partielles et leurs approximations

2. INTRODUCTION

La **modélisation** est la mise en équation d'un problème concret. L'**analyse numérique** est la conception et l'étude mathématique d'algorithmes de résolution approchée d'équations. Le **calcul scientifique** est l'art de la mise en oeuvre de ces algorithmes en pratique.

2.1. **Modélisation.** Quelles équations interviennent dans les questions suivantes ?

- (1) Quelle sera la température moyenne de la terre dans 50 ans ?

- (2) À quelle vitesse un avion décolle-t-il ?
- (3) Comment concevoir un avion indétectable par radar ?
- (4) Quel est l'angle de la molécule d'eau H_2O ?
- (5) Quel est le rendement d'une éolienne ?
- (6) ... d'un panneau solaire ?
- (7) Comment envoyer un satellite en orbite ?
- (8) Comment dimensionner un pont ?
- (9) Quelle est la consommation thermique d'un bâtiment ?
- (10) Comment investir un porte-feuille d'actifs ?
- (11) Quelle est la consommation énergétique d'un circuit électrique ?
- (12) Au-delà de quel seuil de contagiosité une maladie devient-elle une épidémie ?

2.2. Analyse numérique. L'analyse numérique est l'art de concevoir des algorithmes de calcul d'une approximation d'un objet bien défini, et de quantifier l'erreur commise. La plupart du temps, ce sont des procédés itératifs, et on n'obtient jamais la vraie solution. Ex : méthode babylonienne de calcul de $\sqrt{2}$ (il y a ≈ 4 millénaires) : $x_{n+1} = \frac{1}{2}(x_n + \frac{2}{x_n})$. Même en arithmétique exacte et en sachant calculer les racines, on ne peut pas calculer en un nombre fini les racines de polynômes de degré ≥ 5 (donc par exemple les valeurs propres de matrices de taille ≥ 5).

Exemple de problèmes dont on peut vouloir trouver une solution :

- (1) Système d'équations linéaire : $Ax = b$
- (2) Système d'équations non-linéaires : $f(x) = 0$
- (3) EDO : $\dot{x} = f(x)$
- (4) EDP : $\partial_t u(x, t) = f(t, x, u, \partial_x u, \dots)$
- (5) EDS : $dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dB_t$
- (6) Optimisation continue : minimiser $f(x)$, $x \in \mathbb{R}^N$ sous contrainte $g(x) = 0$, $h(x) \geq 0$
- (7) Optimisation combinatoire : minimiser $f(x)$, x à valeurs discrètes
- (8) Interpolation, extrapolation, intégration de fonctions

Comment trouver des algorithmes ? Souvent, l'analyse donne une première réponse : une preuve constructive = un algorithme. Vice versa, les algorithmes peuvent fournir des preuves.

Les preuves d'existence des théorèmes suivants sont-elles constructives ? Si oui, quelle est la méthode numérique associée ?

- (1) Existence de l'intégrale de Riemann d'une fonction continue
- (2) Existence de l'intégrale de Lebesgue d'une fonction mesurable
- (3) Bolzano-Weierstrass
- (4) Théorème des valeurs intermédiaires
- (5) Une fonction continue sur un compact admet un minimum
- (6) Tout polynôme a une racine (complexe)
- (7) Cauchy-Lipschitz
- (8) $x + Ax = b$ a une unique solution si $\|A\|_{\text{op}} < 1$
- (9) Point fixe de Banach
- (10) Point fixe de Brouwer

- (11) Théorème des fonctions implicites
- (12) Lax-Milgram
- (13) Théorème d'approximation de Weierstrass

L'analyse numérique ne peut calculer que des quantités bien *conditionnées*, c'est-à-dire bien définies et pas trop sensibles aux données. Exemple de problèmes mal conditionnés : $Ax = b$ avec A presque non inversible, déterminant, valeurs propres de matrices non symétriques, forme de Jordan, comportement en temps long d'EDO chaotiques, équation de la chaleur en temps négatif...

Chercher à résoudre un problème mal conditionné, c'est poser la mauvaise question. Souvent, la question finale est bien conditionnée, mais des quantités mal conditionnées apparaissent comme intermédiaires de calcul : il faut reformuler le problème (eg pivot de Gauss plutôt que formules de Cramer, forme de Schur plutôt que forme de Jordan...)

2.3. Calcul scientifique. Les points importants sont l'efficacité (en terme de temps d'ordinateur mais aussi de temps de développement !), la maintenabilité, la robustesse... Les technologies utilisées seront : langage de programmation Python, avec bibliothèques numpy (calculs sur tableaux), scipy (méthodes numériques), matplotlib (visualisation) et les notebooks.

Quelques conseils de base de programmation :

- Tester chaque partie du code indépendamment, ne pas se précipiter sur l'écriture d'un code complet
- Quand des bugs surviennent (inévitablement), ne pas relire vingt fois le code, mais tester méthodiquement si toutes les propriétés qu'on pense être vraies sont effectivement vraies
- Éviter le copier/coller
- Séparez le code dans des fonctions, et ne pas hésiter à faire des fonctions qui prennent en paramètre des fonctions
- Faire attention aux noms des variables. Utiliser des identificateurs descriptifs, par exemple `for it in range(Nt)`, `for ix in range(Nx)` : plutôt que `for i in range(N)`, `for j in range(M)`

3. ARITHMÉTIQUE MACHINE

On rappelle qu'un ordinateur stocke les nombres réels sous la forme (modulo détails techniques...) $a2^b$, où $a \in [1, 2[$ est la *mantisse*, stockée via ses décimales en binaire, et b est l'*exposant*, stocké comme un entier signé. Il en résulte que l'erreur entre un nombre x et sa représentation machine $\text{fl}(x)$ vérifie

$$|x - \text{fl}(x)| \leq \varepsilon|x|,$$

pour x dans une plage très large de valeurs. ε est appelé l'épsilon machine : il est de l'ordre de 2^{-N} , avec N le nombre de bits utilisés pour stocker a .

L'arithmétique sur ordinateur fonctionne comme si le résultat exact était calculé, puis arrondi au nombre représentable le plus proche : calculer $x+y$ sur un ordinateur calcule $\text{fl}(\text{fl}(x)+\text{fl}(y))$. Il résulte des arrondis intermédiaires que l'arithmétique machine est commutative, mais non associative : $x + (y + z)$ fait un arrondi intermédiaire pour $y + z$, alors que $(x + y) + z$ fait un arrondi pour $x + y$, ce qui ne revient pas au même.

Le point important, et parfois contre-intuitif, est que les ordinateurs stockent les nombres avec une *précision relative uniforme* : il y a beaucoup plus de nombres entre 0.001 et 0.002 qu'entre 100 et 100.001. En conséquence, la notion importante est celle de précision relative, et non absolue. Diviser par un petit nombre par exemple n'est, en tant que tel, pas très gênant, même si on commet ce faisant une grande erreur absolue : l'erreur relative reste de l'ordre de ε . En revanche, soustraire deux nombres proches est dangereux, car l'erreur relative sur le résultat sera importante.

- (1) Calculer $\sqrt{2}^2 - 2$ en python. Comment expliquer ce résultat ? Chercher sur internet sur combien de bits est représentée la mantisse en python (qui utilise des nombres flottants double précision par défaut), et vérifier la cohérence.
- (2) Par essai et erreur, donner l'ordre de grandeur du plus grand nombre en virgule flottante (attention, pas entier) représentable en python ? Cela est-il cohérent avec le nombre de bits sur lequel est représenté l'exposant ?
- (3) Calculer l'erreur commise par l'approximation

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

par exemple sur $f(x) = \sin x$ en $x = 1$, et la tracer en fonction de h . Quelle valeur de h donne le meilleur résultat ?

- (4) Que peut-t-il se passer si on calcule les racines de $ax^2 + bx + c = 0$ par la formule $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ quand $4|ac| \ll b^2$? Comment y remédier ?

4. ALGÈBRE LINÉAIRE

L'algèbre linéaire joue un rôle prépondérant en calcul scientifique : la résolution d'équations linéaires est un des rares îlots de solvabilité dans une mer de problèmes insolubles sous forme close. Beaucoup de problèmes (optimisation, résolution d'équations non-linéaires, etc) utilisent une résolution de systèmes linéaires comme étape intermédiaire. Plus généralement, le formalisme de l'algèbre linéaire fournit des outils puissants pour traiter tout type de phénomènes linéaires (par exemple, équations différentielles linéaires), qui forment la base de notre compréhension des systèmes même non-linéaires.

4.1. Normes matricielles. On rappelle que l'espace des matrices $\mathbb{R}^{M \times N}$ forme un espace vectoriel. Plusieurs normes naturelles peuvent être placées sur cet espace : les plus utiles sont la norme d'opérateur

$$\|A\|_{\text{op}} = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \max_{x \neq 0} \frac{\sqrt{\langle x, A^T A x \rangle}}{\sqrt{\langle x, x \rangle}}.$$

qui voit une matrice A comme un opérateur de $\mathbb{R}^M$ vers $\mathbb{R}^N$, et la norme de Frobenius

$$\|A\|_{\text{F}} = \sqrt{\sum_{i=1, \dots, M, j=1, \dots, N} |A_{ij}|^2} = \sqrt{\sum_{i=1, \dots, N} \|Ae_n\|^2} = \sqrt{\text{Tr}(A^T A)}$$

qui voit une matrice de $\mathbb{R}^{M \times N}$ comme un tableau de nombres, isomorphe à $\mathbb{R}^{MN}$. On vérifie que $\|A\|_{\text{F}}$ est égale à la racine de la somme des valeurs propres de $A^T A$ (toutes réelles et positives, et donc les racines sont appelées les valeurs singulières de A), alors que $\|A\|_{\text{op}}$ est égale à la racine de la plus grande. Quand on écrit $\|A\|$ sans précisions, c'est souvent la norme d'opérateur qui est sous-entendue. On vérifie facilement que $\|AB\|_{\text{op}} \leq \|A\|_{\text{op}} \|B\|_{\text{op}}$

4.2. Rayon spectral. Une autre notion utile est le **rayon spectral**

$$\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda|,$$

où $\sigma(A)$ est le spectre de A (ensemble de ses valeurs propres). Cette notion est particulièrement utile pour étudier le système dynamique linéaire en temps discret

$$x_{n+1} = Ax_n$$

avec x_0 fixé, qui est d'un grand intérêt dans l'étude de nombreux schémas numériques. On a

Théorème. $\lim_{n \rightarrow \infty} A^n = 0$ si et seulement si $\rho(A) < 1$.

Ce théorème est facile à démontrer si A est diagonalisable : on a avec les notations usuelles $A^n = PD^nP^{-1}$, et $(A^n \rightarrow 0) \Leftrightarrow (D^n \rightarrow 0) \Leftrightarrow (\lambda^n = 0 \text{ pour tout } \lambda \in \sigma(A))$. Quand A n'est pas diagonalisable, on peut procéder par perturbation, ou utiliser la forme normale de Jordan. On a même plus précisément le théorème de Gelfand

$$\rho(A) = \lim_{n \rightarrow \infty} \|A^n\|^{1/n},$$

cette limite étant bien définie et ne dépendant pas du choix de la norme.

Attention à ne pas confondre rayon spectral et norme matricielle, ainsi que valeurs propres et valeurs singulières : les valeurs propres (indépendantes du choix de norme) déterminent le comportement asymptotique des applications répétées de A , alors que les valeurs singulières (dépendantes de la norme) déterminent l'expansion de la norme par une seule application de A . Un bon exemple à garder en tête est la matrice

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},$$

de valeurs propres nulles (donc de rayon spectral nul) mais de valeurs singulières non-nulles. Cet exemple montre également que $\rho(A)$ n'est pas une norme matricielle.

Un autre bon exemple est

$$(1) \quad A = \begin{pmatrix} 0.9 & 1000 \\ 0 & 0.9 \end{pmatrix}$$

La dynamique de A^n (exercice) montre une décroissance à l'infini comme 0.9^n (comme prédit par la formule de Gelfand), mais uniquement après une phase transitoire de grande amplitude.

Un cas particulier où tout est plus simple est celui des matrices symétriques (et plus généralement normales, c'est-à-dire des matrices commutant avec leur transposée, qui se diagonalisent en base orthonormée). Dans ce cas, les valeurs singulières sont les valeurs absolues des valeurs propres, et on vérifie aisément en diagonalisant A que $\rho(A) = \|A\|_{\text{op}}$.

4.3. Conditionnement et stabilité. Imaginons que l'on souhaite résoudre $Ax = b$, mais que, pour une raison ou une autre, notre membre de droite¹ est entaché d'une incertitude (ne serait-ce que dûe à sa représentation numérique). Au lieu de résoudre $Ax = b$, on résoud en fait $A(x + \Delta x) = (b + \Delta b)$. Quelle est la taille de Δx en fonction de celle de Δb ? On a aisément

$$\Delta x = A^{-1}\Delta b$$

et donc

$$\|\Delta x\| \leq \|A^{-1}\| \|\Delta b\|.$$

C'est une première réponse, qui donne une première indication sur la sensibilité aux erreurs : plus A^{-1} est grande, plus x variera. Cette réponse n'est cependant pas très satisfaisante car elle n'est pas invariante par échelle : le système $2Ax = 2b$ n'est pas intrinsèquement plus sensible aux erreurs que $Ax = b$. Pour remédier à cela, on préfère travailler en erreur relative et écrire

$$\frac{\|\Delta x\|}{\|x\|} \leq \|A\| \|A^{-1}\| \frac{\|\Delta b\|}{\|b\|}.$$

Le nombre $\|A\| \|A^{-1}\|$, appelé **conditionnement** $\kappa(A)$ de la matrice A (égal au ratio de la plus grande sur la plus petite valeur singulière) est adimensionnel, et quantifie la perte de précision relative liée à la résolution du système linéaire. Bien sûr, ce n'est qu'une majoration dans le pire cas, mais sa simplicité en fait quand même un concept utile.

1. On peut faire la même analyse pour une erreur sur A .

Plus généralement, le *conditionnement* d'une opération (non nécessairement linéaire) quantifie l'amplification d'erreur relative intrinsèque à l'opération. Par exemple, le calcul du déterminant d'une matrice est mal conditionné en général, parce qu'une petite perturbation relative de la matrice $\begin{pmatrix} \varepsilon & 0 \\ 0 & 1/\varepsilon \end{pmatrix}$ pour ε petit produit une grande perturbation relative de son déterminant. C'est une notion à distinguer de celle de **stabilité** d'un algorithme qui réalise l'opération en question, et qui réfère à sa capacité à obtenir des résultats aussi bons possibles (étant donné son conditionnement) en présence d'erreur d'arrondi. L'élimination Gaussienne sans pivot est un algorithme potentiellement instable, même si le problème lui-même est bien conditionné.

TD/TP

- (1) Calculer le conditionnement de $\begin{pmatrix} \varepsilon & 0 \\ 0 & 1/\varepsilon \end{pmatrix}$ et son déterminant. Même question sur $0.5I_N$, l'identité en dimension N . Pourquoi le déterminant n'est-il pas une bonne mesure de l'inversibilité d'un système ?
- (2) Appliquer la méthode du pivot de Gauss à la main sur le système

$$\begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}$$

et en déduire que l'algorithme du pivot de Gauss sans échange est potentiellement instable.

- (3) Déterminer la matrice correspondant à la discrétisation de l'équation $-u'' = f$ par différences finies sur $[0, 1]$ avec condition aux limites de Dirichlet. Montrer qu'elle est symétrique définie positive. Calculer analytiquement ses valeurs et vecteurs propres ainsi que son conditionnement.

5. ÉQUATIONS LINÉAIRES

On s'intéresse au système de N équations à N inconnues $Ax = b$. Si A est inversible, ce système a une unique solution. Sinon, il possède soit une infinité de solutions (paramétrisées par le noyau de A), soit aucune (alternative de Fredholm). La distinction se fait selon que $b \in \text{ran}(A) = \ker(A^T)^\perp$ ou non (plus prosaïquement : si une combinaison linéaire de lignes donne 0 à gauche de l'équation, on doit avoir 0 à droite). Dans la suite, on se place dans le cas où A est inversible et $Ax = b$ admet une unique solution x_* .

5.1. Systèmes denses. Il existe des formules explicites pour l'inverse d'une matrice, en fonction de déterminants de sous-matrices : les formules de Cramer. Ces formules sont algébriquement plaisantes (par exemple, montrent facilement que l'inverse d'une matrice à coefficients rationnels est à coefficients rationnels), mais analytiquement catastrophiques : les déterminants coûtent cher à calculer, et sont numériquement instables.

La méthode standard pour résoudre $Ax = b$ (numériquement comme à la main) est le pivot de Gauss : on soustrait un multiple la première ligne aux autres de sorte à annuler le coefficient de la première variable. On itère ainsi jusqu'à aboutir à une équation triviale pour la dernière variable, et on remonte ensuite jusqu'à la première par substitutions successives. Techniquement, il faut s'assurer que le coefficient de x_n à la n -ième étape (le pivot) ne s'annule pas (ou n'est pas trop petit) : cela peut être fait en échangeant des lignes ou des colonnes.

Cette méthode est souvent exprimée sous la forme d'une factorisation de matrices. Si on ignore l'échange de lignes ou colonnes, l'algorithme amène A à avoir une forme triangulaire supérieure par manipulations "triangulaires" successives de lignes (c'est-à-dire qu'à l'étape n , on ne modifie que les lignes restantes). On peut donc écrire $E_N E_{N-1} \dots E_1 A = U$, où les E_n sont triangulaires inférieures et U est triangulaire supérieure. Les matrices triangulaires inférieures étant stables par

multiplication et inversion, on peut aussi écrire $A = LU$ avec L triangulaire inférieure. Cette forme est pratique parce qu'elle permet de calculer facilement l'inverse

À l'étape n de l'algorithme, il faut modifier les $N - n + 1$ éléments des $N - n + 1$ lignes restantes : la complexité totale est donc $O(N^3)$. En pratique, on peut résoudre en quelques dizaines de secondes sur un PC de bureau des systèmes de taille 10000. Calcul rapide d'ordre de grandeur : 1Ghz = 10^9 cycles par seconde ; un processeur à 4 Ghz de 8 coeurs qui effectue 4 opérations par cycle fait donc 10^{12} opérations en à peu près 10s. Un modèle de climat typique utilise plusieurs millions d'inconnues, et un pas de temps de l'ordre de l'heure : résoudre un système linéaire par cette méthode à chaque pas d'une simulation de climat de 50 ans prendrait donc de l'ordre de 10000 ans sur un ordinateur personnel : même sur des supercalculateurs, c'est impossible.

5.2. La méthode de Richardson. Très souvent, les matrices provenant des problèmes réels (par exemple, équations aux dérivées partielles, modèles d'optimisation, etc) sont *creuses* : elles ne contiennent que $O(1)$ éléments non-nuls par ligne. L'élimination de Gauss ne permet typiquement pas d'utiliser cette structure de façon optimale : lors des éliminations successives, la matrice se remplit d'éléments non-nuls.

Une idée plus productive pour résoudre les très grands systèmes linéaires est d'utiliser une méthode itérative, dont le prototype est la méthode de Richardson

$$(2) \quad x_{n+1} = x_n - \alpha(Ax_n - b)$$

pour un pas α . La quantité $Ax - b$ est le **résidu** de l'équation $Ax = b$, et quantifie à quel point on est proche de la solution : en effet, on a la relation fondamentale erreur-résidu $Ax - b = Ax - Ax_* = A(x - x_*)$, qui montre que $\|x - x_*\| \leq \|A^{-1}\| \|Ax - b\|$.

L'intérêt de (2) est de n'avoir besoin que d'effectuer un produit matrice-vecteur à chaque itération. Si A n'a que $O(1)$ entrées non-nulles par ligne, ce produit peut être fait en $O(N)$ opérations. Clairement, si l'itération converge, elle converge vers la solution du problème x_* . Cette itération est très naturelle : étant donné un x candidat, on ne peut pas calculer en général grand chose d'autre que le résidu, et faire des combinaisons linéaires.

Pour étudier la convergence de la relation de récurrence inhomogène (2), la première étape est de se ramener au cas homogène en posant $e_n = x_n - x_*$:

$$e_{n+1} = (1 - \alpha A)e_n$$

d'où $e_n = (1 - \alpha A)^n e_0$. L'itération convergera si les valeurs propres de $1 - \alpha A$ sont plus petites que 1 en module. C'est possible pour α suffisamment petit si toutes les valeurs propres de A sont de partie réelle strictement positive (ou, en changeant le signe de α , négative). On peut préciser cette convergence par exemple dans le cas où A est symétrique définie positive, de valeurs propres $0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_N$. Dans ce cas, en projetant sur le i -ième mode propre v_i , on obtient

$$\langle v_i, e_n \rangle = (1 - \alpha \lambda_i)^n \langle v_i, e_0 \rangle.$$

On voit que pour optimiser la convergence, il faut choisir α de façon à ce que $\max_{i=1, \dots, N} |1 - \alpha \lambda_i|$ soit minimal. On voit aisément que le choix optimal est

$$\alpha_{\text{opt}} = \frac{2}{\lambda_1 + \lambda_N},$$

et que dans ce cas le taux de convergence est

$$(3) \quad r(\alpha_{\text{opt}}) = 1 - \frac{2\lambda_1}{\lambda_1 + \lambda_N} = 1 - \frac{2}{1 + \kappa(A)}$$

Plus le conditionnement $\kappa(A)$ est grand, plus $r(\alpha_{\text{opt}})$ est proche de 1, ce qui indique une convergence lente.

5.3. Préconditionnement. On a vu que plus un problème est mal conditionné, plus il est difficile à résoudre par la méthode de Richardson. C'est en fait parce que le résidu $r = A(x - x_*)$ (ce qu'on peut calculer) est une image déformée de l'erreur $e = x - x_*$ (ce qu'on voudrait calculer). Comment peut-on travailler sur le résidu r pour l'amener à être plus proche de e ? Il arrive parfois qu'on dispose d'une approximation P de la matrice A qui soit simple, au sens où on peut inverser l'approximation aisément. On peut alors utiliser la méthode de Richardson avec $\tilde{r} = P^{-1}r$ dans l'espoir que $\tilde{r} = P^{-1}Ae$ soit plus fidèle à e que r :

$$x_{n+1} = x_n - \alpha P^{-1}(Ax - b)$$

La même analyse que précédemment montre que si $P^{-1}A$ a des valeurs propres proches de 1 (ou plus généralement d'une constante), la méthode de Richardson preconditionnée convergera rapidement. P est appelé un **préconditionneur**.

5.4. Méthodes de Krylov. La méthode de Richardson ci-dessus est intéressante pédagogiquement car simple à analyser, mais elle est très sous-optimale, parce que c'est une méthode stationnaire (la même relation, avec les mêmes coefficients, est appliquée à chaque itération) : elle n'est pratiquement jamais utilisée en pratique. Des méthodes plus sophistiquées mélangent l'information de plusieurs itérées précédentes pour obtenir une meilleure approximation : citons la méthode du gradient conjugué (applicable aux matrices symétriques définies positives) et la méthode GMRES (applicable aux matrices quelconques). Leur analyse fait appel à la théorie de l'approximation polynomiale. Ces méthodes, combinées avec un bon preconditionneur, sont celles utilisées en pratique pour la résolution de grands problèmes linéaires.

5.5. En pratique. En pratique, si la matrice est de petite taille ($N \leq 1000$), on utilisera une méthode directe de type pivot de gauss (`numpy.linalg.solve`). Si la matrice est grande et creuse (structures de données de matrices creuses, `scipy.sparse`), on utilisera une méthode de pivot de gauss adaptée (`scipy.sparse.linalg.spsolve`), ou une méthode de Krylov (par exemple `scipy.sparse.linalg.gmres`).

TD/TP.

- (1) On souhaite trouver une courbe qui reproduit un jeu de données $(x_n, y_n)_{n=1, \dots, N}$. Pour cela, on va se donner un ensemble de fonctions de base $\phi_p(x)$ (par exemple des polynômes) et chercher une courbe $\phi(x) = \sum_{p=1}^P \alpha_p \phi_p(x)$ qui minimise la fonction objectif

$$E = \sum_{n=1}^N \left(y_n - \sum_{p=1}^P \alpha_p \phi_p(x_n) \right)^2$$

Par exemple, si $\phi_1(x) = 1$, $\phi_2(x) = x$, on cherche à faire passer une droite.

- Reformuler le problème sous forme vectorielle en l'inconnue $\alpha \in \mathbb{R}^p$.
- Écrire les conditions d'optimalité
- Le coder dans le cas de la régression polynomiale ($\phi_p(x) = x^{p-1}$) et le tester sur des données régulières et bruitées. Que se passe-t-il quand P est grand?
- Refaire avec la fonction objectif

$$E = \sum_{n=1}^N \left(y_n - \sum_{p=1}^P \alpha_p \phi_p(x_n) \right)^2 + \eta \sum_{p=1}^P \alpha_p^2.$$

- (2) Montrer la formule de Neumann : si $\|A\|_{\text{op}} < 1$, alors $1 - A$ est inversible et $(1 - A)^{-1} = \sum_{k=0}^{\infty} A^k$. À quelle méthode numérique peut-on identifier cette formule?

- (3) Écrire explicitement la méthode de Richardson préconditionnée dans le cas où $\alpha = 1$ et $P = D$, la diagonale de A . Montrer que la méthode se ramène à la méthode de Jacobi $Dx_{n+1} + (A - D)x_n = b$
- (4) Montrer que si A est à diagonale strictement dominante, au sens où pour tout $i = 1, \dots, N$,

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|,$$

la méthode de Jacobi converge. On pourra montrer et utiliser le théorème de Gershgorin : les valeurs propres d'une matrice A sont incluses dans l'union de disques $\cup_{i=1, \dots, N} B(a_{ii}, \sum_{j \neq i} |a_{ij}|)$

- (5) Combien d'itérations environ sont nécessaires pour résoudre un système linéaire de matrice

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 100 \end{pmatrix}$$

à une précision proche de la précision machine avec la méthode de Richardson à pas optimal ? Le vérifier numériquement.

- (6) On considère la discrétisation de l'équation de Poisson $-u'' = f$ par différences finies sur $[0, 1]$ avec conditions aux limites de Dirichlet.
- (a) Résoudre l'équation numériquement avec $f(x) = \sin(\pi x)$.
- (b) Déterminer théoriquement et numériquement les vecteurs et valeurs propres de la matrice A apparaissant dans la discrétisation par différences finies.
- (c) Comment croît le nombre d'itérations de la méthode de Richardson avec le nombre de points de discrétisation ? Le vérifier en pratique.
- (d) Que se passe-t-il si on augmente la taille du domaine, à pas de discrétisation fixé ? Même question pour $-u'' + u = f$.
- (e) Résoudre analytiquement par fonction de Green l'équation $-u'' = f$.
- (f) Utiliser cette solution analytique pour construire un préconditionneur P de l'équation $-u'' + e^x u = f$. Vérifier que la méthode préconditionnée converge plus rapidement, et calculer numériquement les valeurs propres de $P^{-1}A$.

6. ÉQUATIONS NON-LINÉAIRES

6.1. Méthode de Newton. Supposons qu'on veuille résoudre une équation non-linéaire $f(x) = 0$, où f est une fonction régulière de $\mathbb{R}^N$ vers $\mathbb{R}^N$ (mais les arguments ci-dessous sont également valables dans un espace de Banach). Supposons qu'on ait une estimation x_n d'un zéro de f ; comment en obtenir une meilleure ? On peut linéariser f autour de x_n ,

$$f(x) \approx f(x_n) + f'(x_n)(x - x_n),$$

puis chercher un zéro de cette fonction affine. Cela mène directement à l'itération de Newton

$$x_{n+1} = x_n - f'(x_n)^{-1}f(x_n).$$

Cette itération converge localement vers des zéros non-dégénérés :

Théorème. Soit $f : \mathbb{R}^N \rightarrow \mathbb{R}^N$ une fonction de classe C^3 , et x_* un zéro de f non-dégénéré, c'est-à-dire que $f'(x_*)$ est inversible. Alors il existe $r > 0$ tel que, pour tout x_0 tel que $\|x_0 - x_*\| \leq r$, l'itération de Newton démarrée de x_0 converge vers x_* . Cette convergence est quadratique, c'est-à-dire qu'il existe $C > 0$ tel que

$$\|x_{n+1} - x_*\| \leq C \|x_n - x_*\|^2$$

Démonstration. Soit

$$G(x) = x - f'(x)^{-1}f(x)$$

l'application représentant une itération. On a $G(x_*) = x_*$, et on calcule facilement que G est C^2 dans un voisinage de x_* , avec $G'(x_*) = 0$. Il existe donc un voisinage de x_* et $C > 0$ tel que, pour tout x dans ce voisinage,

$$(4) \quad G(x) - x_* = G(x) - G(x_*) \leq \|G'(x_*)(x - x_*)\| + C\|x - x_*\|^2 = C\|x - x_*\|^2.$$

Cela montre que, en restreignant le voisinage à avoir un rayon $r < 1/C$, ce voisinage est stable par G , et $\|x_{n+1} - x_*\| \leq rC\|x_n - x_*\|$: la méthode de Newton converge. La convergence quadratique suit par (4). $\square$

En pratique, la méthode de Newton est souvent très coûteuse ; on l'approxime par des méthodes approchant $f'(x)^{-1}$ de façon plus ou moins explicite, comme les méthodes de Broyden. Le choix le plus simple $x_{n+1} = x_n + f(x_n)$ est connu sous le nom de méthode de Picard, et converge localement si $1 + f'(x)$ a des valeurs propres plus petites que 1 en module.

TD/TP.

- (1) Montrer une variante du théorème de convergence de la méthode de Newton : si f est C^3 dans une boule de rayon δ autour de x_0 et $f'(x_0)$ est inversible, alors il existe $\varepsilon > 0$ tel que, si $\|f(x_0)\| \leq \varepsilon$, la méthode de Newton converge vers un zéro de f . L'utiliser pour montrer le théorème des fonctions inverses.
- (2) Comment modifier la preuve ci-dessus si f n'est que de classe C^2 ? Que peut-on dire si f n'est que de classe C^1 ?
- (3) Tracer la fonction W de Lambert, définie par $w(x)e^{w(x)} = x$
- (4) Montrer que, pour une fonction $f : \mathbb{C} \rightarrow \mathbb{C}$ analytique, la méthode de Newton $z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$ est identique à celle qu'on obtiendrait si on considérait $\mathbb{C}$ comme $\mathbb{R}^2$. Tracer numériquement les bassins d'attraction (l'ensemble des conditions initiales pour lesquelles la méthode de Newton converge vers chacune des racines) de l'équation $z^3 = 1$.
- (5) Comment pourriez-vous utiliser le résultat de la question 1 pour démontrer mathématiquement l'existence d'une solution à une équation non-linéaire ?

7. OPTIMISATION

7.1. Optimisation sans contraintes. On considère le problème

$$\min_{x \in \mathbb{R}^N} f(x)$$

avec f une fonction régulière. On rappelle qu'on a existence d'un minimum global si f est coercive (est infinie à l'infini), et unicité si f est strictement convexe (il existe $c > 0$ tel que $\nabla^2 f(x) \geq c$ pour tout $x \in \mathbb{R}^N$). À un minimum x_* , on a $\nabla f(x_*) = 0$ et $\nabla^2 f(x_*) \geq 0$.

Le problème de trouver le minimiseur global d'une fonction non-convexe est extrêmement dur, et on doit souvent se contenter en pratique d'un minimiseur local, et d'une méthode itérative. Au point x , dans quelle direction h aller pour diminuer f ? On a

$$f(x+h) = f(x) + \nabla f(x) \cdot h + O(\|h\|^2)$$

donc la direction de plus forte descente est $h = -\nabla f(x)$. Cela suggère la méthode du gradient :

$$(5) \quad x_{n+1} = x_n - \alpha \nabla f(x_n)$$

avec un pas α à choisir.

Théorème. Si f est C^2 et vérifie $c \leq \nabla^2 f(x) \leq C$ pour tout $x \in \mathbb{R}^N$ pour deux constantes $c, C > 0$. Alors il existe α_0 tel que pour tout pas $\alpha \leq \alpha_0$ et point de départ $x_0 \in \mathbb{R}^N$, l'itération (5) converge vers l'unique minimum global de f .

Démonstration. On rappelle que $c \leq \nabla^2 f(x)$ implique que f est strictement convexe, donc coercive. Il existe un unique minimiseur x_* , unique solution de $\nabla f(x) = 0$.

On a

$$f(x_{n+1}) = f(x_n) - \alpha \|\nabla f(x_n)\|^2 + O(C\alpha^2 \|\nabla f(x_n)\|^2).$$

En choisissant α_0 suffisamment petit, on a

$$f(x_{n+1}) \leq f(x_n) - \frac{\alpha}{2} \|\nabla f(x_n)\|^2,$$

d'où il suit que $f(x_n)$ a une limite, et que $\nabla f(x_n) \rightarrow 0$. Ce n'est pas encore suffisant pour obtenir la convergence. Tous les points d'accumulation de x_n vérifient $\nabla f(x) = 0$, et sont donc égaux à x_* . Supposons maintenant que x_n ne converge pas vers x_* ; cela implique qu'il existe $\varepsilon > 0$ et une sous-suite qui reste à une distance ε de x_* . Par compacité des sous-ensembles de niveau de f (conséquence de la coercivité), on peut en extraire encore une sous-suite qui converge; comme la limite ne peut être que x_* , c'est une contradiction. $\square$

Cette preuve utilise l'hypothèse très forte que f est strictement convexe. Dans le cas contraire, mais quand f reste coercive, on a génériquement convergence, par exemple quand les points critiques de f sont non dégénérés.

Comment choisir α en pratique? Le choix d'un pas fixe est sous-optimal. Il existe une grande variété d'algorithmes de recherche linéaire (c'est-à-dire de choix de α). Par exemple, un algorithme consiste à l'étape n à choisir un pas initial, puis à le réduire tant que la condition d'Armijo (de "descente suffisante")

$$f(x_n - \alpha \nabla f(x_n)) \leq f(x_n) - \alpha \beta \|\nabla f(x_n)\|^2$$

n'est pas satisfaite, pour une valeur de $\beta \in]0, 1[$ fixée.

Quelle est la vitesse de convergence de la méthode de gradient? Supposons qu'on démarre (ou qu'on arrive après plusieurs pas de la méthode) au voisinage d'un minimum local non-dégénéré. Alors on peut linéariser l'équation en calculant la jacobienne de l'application d'itération $x \mapsto f(x) - \alpha \nabla f(x)$, qui vaut $1 - \alpha \nabla^2 f(x)$. On retrouve alors exactement l'analyse de la méthode de Richardson, avec $A = \nabla^2 f(x)$. La conclusion est, comme pour la méthode de Richardson, que le taux de convergence se dégrade avec l'augmentation du conditionnement de A .

Que veut dire cette convergence en pratique? On voit sur l'exemple de $A = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$ (faire un dessin, en représentant les lignes de niveau de f) que la méthode converge en zigzag lorsque a et b sont différents. Pour corriger ce zigzag, trois solutions :

- Le préconditionnement $x_{n+1} = x_n - \alpha P^{-1} \nabla f(x_n)$;
- L'utilisation de la méthode de Newton $x_{n+1} = x_n - \nabla^2 f(x_n)^{-1} \nabla f(x_n)$, très efficace localement, mais coûteuse et convergeant également vers les points critiques non minima;
- L'utilisation de méthodes de quasi-Newton visant à approximer $\nabla^2 f(x)^{-1}$ sans le calculer, comme le gradient conjugué ou l'algorithme BFGS.

7.2. Optimisation sous contrainte. On rappelle dans un cadre simple la méthode des multiplicateurs de Lagrange. Supposons par simplicité qu'on veuille optimiser $f(x)$ sous la contrainte $g(x) = 0$. Supposons également que g soit C^2 et $\nabla g(x) \neq 0$ dès que $g(x) = 0$, ce qui signifie que la contrainte $g(x)$ définit localement une sous-variété de dimension $N - 1$. Le plan tangent en x est défini par l'équation $\nabla g(x)^T h = 0$.

Si on est à un minimum x_* du problème, on doit avoir $f(x) \geq f(x_*)$ pour tout y proche de x tel que $g(x) = 0$. Soit h dans le plan tangent en x_* . En appliquant le théorème des fonctions implicites à l'équation $g(x_* + th + u\nabla g(x_*)) = 0$ en u à paramètre t , on obtient une courbe $u(t)$ qui est C^1 et qui vérifie $u(0) = 0, u'(0) = 0$. Cela donne un chemin C^1 sur la variété $x(t) = x_* + th + u(t)h_\perp$ tel que $x(0) = x_*$ et $x'(0) = h$. Il suit que $\nabla f(x)^T h = 0$ pour tous ces h . On a donc que $\nabla f(x_*)$ est orthogonal à l'orthogonal de $\nabla g(x_*)$, c'est-à-dire qu'il existe $\lambda \in \mathbb{R}$ tel que $\nabla f(x_*) = \lambda \nabla g(x_*)$. Ce λ est appelé multiplicateur de Lagrange. Géométriquement, il traduit simplement le fait que le gradient est orthogonal au plan tangent (donc appartient à l'espace normal, c'est-à-dire $\text{Im}(\nabla g(x_*))^\perp$).

Pour résoudre un problème d'optimisation sous contrainte, on peut s'attaquer aux conditions d'optimalité $\nabla f(x) = \lambda \nabla g(x), g(x) = 0$ vu comme un système non-linéaire de $N + 1$ équations en $N + 1$ variables, et utiliser par exemple la méthode de Newton, mais on n'obtient que la convergence locale, et vers tous les points critiques, pas nécessairement minima. On peut également utiliser l'algorithme du gradient projeté : on se dirige dans la direction du gradient projeté sur le plan tangent, puis on se ramène sur la variété des contraintes

- (1) Donner la solution du problème $\min x^2 + y^2$ sous contrainte $2x + y = 1$
- (2) Calculer le gradient et la Hessienne de $x \mapsto \frac{1}{2}x^T A x - x^T b$ avec A symétrique définie positive
- (3) Montrer le théorème d'approximation par des matrices de rang 1 (cas particulier du théorème d'Eckart-Young) : pour toute matrice $A \in \mathbb{R}^{N \times M}$,

$$\min_{u \in \mathbb{R}^N, v \in \mathbb{R}^M} \|A - uv^T\|_F = \sigma_2(A)$$

où $\sigma_2(A)$ est la deuxième plus grande valeur singulière de A , avec le minimum atteint lorsque u et v sont les vecteurs singuliers à gauche et à droite associés à la plus grande valeur singulière (c'est-à-dire que $A^T A u = \sigma_1(A)^2 u, A A^T v = \sigma_1(A)^2 v$).

- (4) Appliquer la méthode du gradient à la fonction $\frac{1}{2}x^T A x - x^T b$ avec A symétrique définie positive, et montrer qu'elle s'identifie à l'itération de Richardson
- (5) Appliquer la méthode du gradient à la fonction ci-dessus pour $A = \begin{pmatrix} 10 & 0 \\ 0 & 1 \end{pmatrix}, b = 0$, et tracer les itérés
- (6) Donner l'algorithme du gradient projeté pour une fonction $f(x)$ soumise à la contrainte $\|x\|^2 = 1$
- (7) Implémenter la méthode du gradient projeté pour le problème de séparation de source (voir code fourni) : on se donne d sources de longueur N stockée dans une matrice $X \in \mathbb{R}^{N \times d}$ (centrée et normalisée), on cherche un vecteur de poids $w \in \mathbb{R}^d$ avec $\|w\| = 1$ qui minimise

$$f(x) = \mathbb{E}(G(Xw))$$

avec $G(x) = \log(\cosh(x))$ (qui mesure la déviation par rapport à la gaussianité).

8. EDO

On considère l'équation

$$\dot{x} = f(x)$$

avec $f : \mathbb{R}^N \rightarrow \mathbb{R}^N$ pour simplicité. La théorie s'étend au cas non-autonome ($f(x, t)$), au cas où x vit dans un espace de Banach, et aux équations d'ordre plus élevé (par exemple via le changement de variable $y_n = \dot{x}_n$).

On rappelle le théorème de Cauchy-Lipschitz : si f est localement Lipschitz (par exemple, C^1) en x_0 , on a existence et unicité pour un temps fini. Si f est globalement Lipschitz (ou qu'elle est

localement Lipschitz et qu'on arrive à montrer que la solution reste bornée), la solution existe pour tout temps.

8.1. Méthode d'Euler. La méthode d'Euler est

$$x_{n+1} = x_n + \Delta t f(x_n)$$

Théorème. Soit $f : \mathbb{R}^N \rightarrow \mathbb{R}^N$ uniformément C^1 (i.e. f est globalement C^1 et sa différentielle est uniformément bornée), $x_0 \in \mathbb{R}^N$, $(x(t))_{t \in \mathbb{R}}$ l'unique solution de $\dot{x} = f(x)$, $x(0) = x_0$, et $(x_n)_{n \in \mathbb{N}}$ la suite définie par

$$x_{n+1} = x_n + \Delta t f(x_n).$$

Alors, pour tout $T \in \mathbb{R}$, il existe $C_T > 0$ tel que, pour tout $N \in \mathbb{N}$, si $\Delta t = T/N$, alors

$$\sup_{n=0, \dots, N} |x_n - x(n\Delta t)| \leq C_T \Delta t$$

Démonstration. On définit l'erreur de troncature locale (erreur de consistance, par mauvaise traduction de l'anglais "consistent", qui signifie "cohérent")

$$t_n = x(t_{n+1}) - (x(t_n) + \Delta t f(x(t_n))),$$

à différencier de l'erreur

$$e_n = x_n - x(t_n)$$

(faire le dessin fondamental : $x(t_n), x(t_{n+1}), x_n, x_{n+1}, t_n, e_n$). Avec un développement limité à l'ordre 2 de $x(t)$, on montre aisément que $|t_n| \leq C \Delta t^2$ avec $C > 0$ ne dépendant pas de Δt .

On a la relation de récurrence

$$\begin{aligned} e_{n+1} &= x_{n+1} - x(t_{n+1}) \\ &= (x_n + \Delta t f(x_n)) - (x(t_n) + \Delta t f(x(t_n))) - t_n \\ |e_{n+1}| &\leq (1 + \Delta t L)|e_n| + |t_n| \end{aligned}$$

où L est la constante de Lipschitz de f , qui est une estimée de stabilité.

On résoud maintenant l'inégalité récurrente ("lemme de Gronwall discret")

$$a_{n+1} \leq \alpha a_n + \beta$$

en montrant que a_n ne peut pas croître plus vite que dans le cas d'égalité. Soit $b_{n+1} = \alpha b_n + \beta$. On pose la différence $d_n = a_n - b_n$, on a $d_{n+1} \leq \alpha d_n$ donc $d_n \leq 0$.

Résolvons maintenant la récurrence inhomogène $b_{n+1} = \alpha b_n + \beta$. On utilise la méthode de variation de la constante : $b_n = c_n \alpha^n$. On remplace :

$$c_{n+1} = c_n + \beta \alpha^{-(n+1)}$$

d'où

$$\begin{aligned} c_n &= b_0 + \sum_{k=1}^n \beta \alpha^{-k} \\ b_n &= \alpha^n b_0 + \beta \sum_{k=1}^n \alpha^{n-k} \\ &= \alpha^n b_0 + \beta \sum_{k=0}^{n-1} \alpha^k \\ &= \alpha^n b_0 + \beta \frac{\alpha^n - 1}{\alpha - 1} \end{aligned}$$

Avec $\alpha = (1 + \Delta t L)$, $\beta = C\Delta t^2$, on obtient

$$|e_n| \leq C\Delta t((1 + \Delta t L)^n - 1) \leq C\Delta t e^{Lt_n}$$

□

8.2. Commentaires et extensions.

- D'un point de vue mathématique, la méthode d'Euler peut être utilisée comme preuve alternative d'existence de solution : schématiquement, on interpole linéairement les points x_n pour construire une solution continue approchée $\tilde{x}_N(t)$, et on utilise une méthode de compacité (théorème d'Arzela Ascoli) pour extraire une sous-suite convergente.
- **Consistance** (on résout l'équation de façon approchée à chaque étape) + **stabilité** (amplification d'erreur raisonnable) = convergence
- La notion de stabilité est la même que celle utilisée pour prouver la stabilité d'une EDO par rapport à des erreurs sur x_0 et/ou f .
- Le schéma de preuve marche aussi pour d'autres méthodes, comme celle ci-dessus : on a besoin de borner l'erreur de consistance, et que la fonction d'incrément $x_n \rightarrow x_{n+1}$ soit Lipschitzienne de constante $1 + \Delta t L'$. Par exemple, pour une méthode de type Runge-Kutta à l'ordre 2

$$x_{n+1} = x_n + \frac{\Delta t}{2}(f(x_n) + f(x_n + \Delta t f(x_n))),$$

en refaisant le même raisonnement, on obtiendrait une erreur totale d'ordre Δt^2 .

- L'hypothèse d'être globalement Lipschitz est très forte et permet d'obtenir facilement l'existence d'une solution globale en temps. On n'utilise en fait dans la preuve que le caractère Lipschitzien autour de la solution.
- En pratique, e^{Lt} peut être énorme. La méthode d'Euler est imprécise en temps long, mais la dynamique sous-jacente potentiellement aussi... systèmes chaotiques, effet papillon.
- La stabilité de l'EDO n'implique pas la stabilité du schéma numérique. Par exemple, le schéma multi-pas

$$x_{n+2} + 4x_{n+1} - 5x_n = \Delta t f(x_n)$$

est consistant, mais instable : si on prend $f = 0$, on voit que la solution est donnée par $\alpha + \beta(-5)^n$. Si la condition initiale (x_0, x_{-1}) est entachée d'une erreur d'ordre h , β sera d'ordre h , et l'erreur $h(-5)^n$ ne tend pas vers 0 pour tout $n = 1, \dots, T/h$.

- En pratique :
 - (1) Ne pas utiliser la méthode d'Euler, utiliser une méthode d'ordre plus élevé
 - (2) Utiliser une méthode à pas variable
 - (3) Ne pas coder à la main (risque d'erreur)
- Notion de raideur (stiffness) : $\dot{x} = -\lambda x$ avec $\lambda > 0$, solution explicite avec la méthode d'Euler : besoin de $\Delta t \leq \frac{2}{\lambda}$ pour ne pas avoir d'explosion. Extension à $\dot{x} = Ax$. Problèmes pratiques avec les oscillations/amortissements rapides (eg EDP). Méthode d'Euler implicite $x_{n+1} = x_n + \Delta t f(x_{n+1})$. Difficultés d'implémentation.

TD/TP.

- (1) Refaire l'analyse de la méthode d'Euler à la main quand $f(x) = Ax$
- (2) Refaire l'analyse pour la méthode de Heun

$$\begin{aligned} \tilde{x}_{n+1} &= x_n + \Delta t f(x_n) \\ x_{n+1} &= x_n + \Delta t \frac{f(x_n) + f(\tilde{x}_{n+1})}{2} \end{aligned}$$

- (3) Coder la méthode d'Euler et de Heun sur l'oscillateur harmonique $\ddot{x} + \omega^2 x = 0$, valider son ordre.
- (4) Interpréter et simuler les équations proie-prédateur de Lotka-Volterra

$$\begin{aligned}\dot{x} &= x(\alpha - \beta y) \\ \dot{y} &= y(\delta x - \gamma)\end{aligned}$$

Calculer le point fixe non trivial, estimer la période des solutions par linéarisation, et l'utiliser pour vérifier votre implémentation.

- (5) Faire la même chose avec l'équation d'un pendule $\ddot{x} + \omega^2 \sin x = 0$. Coder la méthode d'Euler. Comparer son efficacité (erreur par rapport au nombre d'évaluation du membre de droite) par rapport à `scipy.integrate.solve_ivp`. Tracer la courbe de la période en fonction de l'amplitude initiale (comment?).

9. INTRODUCTION AUX EDP

9.1. Modélisation. Dérivation physique des équations de transport, de la chaleur, des ondes. Équation de la chaleur et mouvement brownien. Gravitation/électrostatique et équation de Poisson.

9.2. Classification. Au contraire des EDO, les EDP ne possèdent pas de théorie systématique. On se limite aux EDP linéaires à coefficients constants.

Ordre 1 :

$$a\partial_x u + b\partial_y u = 0$$

Adimensionalisation. Modélisation : transport. Solution explicite. Condition initiale/aux limites.

Ordre 2 :

$$a\partial_{xx}u + 2b\partial_{xy}u + c\partial_{yy}u = 0$$

Formellement, $\nabla^T A \nabla u = 0$ avec $A = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$. On peut diagonaliser A et obtenir ses valeurs propres λ (≥ 0 sans perte de généralité) et μ réels, ce qui permet de classer, par analogie avec les courbes quadriques, en

— $\lambda > 0, \mu > 0$: elliptique (type Laplace/Poisson)

$$\partial_{xx}u + \partial_{yy}u = 0$$

— $\lambda > 0, \mu < 0$: hyperbolique (type ondes)

$$\partial_{yy}u = \partial_{xx}u$$

— $\lambda = 0$: parabolique (type chaleur)

$$\partial_y u = \partial_{xx}u$$

Dans les cas des équations de transport, d'ondes et de la chaleur, on voit y comme une variable temporelle. C'est le type des équations qui dicte les conditions initiales/aux limites qu'il faut mettre pour obtenir une unique solution; il n'est par exemple pas naturel de considérer y comme une variable temporelle d'une équation elliptique, car il faudrait dans ce cas donner une condition initiale et une condition finale.

On peut combiner ces types d'équations. Par exemple, réaction-advection-diffusion :

$$\partial_t u = \alpha \partial_{xx}u + \beta \partial_x u + f(u)$$

9.3. Solution par Fourier. Une méthode très puissante pour la résolution d'EDP linéaires à coefficients constants est l'utilisation des séries (dans le cas d'un intervalle borné) ou de la transformée (dans le cas de l'espace entier) de Fourier. Dans le cas d'un intervalle borné, il faut que les conditions au bord s'y prêtent, et on considérera ici des conditions aux limites périodiques. D'autres transformations dérivées de Fourier peuvent être pertinentes pour d'autres types de conditions aux limites. Pour des domaines multidimensionnels plus compliqués, les exponentielles complexes doivent être remplacés par les modes propres, ce qui sort du cadre de ces notes.

Considérons par exemple l'équation de la chaleur

$$\partial_t u = \partial_{xx} u, \quad u(0, t) = u(2\pi, t), \partial_x u(0, t) = \partial_x u(2\pi, t), u(x, t=0) = u_0(x).$$

On suppose par simplicité que la condition initiale u_0 est C^∞ et 2π -périodique. On a alors

$$u_0(x) = \frac{1}{\sqrt{2\pi}} \sum_{n \in \mathbb{Z}} c_n(0) e^{inx},$$

et $c_n(0)$ décroît plus vite que tout polynôme inverse en $|n|$. Il est naturel de chercher $u(x, t)$ sous la forme

$$u(x, t) = \frac{1}{\sqrt{2\pi}} \sum_{n \in \mathbb{Z}} c_n(t) e^{inx}.$$

À condition que c_n décroisse suffisamment vite (ce que nous vérifions *a posteriori*), on peut dériver sous le signe somme et obtenir

$$\dot{c}_n(t) = -n^2 c_n(0)$$

donc $c_n(t) = e^{-n^2 t} c_n(0)$, et

$$u(x, t) = \frac{1}{\sqrt{2\pi}} \sum_{n \in \mathbb{Z}} c_n(0) e^{-n^2 t} e^{inx}.$$

ce qui fournit une solution de l'EDP. On voit que les coefficients de Fourier sont atténués, d'autant plus qu'ils sont de fréquence élevée. En temps long, u tend vers la fonction constante (la température s'homogénéise).

Cette résolution explicite est permise par le fait que chaque mode propre e^{inx} soit une fonction propre du second membre et donc évolue de façon indépendante. C'est une propriété générale des équations linéaires invariantes par translation.

On pourrait aussi utiliser le théorème de la convolution pour reformuler $u(x, t)$ comme la convolution de u_0 et d'une certaine fonction, la fonction de Green (ici, une gaussienne périodisée de largeur proportionnelle à $\sqrt{t}$).

TD/TP.

- (1) Faire la même étude avec l'équation de transport.
- (2) De même avec l'équation de Schrödinger $i\partial_t \psi = \partial_x^2 \psi$. Parmi les équations ci-dessus, de quelle équation est-elle la plus proche ?
- (3) Résoudre numériquement l'équation de la chaleur sur $[0, 2\pi]$ avec conditions périodiques et condition initiale $e^{\cos x}$
 - (a) Montrer que la formule de quadrature

$$\int_0^{2\pi} f(x) dx \approx \frac{2\pi}{N} \sum_{k=0}^{N-1} f\left(2\pi \frac{k}{N}\right)$$

est exacte quand $f(x) = e^{inx}$ avec $|n| < N$.

- (b) Estimer la décroissance des coefficients de Fourier de $e^{\cos x}$.
- (c) Dédire de ces deux points un ordre de grandeur du nombre de points nécessaires pour obtenir une approximation numérique précise à 10^{-6} près du n -ième coefficient de Fourier de $e^{\cos x}$. Comparer cet ordre de grandeur à celui donné par l'estimation naïve en $O(1/N)$ de l'erreur de quadrature de la méthode de Riemann.
- (d) Résoudre numériquement l'équation de la chaleur avec $u_0(x) = e^{\cos x}$. Visualiser l'évolution de la solution et de ses coefficients de Fourier au cours du temps. Vous pouvez utiliser les fonctions matplotlib `pcolormesh` (pour visualiser $u(x, t)$ en couleur), `plot_surface` (pour visualiser $u(x, t)$ en 3D), et `FuncAnimation` (pour animer $u(x)$ au cours du temps).
- (e) Que se passe-t-il si vous essayez de résoudre l'équation de la chaleur en temps négatif?
- (f) Que se passe-t-il si vous ajoutez un terme en $\partial_x u$ à l'EDP?

10. EDP ELLIPTIQUES

L'EDP elliptique la plus simple est

$$(6) \quad -u''(x) = f(x)$$

$$(7) \quad u(0) = u(1) = 0$$

avec f continue (par exemple). C'est en fait une équation différentielle (une seule variable), mais de type "boundary value problem" (BVP) où les valeurs au bord sont données, plutôt que "initial value problem" (IVP, problème de Cauchy), où la condition initiale est donnée. Cela la rend plus proche d'une EDP que d'une EDO.

10.1. Caractère bien posé. Dans ce cas très précis, l'équation est facile à résoudre explicitement (au sens où elle se ramène à un calcul de primitive) : on trouve une primitive seconde $\tilde{u}(x)$ de $f(x)$, et la solution générale est donc $u(x) = \tilde{u}(x) + a + bx$. Les conditions $u(0) = u(1) = 0$ donnent un système de deux équations linéaires indépendantes pour a et b , qui peuvent donc être résolues. On obtient immédiatement par cette méthode que u est C^2 , et même, si f est C^k , C^{k+2} (en dérivant l'équation k fois).

Noter que cette méthode est très spécifique à cette équation, et ne peut pas par exemple traiter d'équation de type $-u'' + u = f$ ou $-\Delta u = f$ en plusieurs dimensions. Dans un cadre plus général, on peut utiliser des méthodes variationnelles. L'unicité est facile à obtenir par un argument d'énergie : si u et v sont deux solutions, alors $w = u - v$ satisfait

$$(8) \quad -w'' = 0, w(0) = w(1) = 0.$$

On multiplie par w et on intègre, ce qui donne $\int w'^2 = 0$, donc $w' = 0$, ce qui combiné avec les conditions au bord donne $w = 0$. L'existence est plus subtile et peut être obtenue par minimisation de $\frac{1}{2} \int u'^2 - \int u f$ dans l'espace $H_0^1(0, 1)$ (ou le théorème de Lax-Milgram, une généralisation de cette approche), ce qui sort du cadre de ces notes.

10.2. Discrétisation. Cette équation peut être discrétisée de nombreuses manières ; on considère ici les différences finies, qui consistent à poser un maillage $x_n = n/N$ de pas $h = 1/N$ et à chercher $U_N = (u_n)_{n=1, \dots, N}$ tel que

$$(9) \quad \frac{-u_{n+1} - u_{n-1} + 2u_n}{h^2} = f(x_n),$$

avec la convention $u_0 = u_N = 1$. Cette méthode se reformule également en

$$(10) \quad A_N U_N = F_N$$

avec $(F_N)_n = f(x_n)$ et A_N une matrice tridiagonale avec $-1/h^2$ sur la sous et sur diagonale et $2/h^2$ sur la diagonale.

10.3. **Analyse d'erreur.** L'objectif est de montrer une borne sur l'erreur

$$(11) \quad E_N = U_N - P_N(u),$$

où P_N est l'opérateur d'évaluation sur la grille : $(P_N(u))_n = u(x_n)$.

La seule chose qu'on connaisse sur U_N est $A_N U_N$, donc on écrit

$$(12) \quad A_N E_N = A_N U_N - A_N P_N(u)$$

$$(13) \quad = F_N - A_N P_N(u)$$

C'est l'équation centrale : elle énonce que l'erreur E_N vérifie une équation du même type que U_N , mais avec en membre de droite le résidu de la solution exacte dans le schéma numérique (l'analogue de ce que l'on avait appelé l'erreur de troncation locale pour les équations différentielles ordinaires). Nous avons déjà croisé ce type de relation "erreur-résidu", qui jouent un rôle fondamental en analyse numérique.

On va procéder par la méthode de consistance et stabilité : on montre que le résidu est petit (consistance) et que cela implique que E_N est petit (stabilité).

Consistance. Le résidu est relativement aisé à borner : si on suppose par exemple que f est C^2 , alors u est C^2 , et on a

$$(14) \quad f(x_n) = -u''(x_n) = \frac{-u(x_{n+1}) - u(x_{n-1}) + 2u(x_n)}{h^2} + O(h^2),$$

où le O est uniforme en n : explicitement, il existe une constante de consistance C_1 indépendant de N tel que $\|F_N - A_N P_N(u)\|_\infty \leq C_1 h^2$.

Stabilité. On a vu par diagonalisation explicite de A_N sur les modes $\sin(\frac{\pi kn}{N})$ que sa plus petite valeur propre $\frac{1}{h^2}(2 - 2\cos(\frac{\pi}{N}))$ est bornée inférieurement, indépendamment de N . Comme A_N est symétrique, on a pour tout $V \in \mathbb{R}^{N-1}$

$$(15) \quad A_N^{-1}V = \sum_{k=1}^{N-1} \frac{1}{\lambda_k} \langle P_k, V \rangle P_k$$

avec P_k une base orthonormée de vecteurs propres de A_N associés aux valeurs propres λ_k . Cela implique par Pythagore que

$$(16) \quad \|A_N^{-1}V\| = \sqrt{\sum_{k=1}^N \frac{1}{\lambda_k^2} \langle p_k, V \rangle^2} \leq \min_k \frac{1}{|\lambda_k|} \|V\| \leq C_2 \|V\|$$

avec C_2 une constante de stabilité ne dépendant pas de N .

Convergence. On a enfin

$$(17) \quad \|E_N\| \leq \|A_N\|_{\text{op}} \|F_N - A_N U\| \leq C_2 \sqrt{N} \|F_N - A_N U\|_\infty \leq C_1 C_2 \sqrt{N} h^2$$

soit, explicitement,

$$(18) \quad \sqrt{\frac{1}{N} \sum_{n=1}^N |u_n - u(x_n)|^2} \leq C_1 C_2 h^2.$$

C'est une estimation d'erreur en norme L^2 discrète.

Commentaires :

- L'estimation d'erreur est asymptotiquement optimale (on ne s'attend pas à un meilleur ordre en général)
- Elle ne donne pas une estimation asymptotiquement optimale de $|u_n - u(x_n)|$ ponctuellement ($\sqrt{N}h^2$ au lieu de h^2) : pour cela, il faut travailler en norme ∞ , donc borner la norme de A_N^{-1} dans cette norme : cela peut se faire avec un principe du maximum discret

- La notion de stabilité du schéma est que, si $A_N U_N = F_N$ et $A_N \tilde{U}_N = \tilde{F}_N$ avec F_N et $\tilde{F}_N$ proches, alors U_N et $\tilde{U}_N$ sont proches. Cette stabilité existe également au niveau continu.
- Faire l'analyse numérique nécessite de 1) établir que l'équation initiale est bien posée, et en particulier que sa solution dépend de façon continue des données (stabilité continue) 2) montrer que la discrétisation respecte suffisamment la structure du problème pour qu'on puisse prouver la stabilité au niveau discret 3) montrer la consistance.

TD/TP

- (1) Refaire la même analyse pour des conditions au bord périodiques $u(0) = u(1), u'(0) = u'(1)$.
- (2) On considère un schéma général

$$u''(x_n) \approx \frac{\alpha u(x_n) + \beta(u(x_{n+1}) + u(x_{n-1})) + \gamma(u(x_{n+2}) + u(x_{n-2}))}{h^2}.$$

- Montrer que le schéma est consistant si $\alpha + \beta + \gamma = 0, \alpha + \beta + 4\gamma = 2$
- Montrer que, si on utilise ce schéma pour discrétiser l'équation $-u'' = f$ sur $[0, 1]$ avec conditions au bord périodiques, alors les valeurs propres de A_N sont données par

$$\lambda(k) = \frac{\alpha + 2\beta \cos(k) + 2\gamma \cos(2k)}{h^2}$$

avec $k \in \frac{2\pi\mathbb{Z}}{N}$

- Montrer qu'il existe des valeurs de α, β, γ pour lesquelles $\lambda(k)$ n'est pas nécessairement positive sur $\mathbb{R}$. En déduire que le schéma de différences finies peut être instable.
- (3) Refaire la même analyse pour $-u'' + u = f$ avec f de classe C^2 , en admettant que cette équation admette une unique solution de classe C^4 . Même question pour $-u'' + g(x)u = f$, avec $g(x)$ positive, de classe C^∞ et régulière. Même question pour $-u'' + u' = f$, en discrétisant u' par des différences finies centrées $u'(x) \approx (u(x+h) - u(x-h))/(2h)$. Pour cette dernière question, on pourra montrer et utiliser le théorème de Lax-Milgram discret : si S est symétrique définie positive avec $S \geq \alpha$ au sens des matrices symétriques et A antisymétrique, alors $S + A$ est inversible et $\|(S + A)^{-1}\|_{\text{op}} \leq 1/\alpha$
 - (4) Montrer l'unicité de solutions de l'équation $-u'' + u = f$ avec conditions de bord de Neumann $u'(0) = u'(1) = 0$. Proposer une méthode de résolution numérique et montrer la convergence. Que dire de l'équation $-u'' = f$ avec ces conditions de bord ?
 - (5) Résoudre analytiquement et numériquement l'équation $-u'' = 1$ avec conditions au bord de Dirichlet sur $[0, 1]$. Calculer et tracer quelques modes propres du Laplacien. Mêmes questions pour $-\Delta u = 1$ sur $[0, 1]^2$. Pour simplifier la manipulation d'indices, on pourra utiliser des tableaux quadri-dimensionnels et la fonction `reshape`.
 - (6) Empiriquement, comment se comporte la solution de $-\Delta u + \alpha \chi_D(x)u = f$ quand α devient grand, si χ_D est l'indicatrice d'un sous-ensemble du domaine de résolution de l'équation ? En utilisant cette méthode, calculer des approximations des fonctions propres du Laplacien sur un domaine en forme de L .
 - (7) Résoudre numériquement l'équation

$$(19) \quad -\varepsilon u'' + u' = 1, \quad u(0) = 0, u(1) = 0.$$

Que constatez-vous ? Comment faut-il choisir Δx par rapport à ε ?

11. EDP D'ÉVOLUTION

On considère une équation aux dérivées partielles d'évolution, dont le prototype est l'équation de la chaleur

$$(20) \quad \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

toujours sur $[0, 1]$, avec des conditions au bord de Dirichlet et une condition initiale u_0 régulière (par exemple C^∞). Cette équation est bien posée et la solution est régulière : cela peut se voir ici avec une décomposition en série de Fourier (sur les modes $\sin(n\pi x)$). Dans le cadre général, il existe beaucoup de méthodes pour montrer le caractère bien posé (calcul fonctionnel des opérateurs auto-adjoints, théorie des semi-groupes). On a notamment

$$(21) \quad \frac{d}{dt} \|u\|_{L^2(\Omega)}^2 = 2 \int_{\Omega} u \Delta u = -2 \int_{\Omega} |\nabla u|^2 \leq 0.$$

Il s'ensuit que la norme L^2 de u ne peut que décroître au cours du temps, ce qui traduit le caractère dissipatif de l'équation (et donc, l'équation étant linéaire, une certaine forme de stabilité : si u_0 et v_0 sont proches, alors $u(t)$ et $v(t)$ le sont).

Cette équation nécessite une discrétisation en espace et en temps. On commence par étudier l'équation semidiscrete

$$(22) \quad \frac{dU_N}{dt} = A_N U_N, \quad (U_N(0))_n = u_0(x_n)$$

avec les mêmes notations que précédemment.

11.1. Discrétisation en espace. Comme précédemment, on écrit une équation pour l'erreur $E_N(t) = U_N(t) - P_N(u(t))$:

$$(23) \quad \frac{dE_N}{dt} = A_N U_N - P_N(u'')$$

$$(24) \quad = A_N E_N + \underbrace{A_N P_N(u) - P_N(u'')}_{r_N(t)}$$

C'est encore une relation erreur-résidu : cette fois, le résidu apparaît comme terme source de l'équation. On utilise le principe de Duhamel (variation de la constante) pour obtenir, comme $E_N(0) = 0$,

$$(25) \quad E_N(t) = \int_0^t e^{-A_N(t-t')} r_N(t') dt'$$

Comme A_N est positive définie, $\|e^{-A_N t}\|_{\text{op}} \leq 1$ (le montrer par une diagonalisation de A_N en base orthonormée) et on a donc

$$(26) \quad E_N(t) \leq t \sup_{t' \in [0, t]} \|r_N(t')\|$$

(une analyse plus fine montrerait qu'on peut remplacer le facteur t par une constante). Il ne reste plus qu'à borner $\|r_N(t)\|$, ce qui se fait comme précédemment.

11.2. Discrétisation en temps. Il reste maintenant à discrétiser en temps l'équation

$$(27) \quad \frac{dU_N}{dt} = A_N U_N, \quad (U_N(0))_n = u_0(x_n).$$

Comme on l'a déjà vu, la matrice A_N a des valeurs propres d'ordre $1/(\Delta x)^2$; plus précisément, sa valeur propre maximale est inférieure à $4/(\Delta x)^2$. C'est donc une équation *raide* (possédant des échelles de temps très différentes). Comme on l'a vu au chapitre EDO, si on discrétise cette équation par une méthode d'Euler, il est nécessaire de prendre un pas de temps

$$\Delta t < \frac{2}{\lambda_{\max}} < \frac{1}{2} (\Delta x)^2$$

pour ne pas que la solution explose. C'est la **condition CFL** (Courant-Friedrichs-Lewy) pour l'équation de la chaleur. Cette condition signifie en particulier que la convergence du schéma total (discrétisé en espace et en temps) ne converge pas inconditionnellement si $\Delta x, \Delta t \rightarrow 0$: il faut que

la convergence se fasse en respectant la condition CFL. Comme vu au chapitre EDO, une alternative est d'utiliser un schéma implicite, qui garantit la convergence si $\Delta x, \Delta t \rightarrow 0$ dans n'importe quel ordre.

TD/TP

— Refaire la même analyse pour l'équation

$$\frac{\partial u}{\partial t} = -\frac{\partial^4 u}{\partial x^4}$$

Que devient la condition CFL ?

— Que se passe-t-il si dans l'analyse d'erreur on utilise la majoration

$$(28) \quad \frac{d\|E_N\|}{dt} \leq \|A_N\| \|E_N\| + \|r_N(t)\|$$

et on essaie d'appliquer un lemme de Gronwall ?

— Refaire la même analyse sur l'équation

$$(29) \quad \frac{\partial u}{\partial t} = u'' + u.$$

L'analyse marche-t-elle sur l'équation suivante ?

$$(30) \quad \frac{\partial u}{\partial t} = u'' + u'$$

— Charger une image grâce à la commande `image.imread` du module `matplotlib`, et calculer la solution de l'équation de la chaleur avec comme condition initiale l'image vue comme une fonction de $[0, 1]^2$ dans $\mathbb{R}$. Que constatez-vous ?